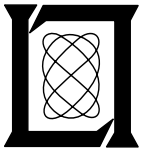
Project Report
ATC-220

Documentation of 9-PAC Beacon Target Detector Processing Function

J. L. Gertz
G. R. Elkin

26 July 1994

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS



Prepared for the Federal Aviation Administration,
Washington, D.C. 20591

This document is available to the public through
the National Technical Information Service,
Springfield, VA 22161

This document is disseminated under the sponsorship of the Department of Transportation in the interest of information exchange. The United States Government assumes no liability for its contents or use thereof.

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
ATC-220		

4. Title and Subtitle	5. Report Date
Documentation of 9-PAC Beacon Target Detector Processing Function	26 July 1994
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Jeffrey L. Gertz and Gabriel R. Elkin	ATC-220

9. Performing Organization Name and Address	10. Work Unit No. (TRAIS)
Lincoln Laboratory, MIT P.O. Box 73 Lexington, MA 02173-0073	11. Contract or Grant No. DTFA01-93-Z-02012

12. Sponsoring Agency Name and Address	13. Type of Report and Period Covered
Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington, DC 20591	Project Report
	14. Sponsoring Agency Code

15. Supplementary Notes

This report is based on studies performed at Lincoln Laboratory, a center for research operated by Massachusetts Institute of Technology. The work was sponsored by the Air Force under Contract F19628-90-C-0002.

16. Abstract

This project report documents the algorithms and flow of the Beacon Target Detector (BTD) processing function incorporated as part of the ASR-9 Processor Augmentation Card (9-PAC). The BTD function combines replies that arise from the same aircraft to form beacon targets, and sends these beacon targets to the 9-PAC merge process where they are combined with primary radar reports.

The 9-PAC BTD process was designed to solve two problems with the ASR-9 Array Signal Processor (ASP) BTD: identifying and removing false beacon targets due to reflections, and preventing merging or splitting of targets due to reply overlap and garble.

The BTD reflection processing algorithm marks each beacon target as either real or false, and provides this information to the 9-PAC merge process. Discrete Mode A reflection false targets are identified when duplicate code reports satisfying stringent conditions are located. In order to find non-discrete Mode A code reflection false targets, the BTD builds an automated, dynamic reflector database based on the geographhy of pairs of discrete real and false targets.

17. Key Words		18. Distribution Statement
9-PAC	ASR-9	This document is available to the public through the National Technical Information Service, Springfield, VA 22161.
radar	false targets	
beacon processing		

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	111	

FORM DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

List of Figures v

List of Tables vii

1. INTRODUCTION 1

2. REPLY PROCESSING 3

2.1 BTD Task 3
2.2 Process Sweep 5
2.3 Process Open Groups 10

3. TARGET FORMATION 13

3.1 Process Mature Group 13
3.2 Find Tracks Near Group 14
3.3 Form Target Reports 15
3.4 Perfect 16
3.5 Perfectible 18
3.6 Two-Track Track Matching 19
3.7 Single-Track Track Matching 24
3.8 Parse 27
3.9 Target Range 33
3.10 Target Azimuth 34
3.11 Code Match 35
3.12 Code Match With One Bit-Drop 36
3.13 Target Altitude With Track 37
3.14 Target Altitude Without Track 41
3.15 One-Timer 44
3.16 Reply Garble 46
3.17 Wide-Pulse Target 49
3.18 Process Target Report 51
3.19 Military Ident & Emergency Target 53

4. REFLECTION PROCESSING 55

4.1 Building Reflector Database 55
4.2 Discrete Reflection 58
4.3 Non-discrete Reflection 60

5. INTERNAL TRACKING 63

5.1 Tracking Overview 63
5.2 Track Association/Initiation 66
5.3 Track Update 69

6. SYSTEM PROCEDURES 77

APPENDIX A 79

APPENDIX B 91

APPENDIX C 97

APPENDIX D 101

LIST OF FIGURES

Figure No.		Page
1	BTD Processing Task	79
2	Process Sweep Routine	80
3	Process Open Groups Routine	81
4	Process Mature Groups Routine	82
5	Form Target Reports Routine	83
6	2-Track Track Match Algorithm	84
7	1-Track Track Match Algorithm	85
8	Parse Algorithm	86
9	Process Target Report Routine	87
10	Target-Track Association	88
11	Discrete Reflection Processing	89
12	Non-Discrete Reflection Processing	90

LIST OF TABLES

Table No.		Page
1	Reply Buffer Data Structure	7
2	Range Count Data Structure	8

1.0 INTRODUCTION

This project report documents the algorithms and flow of the Beacon Target Detector (BTD) processing function incorporated as part of the ASR-9 Processor Augmentation Card (9-PAC). The BTD function within the 9-PAC receives beacon sweep replies from the Beacon Reply Processor (BRP) hardware in the ASR-9 secondary radar processing module. The BTD function combines replies that arise from the same aircraft to form beacon targets, and sends these beacon targets to the 9-PAC Merge process where they are combined with primary radar reports.

The BTD process in the 9-PAC is very different from the original ASR-9 BTD which runs in the ASR-9 Array Signal Processor (ASP). In fact, the 9-PAC BTD process was designed specifically to solve two problems with the original ASP BTD. The first issue was that the ASR-9 had no mechanism for identifying and removing false beacon targets due to reflections, especially noticeable at Los Angeles International Airport (LAX), where several large reflecting surfaces exist near the radar. The second issue was that beacon targets were prone to merging or splitting due to reply overlap and garble, particularly on approach to parallel runways, such as at St. Louis (STL) Airport.

Using the significantly increased memory capacity and processing speed of the 9-PAC board, the 9-PAC BTD is able to implement more advanced algorithms that address both of the problems just described. In particular, the added capabilities permits the 9-PAC BTD to maintain an internal track file and a dynamic reflector database to facilitate reflection false target processing.

In addition, while the ASP BTD had to assign each beacon reply to a target at its time of reception, the 9-PAC BTD can postpone target formation decisions until all replies in an area are received, and then process the group in a batch mode when all information is available. This makes it possible to degarble reply codes and unravel closely spaced reports, which significantly reduces the likelihood of target merging or splitting.

The BTD reflection processing algorithm marks each beacon target as either real or false, and provides this information to the 9-PAC Merge process. Discrete Mode A reflection false targets are identified when duplicate code reports satisfying stringent conditions are located. In order to find non-discrete Mode A code reflection false targets, the BTD builds an automated, dynamic reflector database based on the geography of pairs of discrete real and false targets.

The BTD identifies garbled Mode A and altitude reply codes based on the range separation of the set of replies to a given interrogation sweep. The BTD performs code degarbling on such replies whenever possible using its internal beacon track file. It also eliminates duplicate targets arising from wide reply pulses.

The remainder of this report is organized as follows. Sections 2 through 5 present the details of 9-PAC BTD algorithms. Section 6 describes the 9-PAC BTD system initialization and reset procedures. Appendix A contains the algorithm flow charts that are referenced in the detailed algorithm descriptions. Appendix B presents the Variable Site Parameters (VSP) that can be edited and sent to the 9-PAC via the ASR-9 Remote Monitoring System (RMS). Appendix C presents the Performance Monitor (PM) statistics sent by the 9-PAC to the RMS at the end of each antenna scan. Finally, Appendix D is a glossary of terms used in the detailed algorithm descriptions.

2.0 REPLY PROCESSING

2.1 BTD TASK

Inputs: Buffer of Beacon Reply Sweep(s).

Outputs: Beacon Target Reports to Merge Task.

Algorithm:

The Beacon Target Detector (BTD) Processing Task (see flowchart in Figure 1) is awakened whenever beacon reply data arrives from the ASR-9 ASP. Reply data consists of a 2 word interrogation (sweep) header providing the sweep azimuth and mode, followed by 2 words per reply indicating the reply range and code.

2.1.1 Input Parsing

Since it can not be assumed that data will arrive at a consistent rate, the BTD Task maintains an input state machine in which the expected type of the next valid data word is stored. This way, the BTD works whether a data packet consists of an entire sweep, multiple sweeps, or only a fraction of a sweep.

Each sweep header is checked to make sure the antenna azimuth is increasing by reasonably small positive steps (taking special care to handle the north mark appropriately). If the antenna azimuth skips too far or fails to move forward, the BTD Task sets a Performance Alarm and calls the BTD Initialization/Reset algorithm.

Each reply on a given sweep is checked to ensure that the ranges are increasing. If not, it is assumed that there are bit errors in the input data, and the entire sweep is discarded.

When a new sweep header is parsed, the previous sweep is processed, as described below.

2.1.2 Outputting Targets to Merge Task

If the azimuth of the new sweep to be processed has crossed a 16 ACP boundary (i.e., 0, 16, 32, ...), the current list of completed beacon targets is output to the Merge Task, where the Radar/Beacon Merge occurs.

2.1.3 Updating Internal BTD Tracks

If the sweep being processed has crossed a 30 degree boundary (wedge), the tracks in the 30 degree wedge that is 180 degrees behind the current sweep's wedge are placed on an update track list. The BTD Failsoft algorithm ensures that tracks on this list are updated in a timely manner.

2.1.4 Outputting Performance Monitor Data

If the sweep being processed represents a crossing of the north mark, the BTD Performance Monitor Data is output to the 9-PAC Output Task, which sends it to the ASR-9 ASP along with similar data from the Merge Task and other tasks.

2.1.5 Expunging Old Replies from 1 Range Bin

The Range Bin data is used by the Process Sweep and Process Open Groups algorithms. If a given range bin received a single fruit reply which did not join a reply group, it remains in the buffer indefinitely. Therefore, each time a new sweep is processed, a single range bin is checked so that any old reply data can be removed. This is done on a rotating basis (e.g., range bin 4000 last time, range bin 4001 this time).

2.1.6 Sweep Processing

The entire sweep of replies is passed to the Process Sweep algorithm (see flowchart in Figure 2), where the replies are entered into the Reply Buffer and Range Bin table. Also, the Open Group list is updated as necessary.

2.1.7 Open Group Processing

On each sweep, the Open Group list is updated using the Process Open Groups algorithm (see flowchart in Figure 3). Starting with the group at the longest range, each open group is checked to see if it has matured. A mature group presumably has received all the replies that are part of its target or targets.

2.1.8 Mature Group Processing

If any of the Open Groups processed above have matured, then the set of mature groups is passed to the Process Mature Groups algorithm (see flowchart in Figure 4). This routine declares target reports and then calls the BTD Association and Reflection Processing algorithms. Declared target reports are placed on an output list, so that they can be sent to the Merge Task at the appropriate time.

2.2 PROCESS SWEEP

Algorithm Subsystem:	Reply Processing
Inputs:	Single sweep of replies Mode and azimuth (ACP) of the interrogation
Outputs:	Updated Open Group structures

Algorithm:

The Process Sweep routine (see flowchart in Figure 2) handles the replies from a single interrogation sweep, which can be Mode A (identity) or Mode C (altitude). Each reply is entered into the proper Open Group structure, or if no Open Group is applicable, will form a new group or be considered a potential fruit.

If the new sweep has a smaller azimuth than the previously processed sweep, the new sweep is discarded. This avoids data structure inconsistencies that would occur after system recoveries. In addition, if the reply ranges on the sweep are not in increasing order, it is presumed that a bit error has occurred, and for safety the sweep is discarded.

It is highly unlikely that the number of target + fruit replies, even in the worst case, can exceed 42 replies on a sweep. Thus if this number is exceeded, it is probable that a strobe has occurred. Rather than waste time on garbage replies, any sweep having more than 42 replies is automatically dropped. Also, the reply overflow alarm (PM PRTOVFL) is set and sent to the ASPX at the end of the current scan.

2.2.1 Failsoft Provisions

If, because of heavy target loading in a sector, the processing of sweeps falls significantly behind real time, report dissemination will occur too late for Merge to accept. To preclude this event, range truncation of overloaded sectors occurs. A counter of troubled sweeps signals when truncation becomes necessary.

At the start of processing for each sweep, a check is made as to how far sweep processing is behind the real time reception of the sweep. If the sweep processing delay exceeds T_{bad} (nominally 3°), the counter is incremented; if the delay is less than T_{OK} (nominally 1°), the counter is decremented (but not below 0). Should the counter ever reach a value of 3, range truncation is implemented to prevent recurrence of unacceptable delays on subsequent scans.

The set of actions performed at this time are as follows:

1. Any sweep delayed beyond T_{bad} is dropped.
2. The range of the sector in which the triggering sweep resides is reduced to the next lower range quantum.
3. If the azimuth of the sweep is in the first half of the sector, the range of the previous sector is also reduced, unless its range is already below the new range of this sweep's sector.

The set of range quanta used for truncation are:

50,40,35,30,25,20,18,16,14,12,10,9,8,7,6,5,4,3,2,1,0.

Any subsequent sweeps in the same sector on the same scan that are late beyond T_{bad} are simply dropped, with no further range reductions. However, if the problem reoccurs on a subsequent scan, further range reductions are taken.

If a sector that has its range reduced recovers, because of reduced target loads, its range is allowed to drift back toward the normal full-range value. In particular, the range limit of a sector is incremented by 1 mile whenever every sweep in the sector, and every sweep in the next half sector, all have processing delays that are less than T_{bad}.

2.2.2 The Reply Buffer

Each reply on the sweep is processed, in increasing range order. First the reply is entered into a reply buffer, which is a circular buffer containing 10,000 entries. The attributes stored in the reply buffer are described in Table 1. Note that the sweep header is also entered into the reply buffer, immediately in front of the first reply on the sweep. A hdr field in the reply buffer indicates whether an entry is a sweep header or a reply. There is also a special value for the hdr field to indicate the top of the reply buffer. The rngcnt, acp, mode, code, vali, x, and spi fields describe the position and identity of the reply. A time field in ACP units is used to maintain the age of a reply relative to the current sweep. Time is stored in the lower 31 bits of a 32-bit word, so that time wraps around to zero every 2,147,483,647 ACPs, which is approximately 28 days assuming a 4.8 second antenna rotation rate.

There are also two pointer fields in the reply buffer. The bin_ptr contains the reply buffer index of the next reply with the same rngcnt value, if any. The linking of replies at the same range count forms the basis for determining which replies belong in the same group. The grp_ptr contains the reply buffer index of the next reply in the same group, if any. The use of these pointers will be described later in detail.

A reply whose range exceeds 62.5 Nmi is not placed in the reply buffer; instead, such a reply is taken to be a Software Test Reply, and is copied into a special test reply buffer. Test replies are sent back to the 9-PAC Output Task at the end of the sweep.

If the volume of replies exceeds the system design limit, so that a new reply would be copied over a still needed reply in the buffer, the new replies are dropped and not placed in the reply buffer. In this case, a reply overflow alarm (PM PRTOVFL) will be sent to the ASPX at the end of the current antenna scan. An overflow is detected whenever the reply to be overwritten is less than 200 ACPs from the current sweep. Maintaining a 200 ACP reply history provides ample opportunity to allow reply groups to mature.

Table 1. Reply Buffer Data Structure

Attribute	Description
hdr	1 if entry is sweep header, 0 if entry is reply
mode	Reply mode (A, C, or 2)
acp	Sweep azimuth (ACP)
rngcnt	Range count
code	Code value
vali	Code validity from ASR-9 BRP (0, 1, 2, or 3)
spi	SPI value
x	Code X-bit value
bin_ptr	Pointer to next reply at same range count
grp_ptr	Pointer to next reply in same group
time	Time (ACP) of current sweep

2.2.3 Wide Pulse Reply Matching

Next, each reply is checked to see if it is part of a wide-pulse pair caused by an out-of-spec transponder. Such transponders generate a long pulse, causing the ASR-9 BRP to declare two replies, generally with the same code, at slightly different ranges.

To detect wide pulse reply matches, each reply is compared with the previous reply on the same sweep. If the two replies are at most 10 range counts apart, and have "matching" codes, then a wide pulse match is declared.

The rules for determining a matching code depend on the interrogation mode. For Discrete Mode A codes, the two reply codes must match, except that the longer range code is allowed at most 2 bit drops from the shorter range reply code. For Non-discrete Mode A codes, the general rule is that the two reply codes must be exactly the same, and both must have a 0 or 1 validation flag (i.e., they are un-garbled). However, a special rule is used to permit a wide-pulse match declaration if both replies are in the set {0000, 0200, 1000, 1200}. For Mode C codes, the two reply codes must be exactly the same, and both must have a 0 or 1 validation flag.

2.2.4 Range Counts

The next algorithm step is to determine if the reply being processed belongs to an existing group. Each reply is grouped with replies from other sweeps that have the same range count, using the range count data structure shown in Table 2. It takes two replies at a given range count, within 77 ACPs of each other, to "open" the range count.

When a range count is first opened, the open group list is searched to find an existing group for the range count to join. The range count will join an open group if it is within 4 range counts of the group's current limits. If the range count is found to be between two open groups, and within 4 range counts of each group, the two groups are combined into a single group. If the search fails, a new open group is created, with the new group's starting time set to the time (ACP) of the reply.

Table 2. Range Count Data Structure

Attribute	Description
first	Reply buffer index of oldest reply at range count
first_time	Time of oldest reply at range count
last	Reply buffer index of newest reply at range count
last_time	Time of newest reply at range count
count	No. of replies at range count
mode_a_count	No. of Mode A replies at range count
mode_c_count	No. of Mode C replies at range count
wp_a_disc_count	No. of Disc A wide pulse matches at range count
wp_a_nond_count	No. of Nondisc wide pulse matches at range count
wp_c_count	No. of Mode C wide pulse matches at range count

If the reply range count is already open and therefore already part of an open group, the reply simply updates the group's ending time.

Wide-Pulse reply matches for an open range count are added to comprise the total matches of the open group. This statistic is used later in the Process Open Group algorithm.

2.2.5 Military Ident and Emergency Pre-processing

The next step of reply processing is to determine if the reply is potentially part of a Military Identification or Military Emergency message. This military preprocessing step only applies to Mode A replies.

A Military Identification consists of two replies, with the longer range (second) reply in the SPI position of the shorter range (first) reply. The code of the second reply is either zero or the same as the first reply. Each reply that is part of a Military Identification is flagged as either the "first" or "other" reply. The code of the first reply is saved in the "military code" field.

A Military Emergency consists of four replies, with each reply in the SPI position of the preceding reply. The codes of the second, third, and fourth replies are either zero or the same as the first reply. Either the second or the third reply may be missing, but not both. Each reply that is found to be part of a Military Emergency is flagged as either the "first" reply or an "other" reply. The code of the first reply is saved in the "military code" field.

A reply that cannot find other replies that together satisfy either the Emergency or Ident range test is labeled as a "military failure" reply. A reply that found other replies in the correct position(s) but failed on code, such as because of the presence of garble, is considered neither a failure nor any type of success.

To fully implement the search for military ident and military emergency situations, every reply should be tested against every other reply on the same sweep for the characteristic range spacings. Unfortunately, the processing overhead incurred by these tests would be a significant fraction of the total BTD load. In order to reduce processor utilization and preclude range truncation in heavy traffic periods, a reduced checking scheme has been implemented.

Half or more of all replies received by the BTD are fruit. In order to eliminate military processing of fruit replies, the first rule is that only replies that are part of open range bins are tested. This has the possible adverse side effect of reducing the number of replies in a real target that get tested, as the first reply on every range bin of the target will be skipped. However, the number of replies remaining are more than enough to establish the existence of the military situation.

In fact, so many replies remain to be tested that the second rule is that if the first 5 tested Mode A replies of a target fail to indicate the presence of a military ident or emergency situation, no further tests are made unless a new range bin is opened, in which case its opening reply is tested. Since the military situation should occur on every Mode A sweep, the absence of any evidence 5 times in a row should be sufficient to conclude the situation does not exist. The test on a new range bin covers the possibility that a second target joins the open group. If any reply test indicates the possible presence of a military ident or emergency situation, all future replies in the group are tested.

To implement this test, each open group maintains counts for military failures, military ident first replies, military emergency first replies, and military other replies. The group also stores the value of the last military code, and the azimuths of first military success and last military success. These statistics are used later in the Military Ident and Emergency Target algorithm.

2.3 PROCESS OPEN GROUPS

Algorithm Subsystem:	Reply Processing
Inputs:	List of Open Reply Groups ACP of newest sweep processed.
Outputs:	List of mature reply groups, if any, each with military and wide-pulse results.

Algorithm:

The Process Open Groups algorithm (see flowchart in Figure 3) processes a range-ordered list of open groups, starting with the group at the longest range, to determine if any group has matured. Once a group has matured, it is passed to the BTD Target Formation algorithm subsystem.

2.3.1 Merging Wide-Pulse Groups

If two groups at slightly different ranges are believed to have come from a single aircraft with an out-of-spec wide-pulse transponder, then the two groups are merged.

In order for two groups to be merged, the longest range reply of the shorter-range group must be no more than 10 range counts from the shortest-range reply of the longer-range group, and the longer range group must have either:

(a) ≥3 Discrete Mode A wide-pulse reply matches, or

(b) ≥3 Non-discrete Mode A wide-pulse reply matches and

≥1 Mode C wide-pulse reply match.

2.3.2 Group Maturity Test

The maturity test for an open group is based on two quantities:

E = ACPs elapsed since the group was opened,

G = ACPs elapsed since a reply was assigned to the group.

An open group is considered mature when both of the following conditions are satisfied:

(a) $E \geq 55$ ACP, and

(b) $G \geq (20 - (E - 66)/5)$.

The reduction of G when a group has existed for greater than 66 ACPs insures that a group will close in time to meet the maximum delay criterion even if several targets are strung together in azimuth.

When a group is declared mature, its first and last times are set for use by later functions. The first time is the time of the first reply in the range bin that originally opened the group; the last time is the time of the last reply in any of the group's open range bins.

2.3.3 Single Mode Groups

If a group has matured, and all of the replies in a group are of the same mode (Mode A only or Mode C only), and there is another group within 10 range counts that is a single mode of the opposite type, then the two groups are merged. This action compensates for aircraft whose transponder inter-mode delay is out of spec, causing the A and C modes to appear at different ranges.

The quantities E and G are re-computed for the new merged group, and the group maturity test is repeated. It is possible that the merged group might not yet be mature.

2.3.4 Extending Groups

An open group consists of a list of open range bins, each of which by definition has received two or more replies. When the group has matured, any additional bins within 4 range counts of the group range limits that have only 1 reply are considered for addition to the group. In order for such a reply to be added to the group, its time must be "compatible" with the group times, defined as follows:

1. The reply time must not exceed the greater of

   the group first time + 55, or
   the group last time + 10

2. The reply time must equal or exceed the lesser of

   the group last time - 55, or
   the group first time - 10

The exception to this rule is that any one-hit range bin that is closer to another group is not assigned to the group being processed; intrusion into another group's space is not permitted.

For a group that has only Mode A replies, the 4 range count extension limit is relaxed (subject to the non-intrusion rule) in an attempt to find Mode C replies that were separated from the Mode A replies by the transponder out of spec problem noted above. This action covers the case when these Mode C replies failed to form their own open group because no range bin had 2 replies.

First, if any Mode C replies are encountered in the one-hit range bins being examined during the normal extension process, the extension limit in that direction is set to 3 beyond the first such occurrence. Otherwise, if no Mode C replies are found in the normal extension bins, the limit on each side of the group is increased 1 range count at a time until the limit reaches 10, or until a Mode C reply is encountered, at which time the limit is further increased by 3. In either case, no Mode A replies encountered beyond the normal 4 range count extension limit are added to the group.

2.3.5 Linking Replies

After a mature group has been extended to include one-hit range cells, all of its replies are linked together in time order in the reply buffer. The group is passed on to the target formation algorithms, along with its military and wide-pulse results, when the processing pass through the open groups is complete.

A rare timing event could find a reply to be linked already linked as part of another closed, but not yet processed, group. To prevent catastrophic linked list errors, such replies are excluded from being linked with the new group.

2.3.6 Held Over Replies

It is possible that some of the single replies in the extension bins really belong to other groups. To prevent such groups from being biased due to loss of some of their replies, certain replies are returned to the bin structure when the Target Formation Algorithms are finished processing the group. These replies are the ones from extension ranges that satisfy either:

(a) located equi-distant between 2 open groups, or

(b) occur within 20 ACPs of the ACP of the sweep currently being processed.

The first case covers ambivalent replies, the second covers replies from groups that are just beginning when the old group closes.

3.0 TARGET FORMATION

3.1 PROCESS MATURE GROUP

Algorithm Subsystem: Target Formation

Inputs: Mature Group of replies.

Outputs: From 0 to 5 Target Reports, each with its set of attributes.

Algorithm:

The Process Mature Group algorithm (see flowchart in Figure 4) takes a mature group of replies and determines how many aircraft target reports are likely contained within the group. The attributes of each target are computed, including range, azimuth, code, altitude, and validity flags.

3.1.1 Removing Azimuth Outliers

First, any azimuth one-timers are deleted from the group (see One-Timers description). These are replies on either end of the group that are more than 1 degree from their nearest neighboring reply in the group.

3.1.2 Rejecting Weak Groups

Next, the group is checked to make sure there are enough replies to possibly cause a target report to be declared. If a group has at most six replies, and it spans more than three degrees (33 ACP) in azimuth, then it is declared a weak group, and no target report is allowed. Also, any group that does not satisfy the minimum reply VSP (MIN_REPLIES) is thrown out.

3.1.3 Finding Nearby Tracks

Next, the BTD Track File is searched for tracks "nearby" the group's position, using the Find Tracks Near Group algorithm. A list of at most 10 nearby tracks is created. The range centroid of the group, computed by averaging all reply ranges in the group, is used as the group's range parameter. Two azimuths are used to define the group's azimuth, a minimum and maximum. If the group is more than 4 degrees wide in azimuth, the azimuths are set to be 1 degree inside of the group min and max azimuth, respectively. If the group spans less than 4 degrees, then the azimuths are set to one degree out from the group's azimuth centroid.

3.1.4 Declaring Target Reports

The actual declaration of the target reports contained within the group of replies is handled by the Form Target Reports algorithm (see write-up). Each declared report is then processed by the Process Target Report algorithm (see write-up).

After the reports are processed, the group, and all its replies, are expunged from their various lists and data structure. The exception is that replies labeled as holdovers are left on their range bin lists for processing by subsequent open groups.

3.2 FIND TRACKS NEAR GROUP

Algorithm Subsystem:	Target Formation
Inputs:	Range Centroid of reply group, Azimuth extent of reply group.
Outputs:	List of tracks near position of given reply group.

Algorithm:

The Find Tracks Near Group algorithm searches the BTD internal track file for tracks that are "near" a given reply group location. The range location of the reply group is given by its range centroid (grp_rng), while the azimuth is assumed to cover the spread from one degree after the first reply in the group (grp_az1) to one degree before the last reply in the group (grp_az2). This spread is necessary to cover the cases where two or more abutting targets are represented by the reply group.

For a track to be considered "near" the reply group, the reply group range must fall within the track's "range box", and some part of the group azimuth extent must fall within the track's "azimuth Box". Mathematically, the latter relationship can be expressed as the track's azimuth prediction must fall within the region defined as:

$$(grp_az1 - track_az_box) \leq trk_az \leq (grp_az2 + track_az_box)$$

The track_range_box and track_az_box quantities are defined in the Track Update algorithm. A newly initiated track, when it is first updated, has its range and azimuth boxes set to cover an area based on a 600 knot maximum velocity. (Initiated tracks not yet updated are not relevant to this search routine.) On subsequent updates, the association boxes are set to one of two values, the smaller for normal cases, and the larger for close-in and coasting tracks.

The BTD track file is set up so that tracks are grouped into geographic "sort boxes". Thus, the task of searching for nearby tracks consists of two steps. The first step is to determine which sort boxes to search; the second step is to check every track in the sort boxes on the search list.

When the BTD is initialized, a "search list" of relevant sort boxes is created for each 1-nmi range by 3-degree azimuth region. For each range/azimuth region, at least one and at most four sort boxes must be searched in order to find reasonable tracks for a given reply group location. Consult the Tracking Overview write-up for the details of this searching mechanism.

3.3 FORM TARGET REPORTS

Algorithm Subsystem: Target Formation

Inputs: Mature Group of replies.

Outputs: From 0 to 5 Target Reports, each with its set of attributes.

Algorithm:

The Form Target Reports algorithm (see flowchart in Figure 5) takes a mature group of replies and determines how many aircraft target reports are likely contained within the group. The attributes of each target are computed, including range, azimuth, code, altitude, and validity flags.

If the group satisfies the Perfect algorithm (see write-up), a single target is declared. If group is not Perfect, then the One-Timer algorithm is used to flag anomalous replies in the group, and the "force" part of the Reply Garble algorithm is used to correct replies marked as garbled by the ASR-9 BRP but whose codes match a nearby track. Now, if the group satisfies the Perfectible algorithm, a single target is declared.

If not Perfectible, the group is checked by the Wide Pulse Target algorithm. If the group is found to be a wide-pulse transponder target, then the longer-range replies on multiple-reply sweeps are removed from the group, and the remaining group is restarted back at the Perfect algorithm.

Next, the remaining steps of the Reply Garble algorithm are used to determine which replies are garbled and which are not.

The next step depends on the number of tracks nearby the group. If there are no tracks near the group, the Parse algorithm is used to determine which targets to declare, if any.

If there is exactly one track near the group, the Single-Track Track Match algorithm (see write-up) is tried, and if it succeeds, a single target report is declared. Otherwise, the Parse algorithm is used to determine which targets to declare, if any.

If there are two or more tracks near the group, the Two-Track Track Match algorithm (see write-up) is tried, and if it succeeds, two target reports are declared with the codes of two of the nearby tracks. If unsuccessful, the Single-Track Track Match algorithm is tried, and finally, if still unsuccessful, the Parse algorithm is used.

3.4 PERFECT

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with ASR-9 BRP garble flag List of nearby tracks
Outputs:	One Beacon Target Report if reply group is Perfect

Algorithm:

The Perfect algorithm determines whether a given group of replies are in perfect agreement, so that the position and Mode A and C codes of the corresponding target is obvious. For a reply group to be called perfect, all of the following conditions must be satisfied:

(a) all replies are high confidence, i.e., "clear"

(b) all Mode A replies have the same code

(c) all Mode C replies, if any, have the same code

(d) no two replies in the group come from the same sweep

(e) the reply ranges span no more than 5 range clocks

(f) the group run length is $\leq$ 77 ACPs

(g) no gap > 11 ACPs (1 deg) exists between adjacent replies

(h) there are sufficient group replies ($\geq$ VSP MINREPLIES)

(i) there are sufficient Mode A replies ($\geq$ VSP MIN3A)

If the group satisfies the Perfect algorithm, a single target report is declared. The target Mode A code is set to the replies common Mode A, and made high confidence (3). The target range and azimuth are determined using the "Target Range" and "Target Azimuth" algorithms respectively.

The target altitude and confidence are determined as follows. If there are no Mode C replies, the target altitude is set to unknown altitude (code 0, confidence 0). If the replies common Mode C is a decodable FL, and there is a nearby matching-code track with an altitude within 2 FL of the group, then the target altitude is set to the reply FL with confidence 3. If the replies common Mode C is Brackets-Only (code 0), and there is a nearby matching-code track with a Brackets-Only altitude, then the target altitude is set to the Brackets-Only special value (7766 Octal) with confidence 3.

If there is no matching-code track, or the matching-code track does not agree with the replies common Mode C, then set the target altitude to

(a) the common reply altitude if it is a decodable FL, or

(b) 7766 Octal if the replies are Brackets-Only, or

(c) 6030 Octal if the common reply altitude is not decodable.

The confidence for the case of no track support is 3 if there are 3 or more Mode C replies, 1 if there are 2 Mode C replies, and 0 if there is 1 Mode C reply.

3.5 PERFECTIBLE

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with garble flag and one-timer flags List of nearby tracks
Outputs:	One Beacon Target Report if group is Perfectible

Algorithm:

The Perfectible algorithm determines whether a given group of replies is Perfect (see Perfect algorithm) after the removal of all replies flagged as one-timer replies (see One-Timer algorithm).

If the group satisfies the Perfectible algorithm, a single target report is declared. The target Mode A and C codes and confidences are determined as in the Perfect algorithm. The target range and azimuth are determined using the "Target Range" and "Target Azimuth" algorithms respectively after replies labeled as azimuth, range, and multiple-reply-sweep one-timers are removed.

3.6 TWO-TRACK TRACK MATCHING

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with garble mask and one-timer flags Group wide-pulse indicator Two Tracks, each with code and altitude
Outputs:	Two target reports if match successful

Algorithm:

When two aircraft are in close proximity, the two sets of replies may combine to form a single reply group. The Two-Track Track Matching algorithm (see flowchart in Figure 6) attempts to resolve this situation by determining when two known tracks are necessary and sufficient to account for the replies in a group.

Should there be more than two nearby tracks for the group, each pair of tracks is tested. The ordering of the tracks is by distance from the reply group, closest first. When a successful pair of tracks is identified, two reports corresponding to these tracks are created. If no pair results in success, Single-Track Track Matching is attempted for the group.

As the number of tracks increases, the number of possible track pairs grows exponentially. To prevent prohibitive real-time processing overloads, whenever more than 4 tracks are near the reply group, a simple pre-test is applied to each pair. In particular, a given pair of tracks will be considered only if, for each track, at least one clear Mode A reply exists that matches the code of the track. In addition, if one of the tracks has a code of 1200, and there are 3 or more tracks with code 1200 in the set of nearby tracks, at least one clear Mode C reply must exist in the group that agrees with the altitude of the 1200 track.

The first step in Two-Track Track Matching is called reply testing, in which each reply is allocated uniquely to one track, or definitely to one track and possibly to the other, or jointly to both tracks, or to neither track. If the two targets are at exactly the same range, there may be a region of overlapping replies such that the reply codes consist of the combined code (or altitude) values of the two targets. These overlapping replies are the ones that can be allocated to both tracks.

After testing all Mode A replies, and possibly Mode C replies, a determination is made as to whether there are sufficient replies to support each track's code. This is called the application test. If the two-track matching algorithm applies to the reply group, the actual reply allocation between the tracks is performed. Reply allocation assigns each reply to either, both, or neither track, starting with the results of the reply testing process, and modifying the allocations as necessary to correctly establish the azimuth extent of the two targets.

If both tracks have a 1200 Mode A code, a sanity test is performed to check if the two targets should be merged into one 1200 code target.

The code of each target is set to the code of the corresponding track. The range, azimuth, and altitude values of each target are determined using the replies allocated to that particular target. If the track has a flight level altitude prediction, the Mode C replies allocated to that track are fed

into the Target Altitude With Track algorithm. Otherwise, the allocated Mode C replies are fed into the Target Altitude Without Track algorithm.

3.6.1 Reply Testing

Mode A replies are always tested, by being compared against the Mode A codes of the two tracks. Mode C replies are only tested when a reference altitude can be determined for each track. For a track to have a valid reference, all of the following conditions must apply:

1. the track must be brackets-only or have a predicted FL,
2. a clear Mode C reply must exist that matches one of the valid track-match altitudes (bracket-only for a brackets-only track, within ±2 FL for a FL track),
3. no clear reply can exist that matches a track-match altitude other than that used in 2.

The reference altitude for the track is the matched level.

If both tracks have the same Mode A codes (such as 1200), the Two-Track Track Matching algorithm fails automatically unless a reference altitude was found for each track, and they are not the same. Instead, the Parse algorithm will be employed to determine whether or not two targets should be declared when altitudes are uncertain.

Each Mode A reply is tested against three codes: the first track's code, the second track's code, and the "or-ing" of the first and second track codes. If Mode C replies are being tested, then each encoded Mode C reply is compared with the first track reference altitude (in its encoded form), the second track reference altitude, and the "or-ing" of the first and second track reference altitudes. Range one-timers and multiple-reply-sweep one-timers are ignored in the testing process.

Mode A replies with code 1200 can match only a track with code 1200. For all other Mode A reply code values, and any Mode C reply value, the Code Match algorithm is used to determine whether a match exists. This algorithm takes into consideration the shorter and longer garble masks for the reply to decide which reply pulses are clear and which are garbled in order to construct the proper match test.

Each tested reply has a label from the set {REQUIRED, OK, FAILED} assigned relative to each of the two tracks. REQUIRED (R) indicates that the track code, either by itself or in concert with the other track code, is needed to generate the observed reply code value. OK (K) indicates that the reply code value could exist whether or not the track code were present. FAILED (F) indicates that the reply code could not have resulted if the track code were present. The Code Match algorithm is used for each test.

The labels are determined as follows based on the result of the three code match tests listed previously. Each row of the table indicates the label assignments for a possible outcome of these tests, where 1=passage of a test and 0=failure of a test:

Code 1	Code 2	C1 or C2	Label 1	Label 2
0	0	0	F	F
0	0	1	R	R
0	1	0	F	R
0	1	1	K	R
1	0	0	R	F
1	1	0	K	K
1	1	1	J	J

Any completely failed reply (F F) is re-tested using the Code Match With One Bit-Drop algorithm.

If three or more tested non-one-timer replies are (F F) with respect to both tracks, the Two-Track Track Matching algorithm fails for this pair of tracks. At this point, the next pair of tracks, if any, is tested.

3.6.2 Multiple-Reply Sweep Allocation

Only one reply per sweep can be REQUIRED with respect to a particular track. Once a REQUIRED reply is found for a track, all longer range replies on that sweep are set to FAILED for the track. If any such reply is OK for the other track, it is set to FAILED for that track if it was originally REQUIRED by the first track; to REQUIRED for the second track otherwise. This step has the effect of eliminating from further consideration replies caused by wide pulses.

3.6.3 The Application Test

The next step is to determine whether or not the reply group sufficiently supports the two track match assumption. For each track, the number of REQUIRED and OK replies are totaled. If the tracks have the same Mode A code (i.e., 1200), then each must have two or more REQUIRED Mode C replies, and two or more Mode C replies must be REQUIRED by one of the tracks and not the other. If the tracks have different Mode A codes, each track must have three or more REQUIRED replies (considering both modes) and five or more total REQUIRED + OK replies. In either case, one of the following conditions must be satisfied as well:

(a) both track Mode A codes appear in the reply group, or

(b) the group runlength must be ≥ VSP MAXTGTRUN, or

(c) there are two or more multiple-reply sweeps and the group is not a wide-pulse target.

If it is determined that the Two-Track Track Matching algorithm does not apply to the reply group, the algorithm is terminated. At this point, the next pair of tracks, if any, are tested.

3.6.4 Reply Allocation by Range

For each reply that is labeled OK for both tracks, an attempt is made to resolve the allocation ambiguity via a range test. To prepare for this test, the average range of each track's REQUIRED region is computed for each mode. If no Mode C reference was established for a track, or no Mode C REQUIRED replies were identified, its Mode C average range is set equal to

the Mode A average range. Range one-timers and multiple-reply-sweep one-timers are not used in this computation.

If a sweep has two OK-OK replies, the shorter range one is allocated to the shorter range track for that mode; the longer range one is then allocated to the other track. For OK-OK replies on single reply sweeps, the candidate reply's range is compared with the same-mode REQUIRED range centroids of the two tracks. If the reply range is more than 2 range counts closer to one of the centroids than to the other, the reply is allocated by range to the closer track. In either case, the newly allocated replies are called REQUIRED with respect to the appropriate track and FAILED with respect to the other track.

3.6.5 Elimination of Extraneous OK Replies

Various cases of garble can result in the labeling of OK replies far from the track's true REQUIRED region. To permit accurate azimuth centroiding, these replies must be rejected. Starting just before the first REQUIRED reply for a track, and working backward toward the beginning of the reply group, as soon as there are three consecutive FAILED replies, all earlier OK replies are converted to FAILED for the track. This process is repeated starting just after the last REQUIRED reply, and working forward toward the end of the reply group. In either case, if any affected reply were OK for the other track, it is converted to REQUIRED. Note that if Mode C replies were not tested because no altitude reference were established, these replies are ignored in the counting of consecutive FAILED replies.

3.6.6 Extending REQUIRED Regions

Due to reply garble, the REQUIRED regions of the two tracks may not cover the entire reply group. This could lead to an incorrect target azimuth computation. Thus, the next step is to extend the track whose REQUIRED region is "left-most" to the beginning of the reply group. Likewise, the track whose REQUIRED region is "right-most" is extended to the end of the reply group. In either case, if neither track's REQUIRED region is further left or right, then both tracks are extended in the appropriate direction. Range one-timers and multiple-reply-sweep one-timers are not allocated by this algorithm step.

If there are any un-allocated (i.e., not required) replies between the two REQUIRED regions, they also must be allocated to one of the tracks. If this "middle region" has an azimuth gap of more than one degree (11 ACP) between adjacent replies, then the replies on either side of the gap are allocated to the REQUIRED region on their side. If there is no gap, then the middle region is split up by azimuth. The azimuth center of the middle region is computed, and replies on either side of the azimuth center are allocated to the REQUIRED region on their side of the center. In this case, any Mode C replies found in the middle region are converted to garbled so as not to harm the later altitude determination process. Again, range one-timers and multiple-reply-sweep one-timers are not allocated.

3.6.7 Resolution of OK Replies

Any OK replies within a track's REQUIRED region are assumed to belong to that track, and thus are converted to REQUIRED. These replies could of course be allocated to both tracks if the regions overlap. Any (OK OK) Mode C replies that reside within only one track's REQUIRED region are set to REQUIRED for that track and FAILED for the other track.

Finally, it is possible for a track's OK region to extend well beyond its REQUIRED region, into the other track's REQUIRED region. This typically occurs when one track code is an undercode of the other track code, such as 1200 and 3624. Then any 3624 reply will be labeled OK for the 1200 track. There is no method of determining with certainty when the 1200 target ends in this situation. The best guess is made by cutting off the OK region at the point where the target width reaches the NOMINAL-WIDTH parameter; replies before this cutoff are converted to REQUIRED while replies beyond this cutoff are converted to FAILED.

3.6.8 Declaring Two Targets

Now that the reply allocation has been completed, the two target reports are declared. The target attributes are determined using the replies allocated as REQUIRED to each target. The target range and azimuth are computed using the Target Range and Target Azimuth algorithms, respectively. The Mode A code of each target is the code of the track it matched, with code confidence set to three. If there is a unique nearby track with matching code and known flight level altitude prediction, then the allocated replies are sent to the Target Altitude With Track algorithm; otherwise the Target Altitude Without Track algorithm is used. In either case, any Mode C reply that ended up as REQUIRED-REQUIRED is set to garbled for purposes of the altitude algorithms.

3.6.9 Two 1200 Code Target Sanity Check

If the two targets declared above both have a Mode A code of 1200, then a sanity test is performed to make sure there really are two 1200 targets in the reply group. Two 1200 code targets are maintained if either of the following conditions is satisfied:

(a) both target altitudes are confidence 3, or

(b) the two target ranges differ by at least four range counts.

If the sanity test indicates that the Two-Track Track Matching algorithm is not applicable, both targets are rejected. At this point the next pair of tracks, if any, is tested.

3.7 SINGLE-TRACK TRACK MATCHING

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with garble mask and one-timer flags, Group wide-pulse indicator List of nearby tracks
Outputs:	Target report if match successful

Algorithm:

The Single-Track Track Matching algorithm, under the assumption that all replies in the reply group are part of a single target, attempts to identify the track to which the replies correspond. It does this by considering each nearby track one at a time, comparing the reply codes to the track code and altitude. If one and only one success is registered, a report is made corresponding to that track.

If there is evidence that two reports are contained in the reply group, this algorithm is not executed. Thus if 2 or more sweeps exist that contain multiple replies in the group, or if the group is wider than a target can reasonably be, failure is reported. The specific allowable group azimuth width limit is governed by the VSP MAXTGTRUN as follows:

$$\Delta\theta = \begin{cases} \text{Max}(77, \text{VSP}) & \text{if } \rho \geq 5 \\ 1.5 * \text{VSP} & \text{if } \rho < 5 \end{cases}$$

The algorithm has two parts: testing of each reply against the track, and determining the match decision. In reply testing, each reply's code is compared against the corresponding track code. In forming the match decision, statistics gathered during reply testing are used to determine whether or not the reply group matches the track.

3.7.1 Reply Testing

Reply testing ignores all replies marked as one-timers. If the group has multiple replies on a sweep, and the group is not marked as a confirmed or potential wide-pulse group (see Wide Pulse Transponder Target algorithm), the match test automatically fails; otherwise, only the first reply on any sweep is processed.

Reply testing first makes a pass through the entire reply group, testing the code of each non-ignored reply against the track. For each reply tested, the garble mask constructed by the "Garble" algorithm is used in the code match to determine which code pulses are to be considered

For a Mode A reply, the testing proceeds as follows. If the reply code is specifically 1200, it can only match a track with the same code. Else if the reply code is in the set [1000, 0200, 0000], the track Mode A code must be in the set [1200, 1000, 0200, 0000] for the reply to match the track. Otherwise, the reply with its garble mask is tested against the track using the "Code Match With One Bit Drop" algorithm. If the reply fails to match the track, the "Mode A failure counter" is incremented by one. If the reply matches the track, the "Mode A clear pulse counter" is

increased by the number of clear pulses in the current reply, that is the number of pulses not covered by the garble mask.

For a Mode C reply, the processing depends on the type of track altitude. If the track has a known flight level (FL) altitude, then the Mode C reply code is compared with each of the five altitudes in the set [track alt - 2 FL, track alt - 1 FL, track alt, track alt + 1 FL, track alt + 2 FL] using the "Code Match With One Bit Drop" algorithm. For each FL tested, a separate "Mode C failure counter" and "Mode C clear pulse counter" are maintained.

If the track altitude is Brackets-Only, the Mode C reply is compared with the code 0 using the "Code Match With One Bit Drop" algorithm. A single "Mode C failure counter" and a single "Mode C clear pulse counter" are maintained.

If the track altitude is unknown, then each Mode C reply automatically matches the track. A single "Mode C clear pulse counter" is maintained.

3.7.2 The Match Decision

After all the replies have been tested, the match decision process occurs. First, the "best track altitude" match is determined. This altitude will then be used for the match decision. If the track altitude is FL, then the best match is the altitude in the "track altitude set" with the smallest "Mode C failure counter". If the track altitude is not FL, then the single Mode C failure counter is selected.

Next, a minimum threshold is determined for the Mode C clear pulse counter. For a track with a discrete Mode A code, the threshold is zero. For a non-discrete track, the threshold is given by the following formula:

min_pulses_c = MIN{ 12, 6 * (number of Mode C replies tested -
smallest Mode C failure counter) }

Next, a check is made as to whether the majority of Mode C replies tested matched the best track altitude. Note that if there were no Mode C replies tested, then a majority is affirmed.

Finally, the decision is reached as follows. If

1. the Mode A failure counter added to the smallest Mode C failure counter is at most two, and
2. the majority of altitudes tested matched the best track altitude, and
3. the Mode A clear pulse counter is at least 36, and
4. the Mode C clear pulse counter for the best track altitude is at least as large as the required minimum clear pulses (min_pulses_c),

then the reply group matches the track. Otherwise, the reply group does not match the track.

If failure occurs, but the track has a high validity alternate Mode A code, the process is repeated using that code to test against. The presence of an alternate code often signifies an aircraft changing its assigned code; in this case the second test is the only one that is likely to succeed.

3.7.3 Algorithm Resolution

After all nearby tracks have been tested, the number of successes is examined. If two or more tracks have registered a match, failure is returned, and Parse will have to resolve the situation. If no successes have been found, failure is also reported unless the Second Pass described below applies.

If a single track is identified, the replies are all used to generate a target report. The Mode A code of the report is just that of the track. The altitude is identified by the Target Altitude With Track routine if the track has a clear FL altitude; by the Target Altitude Without Track otherwise.

3.7.4 Second Reply Testing Pass

If only one nearby track existed, but the reply group does not match the track, a second pass through the reply testing process may be performed under certain conditions. The rationale for a second pass is as follows. Experience with real data has shown that there are occasions when a garbled reply can be called clear because the reply causing the garble was never declared. Such a reply would then fail to match the track because it is being tested using a clear garble mask. Usually, when several replies from the same aircraft are garbled, they are garbled by the same garbler. Thus, if the same garble distance appears multiple times for a reply group, a second pass though the reply testing process using that common distance for all replies may compensate for missing garble information.

There are three conditions that must all be satisfied in order to justify a second pass through the reply testing process. The first condition is that the track Mode A code must appear at least once in the reply group. This indicates that the group has a reasonable potential of matching the track. The second condition is that at least all but two of the Mode A and C replies are "Imperfect Supersets" of the track code and altitude. (See glossary for definition of an Imperfect Superset). This indicates that the group can realistically correspond to the track, since a reply that is garbled will have a superset of the track code. The final condition is that there must be a garbled pulse distance that appears more than once in the reply group. This indicates that there is a common garbler for the group which can be applied to replies labeled clear.

If a second pass is warranted, the closest multiply occurring shorter and longer range garble distances are determined. (Either of these distances may not exist). Then the second reply testing pass can be performed. This pass uses for all replies the garble mask created from the common garble distances, instead of using each reply's garble mask as determined by the "Garble" algorithm.

After the second pass is completed, the match decision process is again performed, exactly as after the first pass.

3.8 PARSE

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with garble mask and one-timer flags, Group wide-pulse indicator List of nearby tracks
Outputs:	One or more Beacon Target Reports

Algorithm:

The Parse algorithm is used either when there are no tracks near a reply group or when the reply group failed the Two-Track and Single-Track algorithms. The former usually indicates an isolated pop-up target, the latter either a pop-up target near another target or a group whose garble situation complexity exceeds the capability of the track match algorithms. The basic idea of the Parse algorithm is to determine the number of clear Mode A codes, and then to make a target for each clear code that is supported by sufficient replies.

3.8.1 Counting Clear Mode A Codes

The Parse algorithm starts by making a first pass through the reply group, examining only Mode A replies. Replies that are either range one-timers or multiple-reply-sweep one-timers are excluded from this algorithm. In addition, any reply called garbled by the 9-PAC Garble algorithm is skipped in this pass. Using the remaining replies, a "Clear Code List" is generated, each entry consisting of the Mode A code, the number of occurrences, the minimum and maximum ranges, the minimum and maximum azimuths, and the list of replies with the code. Up to 20 different clear Mode A codes can be accommodated.

Because the 1200 code is the most common Mode A code, it is possible for two 1200 targets to be contained within a single reply group. Thus the 1200 code is placed on the list twice if either of the following cases are encountered:

1. the azimuth extent of the 1200 replies exceeds a reasonable value (≥ VSP MAXRUN), and an azimuth gap of at least one degree exists in the middle of the reply sequence, or
2. multiple 1200 replies exist on two or more sweeps, and the group is not labeled as being wide-pulse

In the first case, the 1200 replies are allocated between the two Clear Code List entries according to which side of the gap they reside on; in the second case allocation is made by reply range.

If the first pass fails to find any clear codes, a second pass is made in which "clear" replies are determined using the ASR-9 BRP garble flags, instead of the 9-PAC Garble algorithm. This is done because the ASR-9 BRP uses a narrower window for identifying garbled replies, and may therefore call a reply clear when the 9-PAC Garble algorithm calls it garbled.

If at the end of the second pass there are still no clear codes, a single target report with unknown Mode A code (0 code, validity 0) is made for the reply group. The target altitude is determined using the "Target Altitude Without Track" algorithm.

3.8.2 Correcting Inter-Mode Mix-ups

On occasion, a Mode C reply will be received on a Mode A sweep (and vice versa). This occurs when uplink pulse interference confuses the transponder mode detection circuitry. To prevent a Mode C code from serving as the basis of target formation by being considered a second Mode A code, whenever two or more clear codes exist the Parse algorithm deletes from its Clear Code list any Mode A code that also appears as a Mode C code, provided that it satisfies either:

(a) the code appears more times as Mode C than as Mode A, or

(b) the code forms a majority of all Mode C replies.

The affected replies are then marked as totally garbled.

3.8.3 Removing Combined Codes

When two aircraft are in close azimuth proximity at virtually the same range, some of the replies corresponding to the two aircraft can overlap. Thus, the ASR-9 BRP declares only one reply instead of two, where the reply code is the OR-ing together of the two real aircraft reply codes.

To combat this effect, if there are three or more clear codes, and there is a code on the Clear Code List that is the OR-ing together of two other non-one-timer codes on the list, the combined code is removed from the list, and all its replies assigned to both of the ORed codes. Three restrictions exist on the use of this rule. First, this combined code cannot match the code of any nearby track. Second, at least one of the combining codes must differ from the combined code by more than two bits. These two rules help to ensure that there really are two aircraft, rather than the two combining codes being just a case of bit drops from the combined code. Finally, the combined code replies must be close enough in azimuth to each of the combining code replies so that the adjusted run lengths of the two aircraft remain reasonable ($\leq$ VSP MAXRUN).

3.8.4 Merging Bit-Drop Codes

On occasion, some of the replies from a single aircraft transponder do not have all their pulses declared by the ASR-9 BRP (bit-drops), resulting in two or more distinct Mode A codes being generated. By the nature of this process, the "real" Mode A code will be a superset of all the bit-drop codes. The Parse algorithm attempts to detect bit-drop codes using the procedure described below whenever two or more clear codes exist. Each bit-drop code is removed from the Clear Code List, and its replies merged with those of the real code.

Since undetected garble will also produce codes which are supersets of other codes, but in this case the real code has fewer bits and the superset code should be deleted, care is taken to differentiate between the two phenomena. Thus several safeguards are built into the bit-drop process.

First, one code is merged with another code only if the former has one bit less than the latter and both codes satisfy a range difference test. By successive merging, however, two-bit-

drop codes will eventually merge with the real code as long as the intermediate code also exists. Thus for example, the codes 2345, 2305, 2245, 2205, and 2244 will all be merged into 2345 if they all occur at approximately the same range.

Second, no code that is a "majority code" can be treated as a bit-drop code, where a "majority code" is defined as a code that constitutes either a simple majority of all Mode A replies in the group or more than 65% of all clear Mode A replies. This indicates that the potential bit-drop code is so prevalent in the reply group that it is likely to be a real code. The code into which the majority code would have merged is set to garbled and treated as in the next section unless the code appears three or more times and a track exists that matches the code.

Finally, the 1200 code is treated differently as befits its stature as the most common Mode A code. In particular, if three or more 1200 Mode A replies exist, 1200 cannot be called a bit-drop code. In addition, if the 1200 code exists, all codes in the set [0, 200, 1000] are automatically called bit-drop codes of 1200 even if any of them constitutes a majority code.

3.8.5 Single Clear Code Case

If after the various code consolidation cases are completed, there is but one clear code remaining, a single report using all the replies in the group is created. The range and azimuth of the report are determined according to the Target Range and Target Azimuth algorithms respectively.

The Mode A code of the report is the clear code, with the confidence automatically set to 3 if a nearby track matches the code. Otherwise, the confidence setting is determined by the Mode-A-Confidence-VSP, with any garbled reply whose code matches the clear code and who was declared ungarbled by the ASR-9 counted as clear.

If there is exactly one nearby track with the same clear Mode A code as the group, and that track has a known FL altitude with two or fewer coasts, then the "Target Altitude With Track" algorithm is used to determine the target altitude and confidence. In all other cases, including a brackets-only track, the "Target Altitude Without Track" algorithm is used.

If, instead, two or more clear Mode A codes exist on the list, the actions described in the following sections are performed.

3.8.6 Processing Garbled Superset Codes

Now that a final Clear Code List has been determined, a new pass is made through the Mode A replies in the group, excluding range one-timers and multiple-reply-sweep one-timers. This time, only replies labeled as garbled are processed. The definition of garble is the same used above in the development of the Clear Code List. Each garbled reply increments the count of each clear code for which it is a "True Superset" (see glossary), provided that the garbled reply is reasonably close in range to the range extent of the clear code. If the garbled reply is not a True Superset of any clear codes, then it increments the count of each clear code for which it is an "Imperfect Superset" (see glossary), again provided that is passes the range test with respect to the clear code.

It is possible in this assignment process for an unusual case of garble to cause a garbled reply far from the true target azimuth extent to be added to the code list. To prevent the deleterious

effects that this error would cause later in the algorithm, garbled replies that are more than 3 Mode A sweeps from the nearest reply on a code list are dropped from that list.

3.8.7 Mode A Code Selection

Next, the Parse algorithm decides how many clear codes have enough support to be used as the basis for a target. This decision is implemented via a multiple pass through the Clear Code List. If more than two codes are on the list, the first two passes may select any Mode A code; passes three and subsequent will be allowed to create additional targets only if discrete Mode A codes matching a track remain. This prevents spurious target formation in severe bit-drop or garble environments.

For each target pass, each code remaining on the Clear Code List is checked to determine the Mode A code on the list with the largest reply count. If two codes have the same total count, then the one with the largest number of clear occurrences is selected. The code selected on the first pass is automatically used to declare a target. For the second pass, the selected code must satisfy one of the following three conditions to be used:

(a) a clear count of at least three, or

(b) a clear count of two and either a total count of at least four or a unique discrete track match, or

(c) a clear count of one and a total count of at least three and a unique discrete track match.

For passes three and above, the selected code must satisfy the more stringent condition:

(a) a clear count of at least three and a total count of at least four and a unique discrete track match

If only one Mode A code is permitted to form a target, all Mode A replies on that code's list, plus all Mode C replies and remaining Mode A replies that fall within the azimuth extent of this list, are used to form the target. If this subset of replies is insufficient to pass the Minimum-Reply-VSP test, all group replies are allocated to the target. The target attributes are then determined as above for Single Clear Code.

If two or more (maximum of five) clear codes are acceptable for report formation, the multiple targets are formed as explained in the next sections.

3.8.8 First and Second Target Creation

If codes can be selected for two targets, the Mode A replies on each code's list are allocated to these targets (note that because of garble, some replies may be allocated to both targets). Mode C replies are then allocated to the targets according to the following two pass procedure:

Pass 1: Mode C azimuth allocation

(a) replies within the azimuth extent of only one Mode A list are allocated to that target

(b) the first reply on multiple-reply sweeps (not sweep-one-timer) are allocated to the shorter range target

(c) the last reply on multiple-reply sweeps are allocated to the longer range target

(d) other replies are left for pass 2

Pass 2: Mode C code allocation of remaining replies

(a) clear replies that agree with a clear reply already on one list (from Pass 1 processing) and not with any on the other list are allocated to that target

(b) garbled replies that are a superset of a clear reply already on one list and not a superset of any clear reply on the other list are allocated to that target

(c) replies failing a. and b. that are within the range limits of one list and not the other are allocated to that target

(d) remaining replies are set to garbled (if clear) and allocated to both targets

Each of the two lists of allocated replies are used to make a target, using the attribute determination rules given above in Single Clear Code.

3.8.9 2-Target Sanity Test

If the above algorithm produces two target reports, a Parse two-target sanity procedure is performed with these two targets before they are output. This procedure seeks some indication that two aircraft are really represented by the replies in the group, rather than some unusual garble or transponder condition splitting the replies from a single aircraft. If the two targets have the same Mode A code, both are accepted only if they have different altitudes. Otherwise, if the codes differ, the two targets are both accepted only if one of the following conditions applies:

(a) there is more than one multiple-reply sweep, and the wide-pulse indicator was not set, or

(b) the group run length exceeds the VSP MAXRUN and each target was allocated Mode C replies not given to the other target, or

(c) each target matches a nearby track's Mode A code, or

(d) neither Mode A code is a True Superset of the other, or

(e) if one target code is a True Superset of the other, then both targets have a known FL altitude and neither altitude is a True Superset of the other.

If none of the above conditions apply, it must be true that one Mode A code is a superset of the other. In this case, a "winner" and "loser" Mode A code are selected. If only one code matches a track, it is the winner; otherwise, the subset code is the winner (the superset is assumed to arise from undetected garble). Both targets are accepted if the loser code is 1200; otherwise, only one target, namely the one with the winner Mode A code, is accepted.

If only one target survives the sanity test, the losing clear code is merged into the winning clear code, and the entire Parse process is re-started at the point where the initial list of clear codes has been produced. This total re-start decision, instead of trying to salvage partial information (such as from garbling allocation), significantly simplifies the coding.

If both targets survive the sanity test, each is subjected to the Minimum-Reply-VSP test. The possible results, and the corresponding actions, are as follows:

(a) both targets pass the VSP — output both targets, proceed to possible third and subsequent target logic below

(b) only one target passes the VSP — output that target only, drop the other

(c) neither target passes the VSP — make a single target using all replies allocated to the two original targets, with the Mode A code of the first target at confidence 1.

3.8.10 Subsequent Targets

If two targets successfully navigated through the above logic, and additional clear codes exist that satisfy the Mode A Code Selection criteria, then a target is made for each such code. The Mode A replies on the code's list are allocated to the target. In addition, all Mode C replies that satisfy the azimuth and range extents of this list are allocated to the target. The rules of Single Clear Code are then applied to the allocated replies to determine the target attributes. These targets, as are all targets produces by any algorithm, are fed to the Minimum-Reply-VSP test before being output.

It is very possible that when three or more targets exist within a single group, the group was closed by the termination algorithm before all replies for the final target were received. Thus the azimuth of the incomplete target will be incorrect if it is output at this time. To counteract this effect, a check is made on the difference in azimuth between the current sweep and the final reply of each subsequent target. If that difference is less than the "normal" closing interval (see Process Open Groups), all replies allocated to the target are returned to the open group structures. In particular, they are processed by the Process Sweep algorithm, starting with the step after reply buffer entry.

3.9 TARGET RANGE

Algorithm Subsystem: Target Formation

Inputs: Group of replies.

Outputs: Target Range Centroid (64ths of nmi).

Algorithm:

The Target Range algorithm determines the range of a target consisting of the given group of replies. The range centroid is computed as the simple averaging of the ranges of all replies in the group other than range one-timers and multiple-reply-sweep one-timers.

3.10 TARGET AZIMUTH

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with one-timer flags
Outputs:	Target Azimuth (ACP).

Algorithm:

The Target Azimuth algorithm determines the azimuth of a target made up of the given group of replies. If there are six or more replies, the azimuth is computed by averaging the azimuths of the first three and the last three replies. If there are fewer than six replies, all reply azimuths are averaged to determine the target azimuth.

If two or more replies exist on the same sweep, only one is used. Replies that are flagged as range one-timers are not used in computing target azimuth. (Replies flagged as azimuth outliers were previously rejected.)

Special care is taken to handle reply groups that straddle the north mark.

3.10.1 Azimuth Adjustment

Because of range reduction, it is possible for a target near a sector boundary to lose some of its replies when the reply stream straddles the sector boundary. To prevent biased azimuths from being declared, an azimuth adjustment is made whenever both of the following conditions apply:

1. The azimuth given by az_1 + REPW falls within a sector S whose truncated range is less than $range_{report}$, where az_1 is the azimuth of the first reply of the group.
2. The report includes a reply whose azimuth is within 1° of the sector S boundary.

In this case, the azimuth of the report will be output as

$$az_{report} = az_1 + REPW/2$$

A similar rule is used whenever the initial replies of a report are lost. In this case the test is on the azimuth az_n - REPW, and the adjustment becomes

$$az_{report} = az_n - REPW/2$$

No report can be sufficiently wide that both tests pass.

3.11 CODE MATCH

Algorithm Subsystem:	Target Formation
Inputs:	Reply code (of any mode), along with garble masks, Track code (of same mode)
Outputs:	Result of match test

Algorithm:

The Code Match algorithm determines if a reply code matches a track code, given garble masks for the reply that indicate which of its pulse positions (if any) are overlapped by shorter and longer range replies. The reply and track must be of the same mode (Mode A or C).

The reply and track codes are said to match if both of the following conditions are satisfied:

(a) the reply code is identical to the track code in the clear pulse region, and

(b) the reply code has no bit-drops with respect to the track code in the garbled pulse regions.

Since the effect of garble is usually to produce additional pulses in the reply (although destructive interference is occasionally encountered), the reply code is permitted to have any number of bit-adds in the garbled pulse regions.

3.12 CODE MATCH WITH ONE BIT-DROP

Algorithm Subsystem:	Target Formation
Inputs:	Reply code (of any mode), along with garble masks, Track code (of same mode)
Outputs:	Result of match test

Algorithm:

The Code Match With One Bit-Drop algorithm determines if a reply code matches a track code, given garble masks for the reply that indicate which of its pulse positions (if any) are overlapped by shorter and longer range replies. The reply and track codes must be of the same mode (Mode A or C). Unlike the standard Code Match algorithm, this procedure allows one bit-drop in the reply clear pulse region and any number of bit-drops in the garbled pulse regions (due to destructive interference).

The reply and track codes match if the following condition is satisfied:

(a) the reply code is identical to the track code in the clear pulse region except for at most a single bit drop

In the garbled pulse regions, the reply code may have any number of bit-adds or bit-drops relative to the track code.

3.13 TARGET ALTITUDE WITH TRACK

Algorithm Subsystem:	Target Formation
Inputs:	List of Mode C replies allocated to target, each with garble flags Track Flight Level Altitude prediction.
Outputs:	Target altitude and validity (confidence).

Algorithm:

The Target Altitude With Track algorithm is used to determine the target report altitude when the nearby matching Mode A track has a known flight level altitude prediction. A list of Mode C replies allocated to the target is used to determine the altitude to be placed in the report. In cases of garble or anomalous effects, the track prediction is used in an attempt to resolve the uncertainties. If no ambiguity exists, altitude confidence will be 3 if the Mode C replies agree with the given track altitude (defined as within two flight levels). Otherwise, confidence will be determined depending upon the algorithm path utilized.

3.13.1 No Mode C Replies

If there are no Mode C replies allocated to the target, the report altitude is set to 0, with confidence 0.

3.13.2 Majority Altitude

If there is a "majority altitude" clear reply code (more than half of all replies both have that code and are marked as clear), then the report altitude is set to:

(a) 7766 if the majority altitude code is 0 (brackets-only), or

(b) 6030 if the majority altitude is not decodable, or

(c) the decoded majority altitude flight level otherwise.

If case (c) applies, and the altitude agrees with the track prediction, the altitude confidence is set to 3. Otherwise, the VSP V Altitude Validation algorithm is used to determine the confidence.

3.13.3 Clear Altitude List

If there is no majority clear altitude value, a pass is made through the Mode C replies to compile a list of clear decodable (or brackets-only) altitude codes, along with the number of clear occurrences for each code value, and the total numbers of clear and brackets-only replies. On this first pass, the garble flag for each reply comes from the 9-PAC Reply Garble algorithm, except that undecodable clear replies are considered as garbled.

If at the end of the first pass, there are no clear altitude codes, then a second pass is made through the altitude replies. This time, any reply with a decodable flight level that agrees with the track altitude prediction, and that is considered ungarbled by the ASR-9 BRP garble flags, is called clear.

3.13.4 Altitude Transitions

It is possible for an aircraft to make an altitude transition during the beam dwell of the antenna. Thus if there are exactly two clear decodable altitudes on the list, and they differ by one flight level, the more frequently occurring altitude (using total counts) is merged with the other one. In case of a tie, the later occurring altitude is selected.

3.13.5 Merging Bit-Drops

If there are two or more clear altitude values, it is possible that the situation arose because several of the replies had bit drops from the true code. Thus a check is made to see if, relative to the clear code with the most bits set (the parent code), all other clear altitude values are single bit-drops. If this condition is met, and if the parent code occurs more frequently than any of the other codes, and if the parent code is decodable, then all the other clear codes are merged with the parent code.

3.13.6 Track Altitude Agreement

If there is exactly one clear altitude value on the list, and it agrees with the track altitude prediction (within 2 flight level of the prediction), the report is assigned the decoded reply flight level with confidence 3.

3.13.7 Resolution of Clear/Garbled Altitudes

If any clear altitude value is also found as being called garbled in other Mode C replies, an attempt is made to decide whether the clear or the garbled designation is correct. If the value is equal to one if the five track agreement levels, and it is not a superset of another agreement level, all garbled replies with that value are set to clear and the clear count increased accordingly. If the agreement clause exists, but a superset condition also exists, the replies are left as labeled. Finally, if the value disagrees with all track agreement levels, all clear replies for that value are converted to garbled, and the clear level is removed from the list.

3.13.8 No Track Altitude Agreement

If no Mode C clear altitude value on the list agrees with the track altitude prediction, it is possible that agreeing altitude replies exist, but that they were all garbled. To allow for this situation, five new "fake" altitude code values are created and added to the clear list. These values correspond to the five agreeing track altitudes, namely the ones within 2 levels of the prediction. The clear count for each fake altitude is initialized to the number of garbled replies that match the code; all such replies are then considered as ungarbled below.

If any of the original clear non-agreeing altitude values represent a single bit drop from any of the fake altitudes, that clear altitude is removed from the list, and its total count added to the counts of the matching fake altitudes.

3.13.9 Processing Garbled Replies

Using the same definition of a "garbled" reply as was used in making the clear altitude list, each remaining garbled reply is processed. If the garbled reply, with its garble mask, matches a clear altitude code using the Code Match With One Bit-Drop algorithm, then the total count of the

clear altitude is incremented. As a result of this process, each clear altitude ends up with a clear count and a total count.

At the end of the garble processing loop, any fake altitude whose count has a zero setting, meaning it was unsupported by any garble replies, is removed from the clear list.

3.13.10 No Agreeing Flight Level Altitude

If there are no clear altitudes left on the list that agree with the track prediction, processing is terminated, and the Target Altitude Without Track algorithm is called instead (with the original unmodified group of replies).

3.13.11 Scoring Altitudes

Each clear altitude is assigned a score based upon its clear count, its garbled count, and the number of Mode C replies not applicable to it, in the following manner:

$$score = 3 * CLR + 2 * (CNT - CLR) - (NUMC - CNT)$$

where CLR = clear count for a given altitude

CNT = total count for a given altitude

NUMC = number of Mode C replies allocated to the target

3.13.12 One Clear Altitude

If there is only a single clear altitude value (including brackets-only) left on the list, that value is placed in the report. The corresponding altitude confidence setting depends upon the score S attained by the altitude:

confidence = 3	$S \geq 3$
confidence = 2	$S = 2$
confidence = 1	$S \leq 1$

3.13.13 Multiple Clear Altitudes

If there are two or more clear altitudes (including brackets-only) still on the list, the one with the highest score is selected for the report. If two altitudes have the same score, the tie is broken in the following precedence order:

(a) if one agrees with the track and the others do not, select the one that agrees with the track, otherwise

(b) if one has more bits set than any other, select it, otherwise

(c) select the one that is closest to the track predicted altitude

If the winning altitude disagrees with the track prediction, processing is terminated, and the Target Altitude Without Track algorithm is called instead (with the original unmodified group of replies).

Otherwise, the report altitude confidence setting depends upon the difference D in score between the winning and runner-up altitudes, and whether or not the runner-up altitude disagrees or agrees with the track:

	disagree	agree
confidence = 3	$D \geq 3$	$D \geq 6$
confidence = 2	$D = 2$	$D \geq 3$
confidence = 1	$D \leq 3$	$D \leq 2$

3.14 TARGET ALTITUDE WITHOUT TRACK

Algorithm Subsystem:	Target Formation
Inputs:	List of Mode C replies allocated to target, each with garble flags Matching Brackets-Only Track indicator.
Outputs:	Target altitude and validity (confidence).

Algorithm:

The Target Altitude Without Track algorithm is used to determine the target report altitude if there is no known matching track, or if the track matching the group on its Mode A code has a brackets-only altitude. The list of Mode C replies allocated to the target is used to determine the altitude to be placed in the report.

3.14.1 No Mode C Replies

If there are no Mode C replies allocated to the target, the report altitude is set to 0, with confidence 0.

3.14.2 Majority Altitude

If there is a "majority altitude" clear reply code (more than half of all replies both have that code and are marked as clear), then the report altitude is set to:

(a) 7766 if the majority altitude code is 0 (brackets-only), or

(b) 6030 if the majority altitude is not decodable, or

(c) the decoded majority altitude flight level otherwise.

If case (a) applies, and there is a matching brackets-only track, the altitude confidence is set to 3. Otherwise, the VSP V Altitude Validation algorithm is used to determine the confidence.

3.14.3 Clear Altitude List

If there is no majority clear altitude value, a pass is made through the Mode C replies to compile a list of clear decodable (or brackets-only) altitude codes, along with the number of clear and garbled occurrences for each code value, and the total numbers of clear and brackets-only replies. The number of clear occurrences of each value is called its clear count, while the number of clear plus garbled occurrences is called its total count. On this first pass, the garble flag for each reply comes from the 9-PAC Reply Garble algorithm.

If at the end of the first pass, there are no clear altitude codes or brackets-only codes, then a second pass is made, this time using the ASR-9 BRP garble flags to determine which codes are clear. On either pass, undecodable clear replies are considered as garbled and not counted.

If there are still no clear altitudes or brackets-only altitudes after pass two, or if all clear altitudes are undecodable, the report altitude is set to 6030 (altitude unknown) with confidence 0.

If there is only a single clear decodable flight-level altitude, that level is output in the report. The altitude confidence is then determined from the level's total count using the VSP V Altitude Validation algorithm.

3.14.4 Matching Brackets-Only Altitudes

If there is a matching brackets-only track, and there is at least one brackets-only reply, then the report altitude is set to brackets-only (7766) provided one of the following cases applies, where the applicable confidence is as indicated with the case:

(a) at least 3 brackets-only replies exist; confidence = 3

(b) all clear replies are brackets-only; confidence = 3

(c) exactly 3 clear replies exist, of which 2 are brackets-only; confidence = 3

(d) >3 clear replies exist, of which 2 are brackets-only; confidence = 2;

(e) exactly 2 clear replies exist, of which 1 is brackets-only; confidence = 1

In all other cases of brackets-only replies existing, the resolution requires the algorithm steps described below.

3.14.5 Altitude Transitions

It is possible for an aircraft to make an altitude transition during the beam dwell of the antenna. Thus if there are exactly two clear decodable altitudes on the list, and they differ by one flight level, the more frequently occurring altitude (using total counts) is output with the report. In case of a tie, the later occurring altitude is selected. The altitude confidence is determined using the VSP V Altitude Validation algorithm on the combined total counts.

3.14.6 Merging Bit-Drops

If there are two or more clear altitude values, it is possible that the situation arose because several of the replies had bit drops from the true code. Thus a check is made to see if, relative to the clear code with the most bits set (the parent code), all other clear altitude values are single bit-drops. If this condition is met, and if the parent code occurs more frequently than any of the other codes, and if the parent code is decodable, then the report altitude is set to the decoded parent code flight level with confidence 3.

3.14.7 Merging Undetected Garble

If there are two or more clear altitude values, it is possible that the situation arose because undetected garble produced additional clear codes. Thus a check is made to see if, relative to the clear code with the fewest bits set (the undercode), all other clear altitude values are supersets. If this condition is met, and if the undercode is decodable, and if each superset code has at least one garbled occurrence (total count exceed clear count), then the report altitude is set to the decoded undercode flight level with confidence 3.

3.14.8 Resolution of Clear/Garbled Altitudes

If any clear altitude value is also found as being called garbled in other Mode C replies, (that is its count exceeds its clear count), an attempt is made to decide whether the clear or the garbled designation is correct. If there exists another clear code that is a subset of the code in question, and the found code is always clear (count = clear count), then all clear replies of the code in question are converted to garbled and the clear level is removed from the list. Otherwise, all garbled replies with the value in question are set to clear and the clear count increased accordingly.

3.14.9 Processing Garbled Replies

If there are two or more clear altitudes (including brackets-only), the remaining garbled replies must be allocated, where the definition of a "garbled" reply is that used in making the clear altitude list. If a remaining garbled reply is a true superset of one or more clear altitude codes, the total count of each such clear altitude is incremented. As a result of this process, each clear altitude ends up with a clear count and a total count.

3.14.10 Altitude Selection

The altitude selected for the report, from the list of clear altitudes, is the one that has the largest total count. If two altitudes have the same total count, the one with the larger clear count is chosen. If a tie still exists, the altitude with the most bits set is used. The final tie breaker is the altitude appearing last.

If the clear count for the chosen altitude is a majority of all Mode C replies, and the clear count for any other clear altitude is one, the altitude confidence is set according to the VSP-V rule on the clear count of the majority altitude. Otherwise, the altitude confidence placed in the report is determined by the difference D in total count between the winning flight level (or brackets-only) and the runner-up level:

$$D \leq 1 \qquad \text{confidence} = 1$$

$$D \geq 2 \qquad \text{confidence} = 2$$

3.15 ONE-TIMER

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with garble flag
Outputs:	One-timer flags set for appropriate replies

Algorithm:

A "one-timer" is a reply in the group that is an anomalous reply. Either it is a real reply that does not belong in the group, or a reply corrupted by a fruit reply, or a reply produced as the result of a transponder or reply processor (ASR-9 BRP) input data misinterpretation.

The One-Timer algorithm flags those replies that satisfy one or more of the following one-timer rules.

3.15.1 Multiple-Reply-Sweep one-timer

If only one sweep contributes two or more group replies, that sweep and its replies are called a multiple-reply-sweep one-timer.

3.15.2 Range one-timer

A range one-timer is a reply whose range is not consistent with its neighboring same-mode replies. If any multiple-reply sweeps exist for the group, there can be no range one-timers. Also if either mode has fewer than 3 replies, no range one-timer can exist for that mode.

The centroid range is computed for the replies of each mode. If a reply is >3 range clocks from the centroid for its mode, and its 3 same-mode neighbors on each side are all ≤3 range clocks from the centroid, the reply is called a range one-timer. If either side has less than 3 neighbors, all existing neighbors are used for that side's test.

3.15.3 Azimuth one-timer

If the reply on either end of the group differs in azimuth from its neighbor by more than 11 ACPs (1 degree), the reply is called an azimuth one-timer. This test is then repeated for the new end reply.

3.15.4 Garble one-timer

If a reply is marked as garbled by the ASR-9 BRP, and none of its 3 neighbors on either side is so marked, the reply is called a garble one-timer.

3.15.5 Clear Mode A (C) one-timer

If any clear Mode A (C) code appears 3 or more times in the group, any other clear code that appears only once is called a clear Mode A (C) one-timer.

3.15.6 Garbled Mode A (C) one-timer

If any garbled Mode A (C) code appears 3 or more times in the group, any other garbled code that appears only once is called a garbled Mode A (C) one-timer.

3.16 REPLY GARBLE

Algorithm Subsystem:	Target Formation
Inputs:	Group of replies, each with ASR-9 BRP garble flag, Group Wide-Pulse indicator
Outputs:	For each reply in group, a modified reply garble flag, a mask due to shorter range garblers, and a mask due to longer range garblers.

Algorithm:

The Reply Garble algorithm determines for each reply in the group which pulse positions are assumed to be clear and which are potentially garbled. Reply garble occurs when two or more replies to the same interrogation (i.e., on the same sweep) exist at a range separation such that some of their pulse positions overlap in time. A reply corrupted by garble will usually have one or more of its pulses changed from 0 to 1 (although destructive interference is possible). For example, the code 1234 may appear as 3274, where two pulse positions have been corrupted.

Garble information is kept in two garble masks for each reply. Each garble mask indicates which reply pulses are clear and garbled; the first due to replies at shorter ranges, the second due to replies at longer ranges. A mask consists of a series of bit-fields, one for each reply pulse, with a '1' indicating garble in the corresponding bit position.

The Reply Garble algorithm is performed in four separate passes. In the first pass, called the 1200 Force Clear Rules, 1200-code Mode A replies and Brackets-Only (code 0000) Mode C are forced to be clear. Code 1200 is the most common Mode A code, and since this code value has only 2 bits set, it is extremely unlikely to be caused by another code being garbled. A Brackets-Only Mode C reply value cannot be garbled, since it has no bits set. In addition, if 3 or more 1200-code Mode A replies are found in the group, all replies with codes 0000, 1000, and 0200 are assumed to be bit-drop versions of 1200.

In the second pass, called the General Force Clear Rules, Mode A replies that are the same as the Mode A code of a nearby track are forced to be clear. Mode C replies are forced to be clear only if they agree with the altitude prediction band of one of the nearby tracks whose Mode A code was found in the group. Special care is taken with Mode C replies because two adjacent flight levels differ by exactly one bit.

In the third pass, called the Reply Garble Rules, each reply is compared with the other replies on the same sweep. Garbled reply pulses are determined based on range separation between the replies, and the garble masks are set accordingly.

In the fourth and final pass, called the Neighbor Rules, each reply's garble masks are compared with those of its neighboring replies. Under certain conditions, when a neighbor has more encompassing garble masks, the reply's garble masks are set to the same values as the neighbor. This rule is intended to correct cases where pass three failed to detect the proper garble because the garbling reply was not detected by the ASR-9 BRP.

The complete set of rules for Reply Garble are as follows:

Pass 1: The 1200 Force Clear Rules

A. All Mode A replies i with code 1200 are forced to be clear:

GP_i = null (Garble Plus mask)

GM_i = null (Garble Minus mask)

B. If 3 or more 1200 replies are in the group, all Mode A replies with codes of 0000, 0200, or 1000 are assumed to be bit-drops from 1200, and are thus set to 1200 and forced to be clear.

C. All Mode C replies with code 0000 are forced to be clear.

Pass 2: The General Force Rules

A. Identify the set S_1 of all tracks whose predicted position falls within the search box for the reply group.

B. Identify the subset S_2 of tracks for which 1 or more Mode A replies, clear or garbled, match the track's Mode A code.

C. For each Mode A reply i that matches the Mode A code of a track in S_2, set the reply to be forced clear:

GP_i = null

GM_i = null

D. For each Mode C reply i, attempt to find a track j in the set S_2 such that the following conditions are met:

a. reply i represents a decodable FL f_i

b. $| FL_j - f_i | \leq 2$

c. no other FL f_k that satisfies this equation exists in the reply group

d. the Mode C code of reply i is not a superset of the code of any FL f_m which satisfies:

$$| FL_j - f_m | \leq | FL_j - f_i |$$

that is, f_i cannot be the result of a closer flight level to the track being garbled

If successful, set the reply to be forced clear:

GP_i = null

GM_i = null

Pass 3: The Reply Garble Rules

A. For each reply i, check later replies j on the sweep until max range difference with i is exceeded, or until a garbling reply is located that satisfies:

$$n*17 - 6 \leq \Delta\rho \leq n*17 + 4$$

If such reply is found, set i's GP to n.

B. For each reply i, check earlier replies j on the sweep until max range difference with i is exceeded, or until a garbling reply is located that satisfies:

$$n*17 - 4 \leq |\Delta\rho| \leq n*17 + 6$$

If such reply is found, set i's GM to n.

Pass 4: The Neighbor Rules

A. For each Mode A reply i, if his neighboring Mode A reply j has the same code as him, set:

$$newGP_i = Min\{ GP_i, GP_j \}$$

$$newGM_i = Max\{ GM_i, GM_j \}$$

B. For each reply i, if his neighboring reply j is of the opposite mode from him, set:

$$newGP_i = Min\{ GP_i, GP_j \}$$

$$newGM_i = Max\{ GM_i, GM_j \}$$

C. For each reply i that was called garbled by the ASR-9's reply processor, but for which no garbling GP or GM has been found, set:

$$newGP_i = 1$$

D. For each reply i, store new values

$$GP_i = newGP_i$$

$$GM_i = newGM_i$$

3.17 WIDE-PULSE TARGET

Algorithm Subsystem: Target Formation

Inputs: Group of replies,
each with garble mask

Outputs: Wide-pulse transponder decision:

Yes = Wide-pulse target,
longer range replies on multiple-reply sweeps removed from group.

No = Not a wide-pulse target,
all replies maintained.

Maybe = Decision deferred

Algorithm:

The Wide-Pulse Transponder Target algorithm attempts to determine whether a group of replies previously flagged as a potential wide-pulse transponder target should be deemed an actual wide-pulse target. To be safe, the group is required to satisfy four necessary conditions tests: the Sweep Test, the Run Length Test, the Leading Edge Range Test, and the Trailing Edge Range Test. Each test is described below.

1. Sweep Test

   If multiple-reply sweeps are really caused by a wide-pulse transponder, the clear pulses of each longer range reply should line up with a pulse of the shorter range reply. That is, the longer reply ungarbled-region code should be a subset of the shorter range reply code. If two or fewer multiple-reply sweeps fail this condition, the Sweep Test is passed.

2. Run Length Test

   If the run length of the reply group is no greater than the VSP MAXTGTRUN, the Run Length Test is satisfied. Otherwise the replies are probably due to two aircraft.

3. Leading Edge Range Test

   The rationale behind the Leading and Trailing Edge Range Tests is that for a wide-pulse transponder target, the longer range replies of the multiple-reply sweeps are extraneous, rather than belonging to a partially overlapping longer-range aircraft. Thus, for the edge sweeps (if any) that have only a single reply, the average reply range should agree with the shorter range replies of the multiple-reply sweeps. If they instead agree with the longer range replies, the longer range replies and the edge replies probably constitute a second , longer-range and overlapping, target.

   To perform the edge tests, the following average range quantities are computed:

   r0 = replies before first multiple-reply sweep

   r1 = first replies on multiple-reply sweeps

r2 = second replies on multiple-reply sweeps

r3 = replies after last multiple-reply sweep

If there are no single-reply sweeps before the first multiple-reply sweep, the Leading Edge Range Test is satisfied automatically. Otherwise, the average range of these replies must be closer to the shorter range replies of the multiple-reply sweeps than to the longer range replies:

$$|r1 - r0| \leq |r2 - r0|$$

4. Trailing Edge Range Test

   If there are no single-reply sweeps after the last multiple-reply sweep, the Trailing Edge Range Test is satisfied automatically. Otherwise, the average range of these replies must be closer to the shorter range replies of the multiple-reply sweeps than to the longer range replies:

$$|r1 - r3| \leq |r2 - r3|$$

3.17.1 The Decision

If all four conditions are satisfied, the reply group is called a wide-pulse transponder target. The longer range replies on the multiple-reply sweeps are removed from the group, and the reduced reply group is returned to the start of the Target Formation process. In particular, the group may now be found to be Perfect or Perfectible.

If the reply group fails the Sweep Test, it is rejected as a wide-pulse transponder target. If enough longer range replies exist, they will normally be used to generate a second target.

If the group passes the Sweep Test, but fails either the Run Length Test or one of the Edge Range Tests, there are conflicting test results, and the group can neither be confirmed nor rejected as a wide-pulse transponder target. The group is thus labeled as a "maybe"; the final resolution will come in the Single-Track Track Match algorithm or the Parse algorithm.

3.18 PROCESS TARGET REPORT

Algorithm Subsystem: Target Formation

Inputs: Target Report
with Military Statistics.

Outputs: Possibly Modified Target Report.

Algorithm:

The Process Target Report algorithm (see flowchart in Figure 9) first determines whether the report is part of a military situation. If it is not deleted as a redundant military report, it is associated to the proper track and passed through the false target reflection tests. Finally, it is checked for RTQC and output to the Merge Task.

3.18.1 Military Identification and Emergency Targets

The Military Identification and Emergency Target algorithm is used to reject the extra targets (at longer range than the real target) produced by a military aircraft transponder sending an identification or emergency. These extra targets are not used to update the BTD Track File, nor are they output to the Merge Task. See the Military Ident and Emergency Target algorithm write-up for the details of this process.

3.18.2 Associating Target Reports

Each target report declared above attempts to associate to an existing track in the BTD Track File, or else possibly initiate a new track. See the Track Association/Initiation algorithm write-up for the details of this process.

3.18.3 Checking For Reflection False Targets

Each target report declared above is checked using the Reflection Processing algorithms. Discrete code reports use the Discrete Reflection algorithm (see write-up), and if unsuccessful, the Non-discrete Reflection algorithm (see write-up). Non-discrete code reports use only the Non-discrete Reflection algorithm.

3.18.4 Flagging False Targets for Merge

The radar/beacon Merge process is not permitted to delete beacon false reports that are reinforced by radar reports unless the BTD can identify a "permanent" reflector to support the beacon false target decision. To aid Merge, each false target is labeled as "supported" or "unsupported" depending upon whether or not a permanent reflector source can be found.

For a Non-Discrete target to be called false, the reflector generating the false target decision must be identified; BTD sends the target to Merge as supported if it was a permanent reflector, unsupported otherwise.

For Discrete false targets, where code and not reflectors serve as the discriminant, BTD must perform the Non-Discrete Reflection algorithm to determine whether a permanent reflector

can be found. If successful, the report is output to Merge as supported; if no reflector can be found, or only a mature reflector is found, the report is output to Merge as unsupported.

RTQC Targets. Any target that is at the position specified by the two VSPs RTQCRNG and RTQCAZM is flagged as a beacon RTQC target.

3.18.5 Outputting Targets to Merge

Finally, each target is placed on an output list, which is sent to the Merge Task every 16 ACP.

3.19 MILITARY IDENT & EMERGENCY TARGET

Algorithm Subsystem:	Target Formation
Inputs:	Beacon Target Report Military Statistics for reply group
Outputs:	Target Report deleted if it is the 2nd group of a Military Ident message, or 2nd, 3rd, or 4th group of a Mil Emergency message. SPI flag set to 1 if the first group of a Military Ident. Mode A set to 7700 if the first group of a Mil Emergency.

Algorithm:

The Military Ident and Emergency Target algorithm determines if the given target is either to beginning or end of either a Military Ident or Military Emergency message signaled by an aircraft. This algorithm uses the Mil Ident & Emergency Match Statistics gathered in the Process Sweep routine. These statistics are counters of the number of replies in the group that satisfied one of the rules for being a part of a Military Ident or Emergency message. A Military Ident has 2 reply groups, and a Military Emergency has 3 or 4 reply groups, with special conditions as to the range separation and codes of the two reply groups. See the Process Sweep routine for the military reply matching rules.

This algorithm tries to determine if the group's Mode A code and military match statistics clearly indicate that the target report is a part of a military ident or military emergency message. A report whose group has zero counts for all the military statistics is ignored by this algorithm. The decision for reports with non-zero counts is reached as described in the following rules.

If two (or more) reports were made from the group's replies, only one is considered in this algorithm, namely the one whose allocated replies azimuth extent encompasses the military starting and ending azimuths reported with the group. If no report, or more than one report, passes this test, no military action is taken for any of the reports.

If the report under consideration has a Mode A code of 0000, and a non-zero military other reply count, it is deleted without further testing. No action is taken for any other report with a code of 0000.

If the report under consideration has a discrete Mode A code that matches the military code field of the reply group, it is:

1. considered a military emergency report if the emergency start count is ≥ 3, else
2. considered a military ident report if the ident start count is ≥ 3, else
3. deleted as a military duplicate report if the military other reply count is ≥ 3, else
4. left as a normal report if no count is sufficient to indicate definitive military processing action.

If the report under consideration has a Mode A code that either fails to match the military code field of the reply group or is non-discrete, more care is required to declare a military situation. In particular, a military emergency or military ident start requires that the military failure

count be ≤ 3; a duplicate report deletion requires in addition that the report have no high confidence altitude. Success in any step of the above counts test, followed by failure in the additional test, ends the processing, and the report is left unmodified.

If the report is called the start of a Military Emergency, its Mode A code is set to 7700 Octal. If the report is deemed the start of a Military Ident, its SPI flag is set to 1.

4.0 REFLECTION PROCESSING

4.1 BUILDING REFLECTOR DATABASE

Algorithm Subsystem:	Reflection Processing
Inputs:	Discrete code target report Track updated by report.
Outputs:	Updated discrete sample database Updated Reflector Database.

Algorithm:

The Building Reflector Database algorithm determines when a valid Reflector Sample can be constructed, and updates the Reflector Database based on the new sample. This process is called whenever a discrete report updates a track with a matching code.

A Reflector Sample requires three reports with the same discrete code, a false one between two real ones, where the real reports correlated to the track that caused the reflection. A sample database is maintained for every discrete code.

4.1.1 The Reflector Database

The Reflector Database maintains a list of known reflectors. The following attributes mathematically define each reflector: average range, average orientation angle (180°-360°), and the minimum, maximum, and average azimuths.

Each reflector has a status based on the number of different aircraft that have contributed samples and the total number of samples for the reflector. A reflector that has been seen by at least one aircraft is called ACTIVE. A reflector that has been seen by two or more aircraft, with three or more total samples, is called MATURE. A mature reflector that is seen for more than one day is called PERMANENT. To aid in creating and deleting reflectors, the times of the oldest and newest siting are also recorded.

4.1.2 False Report Processing

If the input report was declared false by the Discrete Reflection algorithm, and it has a validated flight level altitude, and the report's reference track is the same as the one in the sample, the report is stored in the false report slot of the Reflector Sample.

4.1.3 Real Report Processing

If the input report is labeled real and has a validated flight level altitude, but there is no track in the sample database for its discrete code, both the real report and the correlated track are stored in the sample database. This handles the track start-up case.

If the report is real and has a validated flight level altitude, but the track correlated to the report is not the one stored in the Reflector Sample, then the entire sample database entry for this report's code is cleared. This indicates that a tracking switch or error has occurred.

If the report is real and has a validated flight level altitude, and the track correlated to the report is the one stored in the Reflector Sample, the report is stored in the new-real-report slot of the Reflector Sample. A reflector sample can then be made provided all of the following conditions are satisfied:

(a) there is a previous real report in the sample database within 2.5 scans of the new real report, and

(b) there is a false report between the times of the two real reports, and

(c) the real report altitude interpolated to the time of the false report is within 2 flight levels of the false report altitude.

4.1.4 Updating Reflector Database with a Sample

Making a Reflector Sample consists of computing the range, azimuth, and orientation of the reflector that would create the false report position when reflecting an interrogation to the real track. Next, the Reflector Database is checked to see if the computed reflector is already known. In order to join (and perhaps modify) an existing reflector, the range, azimuth, and orientation of the computed Reflector Sample must be within VSPs REFL_RNG_TOL, REFL_AZM_TOL, and REFL_OR_TOL of the existing reflector, respectively. For wide reflectors, the azimuth extent is widened to be within 1° of the edge of the reflector, except that no reflector can exceed 30°.

If the computed Reflector Sample merges with an existing reflector, that reflector's dimensions are updated as necessary to include the new sample. If a choice of existing reflectors exists for the sample, the order of selection is determined as follows:

1. the best fitting Permanent or Mature reflector.
2. the best fitting Active reflector.

The goodness of fit is scored as

$$S = \frac{\Delta\theta}{\text{REFL_AZM_TOL}} + \frac{\Delta\rho}{\text{REFL_RNG_TOL}} + \frac{\Delta\phi}{\text{REFL_OR_TOL}}$$

where lowest score wins.

If the computed Reflector Sample does not merge with any existing reflectors, a new reflector is started with an ACTIVE status. The mathematical attributes for the new reflector are initialized.

4.1.5 Adjusting the Coverage Window

Each reflector has a coverage window, which is the azimuth region in which the reflector may be expected to cause false targets. This region is defined as the minimum and maximum azimuths of the reflector, adjusted by adding a reflector extension parameter:

CW_{min} = Reflector min azimuth - VSP REFL_EXT

CW_{max} = Reflector max azimuth + VSP REFL_EXT

The reflector coverage window is adjusted every time the reflector's geometrical attributes are updated by a new sample.

The coverage window is used by the Non-discrete Reflection algorithm to determine which reflectors should be considered for a candidate false report. To aid in this determination, a list is maintained for every ACP of all the reflectors whose coverage windows include it.

4.1.6 Widening Reflector Search Window

Each reflector has a range-azimuth search window that is used by the Non-discrete Reflection algorithm to determine whether or not a candidate report is a reflection false target. The reflector attributes are used to project where the real aircraft would have to be if the candidate report is false due to that reflector. The search window expresses how far from this predicted position the track is allowed to be.

The reflector search window starts at a nominal size specified by the VSPs REFL_SRCH_RNG and REFL_SRCH_AZM. When an existing reflector is updated by a new discrete Reflection Sample, the reflector's search window is checked for applicability. In particular, using the technique described in the Non-discrete Reflection algorithm, the expected position of the real track is computed using the reflector attributes and the false report from the discrete Reflector Sample. If the real time-aligned report from the Reflector Sample falls outside the current reflector search window, the search window may be too small.

A score is maintained for the search window. Every time the real time-aligned discrete report sample falls within the current window, the score is incremented by one. Every time the real report sample falls outside the window, the score is decreased by three. Whenever the score falls below zero, the search window is widened by three percent, and the score is reset to zero. The search window is not allowed to grow beyond 300 percent of its nominal size.

4.1.7 Removing Outdated Reflectors

Any PERMANENT reflector that has not been seen in three weeks, or any MATURE or ACTIVE reflector that has not been seen in three days, is removed from the Reflector Database.

4.2 DISCRETE REFLECTION

Algorithm Subsystem:	Reflection Processing
Inputs:	Target report Track associated to report.
Outputs:	Discrete Reflection status of target Track type.

Algorithm:

The Discrete Reflection algorithm uses the BTD Track File to determine if the given discrete target report is a reflection of an existing real discrete track.

A report that is associated to a mature real track can not be labeled as a reflection, and hence is not input to this procedure.

In order for a report to be declared a discrete reflection, there must exist a real track (immature or mature) with no more than one coast, with the same discrete code, and with a known altitude, such that the report:

(a) is at a longer range than the time-aligned track prediction, and

(b) does not disagree with the track altitude.

Each track with the same discrete code as the report is considered, until either the report can be declared a reflection, or until there are no more tracks to consider.

If no track was found that produced a reflection result, but a real track was found in the search that had the same discrete code and a shorter time-aligned range than the report, the report is labelled unsure. This means that the track was either coasting or had an unknown altitude.

It is possible for a reflection to be seen prior to the real aircraft reports. Thus if an immature real track is found during the search whose time-aligned range is longer than the report and whose altitude is within 1 flight level of the report, that track's status is changed to UNSURE. Its counts of real and false report associations are reset to zero, and it cannot be used to declare a reflection.

Also, it is possible that tracking errors (or track coasts) or undetected aircraft maneuvers have prevented a real report from associating to its track. Thus if a real track is found whose time-aligned range is longer than the report, whose range and azimuth are within 2 NMI and 200 ACPs of the report, and whose altitude is within 1 flight level of the report, that track's status is changed to UNSURE. Its counts of real and false report associations are reset to zero, and it cannot be used to declare a reflection.

The report and track altitudes are said to not disagree if any of the following conditions are satisfied:

(a) the report altitude and track altitude prediction are within VSP DISC_ALT flight levels of each other, or

(b) either report or track altitude is brackets-only, and the other has at most 2 bits set, or

(c) either the report or track altitude is a True Superset of the other, or

(d) the report and track altitudes have at most 2 bits different in their Grey-code versions, or

(e) either the report or track altitude is unknown, or

(f) the report altitude is low confidence, or

(g) the track altitude has coasted for 2 or more scans. or

(h) the report altitude is a flight level, and its Grey-coded Mode C is within 1 bit of its Mode A code.

4.3 NON-DISCRETE REFLECTION

Algorithm Subsystem:	Reflection Processing
Inputs:	Target report Track associated to report Reflector Database
Outputs:	Report reflection status (false, real, unsure) NOTE: Unsure goes out to Merge as real.

Algorithm:

The Non-discrete Reflection algorithm uses a dynamically updated Reflector Database to determine if a candidate report is a reflection false target.

Any report that is associated to a mature real track can not be considered a reflection, and must be labeled real.

If the report is not within the coverage window of any known mature or permanent reflector, the report is called real. See Building Reflector Database for a detailed description of the reflector "coverage window".

Each mature or permanent reflector whose coverage window contains the candidate report azimuth (ACP) is considered, until the report is declared a reflection or there are no more reflectors to consider. For each reflector, the expected true target position is computed. This is where the real aircraft must be in order for the candidate false report to have been generated by the reflector. In computing the true target position, if the candidate report altitude is unknown, the Default Altitude (see Glossary) is used. If a true target position can not be computed, the current reflector is skipped.

Next, the BTD track file is searched so that a list of all tracks within 2 NMI and 20 degrees of the expected true target position can be compiled. Each of these tracks is considered as possibly being the true target aircraft, until a suitable track is found or there are no more tracks to consider. In order to declare the candidate report a reflection false target, all of the following conditions must be satisfied:

(a) the track can not be the one associated to the candidate false report

(b) the track can not have more than 1 coasting scan

(c) the track can not have an alternate Mode A code (meaning that the last update disagreed with the track code)

(d) the candidate false report must be at a longer range than the time-aligned track prediction

(e) the range and azimuth differences between the computed true target position and the time-aligned real track prediction must be within the reflector's search window parameters

(f) the Mode A code and altitude of the track and candidate report must be compatible.

The details of the code and altitude agreement tests appear below. If (a)-(e) are satisfied, the final determination relative to the current reflector is made as follows:

(1) If both code and altitude tests return AGREE, the report is declared a reflection (false).

(2) If either test returns DISAGREE, the report is declared real.

(3) If both tests return UNSURE, the report is called unsure.

(4) If one test returns AGREE, but the other returns UNSURE, then the report is called a reflection if the report is associated to a false track; otherwise, the report is called unsure.

If the result is not to call the report false, the next track is considered. However, the result is remembered so that if any unsure result were ever obtained, the report can be labeled as unsure.

4.3.1 Mode A Code Agreement

The Mode A codes of the report and track AGREE if they are the same. The test returns UNSURE if one is a True Superset of the other, or the report and track codes differ by at most 2 bits. Otherwise, the codes are said to DISAGREE.

4.3.2 Altitude Agreement

The altitude agreement test is more complicated. The agreement rules depend on the type of track to which the candidate false report is associated.

If the candidate report is associated to a false track, the Discrete Reflection altitude test is used to determine the altitude match (AGREE or DISAGREE).

If the candidate report is associated to an immature real track, a more restrictive Non-discrete Reflection altitude agreement test is used. The report and track altitudes AGREE if both are bracket-only, or if both are flight level and the track's time-aligned altitude is within VSP NOND_ALT of the report altitude. An UNSURE match occurs if this test fails but the Discrete Reflection altitude test is passed. In this event, the track is marked as "unsure due to altitude test". A DISAGREE match occurs if the Discrete Reflection altitude test also fails.

If the report is associated to an unsure track, and the report used to update the track on the previous scan was called unsure because it failed the altitude test, then it is possible that a mistake was made on the previous scan due to an incorrect track altitude extrapolation. In such a case, the previous scan's altitude test is repeated, using the track's latest altitude update to provide a more accurate interpolation of the previous scan time-aligned altitude.

If the repeated altitude test yields agreement this time, the mistake is corrected by incrementing the track's false report count. If this increase causes the track to convert from unsure to false, the Discrete Reflection altitude test can be used to determine the current scan's altitude match. Otherwise, if the test result were not changed, or if the track remains unsure, the previous paragraph's Non-discrete Reflection altitude agreement test is used.

4.3.3 Reflection Decision

Finally, the report is labeled as either real, false, or unsure. It is labeled false if any reflector and track were found that satisfied the above criteria. It is labeled unsure if any reflector and track were found that satisfied the unsure test, but no false result were ever obtained. Otherwise the report is labeled real. An unsure report will be marked as real in the report output to the Merge processing task; the unsure result is used only by track update.

5.0 INTERNAL TRACKING

5.1 TRACKING OVERVIEW

Algorithm Subsystem: Internal Tracking

Algorithm:

The BTD Tracking algorithm maintains a private beacon track file used internally by the 9-PAC BTD task. The track file is used for two purposes: identifying false target reports due to beacon reflections (see Discrete Reflection and Non-discrete Reflection), and degarbling beacon reply codes (See Reply Garble, Single-Track Track Match, Two-Track Track Match, and Target Altitude with Track).

The BTD Tracking algorithm consists of Track Association/Initiation and Track Update, each described in detail elsewhere. In Track Association, just after a target report is declared, an attempt is made to choose the existing track that best matches the report code, altitude, and position. If no track is found, the report is used to initiate a new track file entry, if acceptable. If a report chooses a track that already has a report association, a decision is made immediately as to which report is the best match for the track, and then the losing report looks for another track.

Track Update is performed half a scan after the track predicted azimuth, when the track has presumably had enough time to associate to the correct report. Tracks are placed on the update list in a batch mode in 30 degree azimuth wedges, with one update performed per sweep. Whenever the latest beacon reply sweep indicates the start of a new azimuth wedge, all tracks whose predicted positions fall within the azimuth wedge 180 degrees behind the current sweep are listed. If a given track has a report association, the report is used to update the track. The initial update of a track is handled with special rules. If a track does not receive a report association, the track is coasted. If a track coasts a parametric number of consecutive scans, it is dropped from the track file.

After a track is updated, if the report has a Discrete code, the report and track are passed to the Building Reflector Database algorithm. The Track Update algorithm maintains a Discrete Code Track List, containing up to 10 entries for each Discrete code. This list is used in the Discrete Reflection algorithm.

5.1.1 BTD Track File

Each BTD Track File entry contains all attributes necessary to describe a single beacon track, as follows:

Attribute	Description
range	Range prediction (nmi)
azimuth	Azimuth prediction (rad)
run_length	Azimuth span (ACP) of last rept update
time	Update time (ACP) of prediction
trk_alt	Altitude (FL. or Brackets-Only) prediction
alt_type	{FL, Unknown, No Mode C, Brackets-Only}
alt_profile	UP, DOWN, LEVEL
alt_coast	Number of scans since altitude update
last_alt	Last altitude used to update trk_alt
altern_alt	Possible alternate altitude for track
altern_alt_conf	Confidence of alternate altitude {0,1,2,3}
altern_alt_coast	Number of scans since last altern_alt update
alt_rate	Rate of change of altitude (FL)
x	Position converted to Cartesian plane
y	Position converted to Cartesian plane
x_dot	Position change rate, in x-dimension
y_dot	Position change rate, in y-dimension
trk_range_box	Track's range association box (NMI)
trk_az_box	Track's azimuth association box (rad)
code	Mode A code
code_conf	Confidence of Mode A code {0,1,2,3}
code_coast	Number of consecutive code coasts
altern_code	Possible alternate (transition) code
coasts	Numer of scans track has coasted
assoc_flag	Can track receive associations on this scan?
nassoc	Does track already have an association?
assoc_rpt	Attributes of report associated to track
nreports	Number of reports associated to track
nreal	Number of real report updates
nfalse	Number of false report updates
type	{UNSURE, FALSE, IMMATURE_REAL, REAL}
real_trk_ref	Pointer to REAL track used as reference (for FALSE track only)
unsure_by_alt	Did track fail Non-discrete Reflection due only to altitude test?
trk_link	Pointer to next track in same Geographic Box

5.1.2 Geographic Track Links

Individual tracks are linked together by geographic location, so that the task of searching for tracks near a target report does not require checking the entire track file. Geographical boxes are maintained for every 5 Nmi of range and 30 degrees of azimuth. However, a single "close-in" box exists within 5 Nmi of the radar. For each box, a List Header is maintained, which points to the first track in the box. Each track is then linked to another track in the same box, with the last track's link set to NULL.

Since track predictions are not exact quantities, the track search for a given report position can not always be limited to a single geographic box. This is particularly true when the report position is near one of the edges of a box. Thus, a Search List is maintained consisting of up to four geographic boxes that must be searched for a report position. The Search List has an entry for every 1 Nmi of range and 3 degrees of azimuth.

For a report with a range less than 2 Nmi, only the close-in box is listed. Otherwise, a search box is drawn centered at the 1 Nmi by 3 degrees position (r,a), with a range span of 3 Nmi and an azimuth span of 9 degrees. Therefore, the four corners of the search box are determined as follows:

$r1 = r + 2$

$a1 = a + 6$

$r2 = r - 1$

$a2 = a + 6$

$r3 = r - 1$

$a3 = a - 3$

$r4 = r + 2$

$a4 = a - 3$

Next, the geographic box (5 NMI by 30 degree) is determined for each search box corner. Finally, all unique geographic boxes covered by the four corners of the search box are placed in the Search List for (r,a). Thus, at most 4 geographic boxes can be in the list for (r,a).

5.2 TRACK ASSOCIATION/INITIATION

Algorithm Subsystem:	Internal Tracking
Inputs:	Beacon target report List of nearby tracks
Outputs:	Correlating or Initiating Track Number for Report

Algorithm:

The BTD Track Association algorithm chooses, from a list of nearby tracks, the best track for a given beacon report. If no existing track is available or suitable for the report, the report attempts to initiate a new track.

Each track has a slot where the associated report can be stored until the time later in the scan when the track is updated. If the report chooses a track with no existing report association, then the report is stored in the track's associated-report slot.

If, however, the report chooses a track that already has an associated report, the new and old reports are compared with the track and the "better" match is determined. If the old report is a better match for the track, the new report is put through the association process again, with that track disqualified as a choice for the report. If instead the new report is a better match for the track, then the new report replaces the old report in the track's associated-report slot, and the old report is the one again put through the association process.

5.2.1 Report-To-Track Association

A report that has a 0000 Mode A code is sufficiently suspect that it is not permitted to either associate to any existing track nor initiate a new track.

If the report has a Discrete Mode A code, and one or more matching code tracks exist, the Discrete Association Rules are applied to see if a suitable track can be found.

If there are no tracks with the report's Discrete Mode A code, or if the report has a Non-discrete Mode A code, the Non-discrete Association Rules are applied to see if a suitable track can be found.

If no suitable track was found in either case, the report is sent to Track Initiation to see if a new track can be initiated.

With a suitable track for association, the next step is to check if that track has already been chosen by another report on the current scan. If the track does not have an "old report" association, then the current ("new") report is placed in the track's associated-report slot.

If the track has an old report association, the track decides which report is a better association choice. If the track has a Discrete Mode A code, and both the new and old reports have the same Discrete Mode A code as the track, the Discrete Association Choice Rules are used to select the better report. Otherwise, the Non-discrete Association Choice Rules are used. In either case, if the track chooses the old report, the new report is passed back through the beginning of the association process so that a different track can be found, if possible. The current track is disqualified from selection. If the track chooses the new report, then the new report is placed in the

track's associated-report slot, and the old report is passed through the association process again so that it can choose a different track, if possible.

5.2.2 Discrete Association Rules

The first step in Discrete Association is to determine the number of nearby qualified tracks with the same Discrete Mode A code as the report. A list of "disqualified" tracks is maintained, consisting of those tracks previously selected by this report, but for which the track had to choose between this report and another, and the other report won. In order to be considered "nearby", a track's prediction position must be within 2 NMI and 200 ACP of the report position.

If there are no qualified nearby tracks, the report receives no track association. If there is exactly one qualified track, that track is selected for the report.

If there are multiple qualified tracks, the report altitude is compared to each of the track altitude predictions using a "Strong Altitude Agreement" test. Strong altitude agreement means that the report and track altitudes are either both Brackets-Only or else they are within 1 Flight Level. If exactly one track has strong altitude agreement with the report, and that track is the closest in range to the report, then that track is selected. If multiple tracks have strong altitude agreement with the report, then the track with a predicted range closest to the report range is selected.

If none of the tracks had strong altitude agreement with the report, or the altitude/range ambiguity exists, then the Discrete Reflection Altitude test is tried for each track with respect to the report. (See the Discrete Reflection section for more details.) If exactly one track passes the Discrete Reflection Altitude test, that track is selected. If multiple tracks pass the test, the track whose predicted range is closest to the report range is selected. Otherwise, no association is made.

A special case exception to the above rules is used to resolve the case of two reports created for the same aircraft due to a missed wide pulse transponder or to in-line multipath. If:

(a) the report finds two matching discrete tracks, and

(b) one of the tracks has a discrete association, and

(c) the associating report and this report satisfy a superset relationship with respect to their altitudes

then the shorter range report is associated to the shorter range track, and the longer range report to the longer range track.

5.2.3 Non-discrete Association Rules

For Non-discrete Association, each qualified nearby track, regardless of Mode A code, is "scored" with respect to the report. As with Discrete Association, tracks previously disqualified for the report are not considered. The track's score is computed as follows:

score = (2 * Mode A score) + Mode C score

where the Mode A score is 2 if the report and track have the same code, 1 if report and track codes differ by at most one bit, and 0 otherwise. The Mode C score is 1 if both the report and track altitudes are Brackets-Only, or neither the report nor track have any Mode C, or if the report and track altitudes are within 5 Flight Levels; otherwise, the Mode C score is 0.

If no tracks received a non-zero score, then no association occurs. Otherwise, the track with the best score is selected. To break a tie, the track whose predicted range is closest to the report range is selected.

5.2.4 Discrete Association Choice Rules

When there are two Discrete reports with the same Mode A code as a track, the better report for the track must be chosen. First, the "Strong Altitude" test (discussed earlier) is applied to both reports with respect to the track. If one report passes, and the other fails, then the report with strong altitude agreement is selected. If both reports have strong altitude agreement with the track, then the report with the shorter range is selected.

If neither report has strong altitude agreement with the track, then the Discrete Reflection Altitude test is used. If one report passes, it is selected for association. Otherwise, the one with the shorter range is selected.

5.2.5 Non-discrete Association Choice Rules

When there are two reports that have selected the same track, the better report for the track must be chosen. The report with the better "association score" (see Non-discrete Association Rules above) is selected for association with the track. To break a tie, the report whose range is closest to the track's predicted range is selected.

5.2.6 Track Initiation

The track initiation process consists of allocating an available track file entry, storing some initial track attributes, and placing the track in the appropriate geographical box. A list of tracks is maintained for each box, based on each 5 NMI of range and 30 degrees of azimuth.

When a new track is initiated, several track attributes are recorded. The report is placed in the track's associated-report slot, and the track is marked as "closed" for further associations on the current scan. The number of reports for the track is set to 0, and will be incremented by the Track Update algorithm a half scan later.

If any of the following conditions exists, the report is considered too unreliable to be allowed to initiate a new track:

(a) the report has a 0000 Mode A code; or

(b) the report Mode A code is not validated (confidence < 3); or

(c) the report was one of multiple reports created for a single reply group by the Parse algorithm.

In addition, if there are no available track file entries, no track is initiated.

5.3 TRACK UPDATE

Algorithm Subsystem:	Internal Tracking
Inputs:	Track file entry Associated report, if any
Outputs:	Updated or coasted track file entry

Algorithm:

The BTD Track Update algorithm updates the given track file entry using the track's associated report. If the track has just been initiated on the current scan, the Initial Update Rules are used; otherwise, the Mature Update Rules are used. If the track does not have an associated report, the track's position and altitude are predicted ahead using the Coasting Rules.

Tracks are updated in 30 degree azimuth wedges. Whenever the latest beacon reply sweep enters a new azimuth wedge, all tracks whose predicted positions fall within the azimuth wedge 180 degrees behind the current sweep are updated. A track is not updated until it is a half scan old to guarantee that the track has sufficient time to locate its best report association.

In order to prevent an entire sector of track update actions from occurring on 1 sweep, thereby significantly delaying the next several sweeps, only 1 track per sweep may be updated. As tracks become ready for updating, they are placed on an update list. This list is processed 1 track per sweep until emptied.

The Track Update algorithm maintains a Discrete Code Track List, containing up to 10 entries for each Discrete code. Whenever a discrete track is updated by a matching discrete report, the report and track are passed to the Building A Reflector Database algorithm.

The algorithm also maintains a track list for each 5 NMI by 30 degree geographic box.. Whenever the track's updated position prediction places it within a different box, the track is moved from its previous box list to its new box list.

5.3.1 Coasting Rules

The coast count of a track without an associated report is incremented. Should this count reach the drop value, VSP DROPCNT-INITF for a track that consists of only one report (called an Initial Track), or VSP DROPCNT for other tracks (called Mature), the track is dropped from the track file. Whenever a discrete track is dropped, the Discrete Code Track List is updated.

If the track has not been dropped, the following attributes are modified or updated as indicated:

(a) update time is increased by 4096 ACPs;

(b) association is enabled for the next scan;

(c) if a Flight Level altitude prediction exists, it is predicted to the new update time based on the previous prediction and altitude rate attributes;

(d) Mode A code coast count is incremented;

(e) altitude and alternate alt coast counts are incremented;

(f) range and azimuth are predicted to the new update time based on previous prediction and motion attributes (update is actually done is [x,y] space and then converted back to [rho,theta] space); if the new predicted range exceeds 70 nmi, the track is dropped;

(g) track's association box (track_range_box, track_az_box) is set as follows:

track_range_box = 5*SIG_RNG + 0.0421*sq(SCAN_RT/4)*8

track_az_box = 5*MAX(SIG_AZM, SIG_RNG/rpt range) + 0.0421*sq(SCAN_RT/4)*8

where SIG_RNG = 100 feet,
SIG_AZM = 3 milliradians,
SCAN_RT = 4.8 seconds.

5.3.2 Initial Update Rules

For the initial update, the track nreports is set to 1. The track update time is set to the report time + 4096 ACP. The track range and azimuth are set to the report attributes. The track motion attributes (x_dot, y_dot) are set to 0.

The initial altitude update is handled as follows. The track altitude is set to UNKNOWN. The altitude coast count is set to 1. The altitude rate is set to UNKNOWN. The altitude profile is set to 0, meaning level. If the report altitude is a Brackets-Only or decodable Flight Level, with a confidence of 2 or 3, the track altern_alt is set to the report altitude, with the altern_alt_conf set to the report altitude confidence. Otherwise, the track altern_alt is set to UNKNOWN, with altern_alt_conf set to 0. The altern_alt_coast count is set to 1.

The track Mode A code and code_conf are set to that of the report. The code_coast count is set to 0. The altern_code is set to 0, as is the track coast count. The associated-report slot is cleared, and the track is marked as able to accept associations on the next scan.

The track's association box (track_range_box, track_az_box) are set for maximum reasonable aircraft motion, since a position prediction can not be made for next scan.

track_range_box = MAX_VEL * SCAN_RT

track_az_box = MAX(track_range_box/report range, 3 deg.)

where MAX_VEL = 600 knots,

SCAN_RT = 4.8 seconds.

A FALSE report starts the track type as UNSURE, with nfalse = 1, and nreal = 0. A REAL report starts the track type as IMMATURE_REAL, with nfalse = 0, and nreal = 1. An UNSURE report starts the track type as UNSURE with nfalse = 0, and nreal = 0.

If the report Mode A code is Discrete, the Discrete Code Track List is updated to include the new track, with a maximum of 10 tracks remembered for each Discrete code.

5.3.3 Mature Update Rules

The first step is to convert the track current predicted position and the report position into [x,y] space. To do this, an altitude must be selected for the track and for the report. The same

altitude will be used for both the track and the report. The conversion altitude is selected in the following order of preference:

(a) track altitude prediction, if known Flight Level; or

(b) track altern_alt, if known Flight Level with confidence 3; or

(c) report altitude, if known Flight Level with confidence 3; or

(d) the Default Altitude.

The Default Altitude is defined as MIN(slant range/2, 0.5 NMI). Note that the conversion altitude is not allowed to be greater than 70 percent of the slant range (of the report or track).

Next, the track's motion rate attributes (x_dot, y_dot) are corrected based on the report that updated the track. The motion attributes can be corrected. If the track only has one previous report update, then use the following formulae:

x_dot = (rpt_x - trk_x) / (coasts + 1)

y_dot = (rpt_y - trk_y) / (coasts + 1)

If the track coasted last scan, use the following formulae:

x_dot = (rpt_x - last_x) / (coasts + 1)

y_dot = (rpt_y - last_y) / (coasts + 1)

where last_x and last_y are the previous scan report positions, determined as follows:

last_x = trk_x - (x_dot * (coasts + 1))

last_y = trk_y - (y_dot * (coasts + 1))

Otherwise, use the following formulae to determine the new motion attributes:

x_dot = x_dot + (rpt_x - trk_x)

y_dot = y_dot + (rpt_y - trk_y)

Now, predict the track [x,y] position ahead for the next scan, as follows:

trk_x = rpt_x + x_dot

trk_y = rpt_y + y_dot

Next, the track altitude attributes are updated, as discussed below in Altitude Update.

Next, the track position can be converted back into [rho,theta] space, using the track's updated altitude prediction if it is a known Flight Level, or else using the same conversion altitude used earlier in going from [rho,theta] to [x,y] space. As discussed earlier, the conversion altitude is not allowed to be greater than 70 percent of the track's range.

Next, the track association box is updated, if necessary. For tracks within 2 NMI of the sensor that do not have a known Flight Level altitude prediction, set the track_range_box and track_az_box attributes to the parameters used for coasting tracks (see Coasting Rules above). Otherwise, the box attributes are set to their normal values, as shown below:

track_range_box = (5*SIG_RNG) + (0.0421*sq(SCAN_RT/4) * 3)

track_az_box = (5 * MAX(SIG_AZM, SIG_RNG/rpt range)) +
(0.0421 * sq(SCAN_RT/4) * 3)

where SIG_RNG = 100 feet (in NMI),

SIG_AZM = 3 milli-radians,
SCAN_RT = 4.8 seconds.

The Mode A code update works as follows. If the report and track have the same code, the track code is unchanged, and the altern_code and code_coast fields are set to 0. Otherwise, if the report code code is the same as the track's altern_code, the track code is changed to the altern_code, the altern_code and the code_coast are set to 0, and the Discrete Code Track List is modified as necessary. Otherwise, the track code is unchanged, the altern_code is set to the report code, and the code_coast count is incremented.

Once the track has been updated, the nreport field is incremented, the associated-report slot is cleared, and the track is marked as able to accept new report associations next scan. The update time is set to the report time + 4096 ACP, and the coasts field is set to 0.

If the report is REAL, the nreal field is incremented. If the report is FALSE, the nfalse field is incremented. Now, the track type updated, as follows. Once a track is REAL, it remains REAL forever. If an UNSURE track now has at least VSP MATURE-REAL REAL reports, the track type becomes REAL. If an UNSURE track now has at least VSP MATURE-FALSE FALSE reports, the track type becomes FALSE. If an IMMATURE_REAL track has been updated with a FALSE or UNSURE report, the track type is changed to UNSURE. IMMATURE_REAL tracks are those that have received only REAL reports, but less than VSP MATURE-REAL REAL reports. If an IMMATURE_REAL track now has at least VSP MATURE-REAL real reports, then the track type is changed to REAL. Finally, if a FALSE track now has at least VSP MATURE-REAL REAL reports, then the track type is changed to REAL.

If the updated track predicted range exceeds 70 NMI, the track is dropped from the track file. Whenever a track is dropped, the Discrete Code Track List is updated, if necessary.

5.3.4 Altitude Update

The track altitude update process works as follows. If the report alt_conf is ≤ 1, the track altitude is coasted if it is a known Flight Level, with its alt_coast incremented. Also, if there is an altern_alt for the track, then the altern_alt_coast is incremented. The following shows the altitude coasting procedure:

trk_alt = trk_alt + alt_rate + (sign(trk_alt) * 0.5)
alt_coast = alt_coast + 1
if altern_alt exists
altern_alt_coast = altern_alt_coast + 1

Otherwise, the report altitude must be either a decodable Flight Level or Brackets-Only, and has an alt_conf of 2 or 3. This report altitude can be used to update the track altitude. If the report altitude agrees with the track altitude (within 2 Flight Levels of each other, or are both Brackets-Only), and the report altitude has confidence 3, the report altitude updates the track, and the track altern_alt fields are cleared, as follows:

altern_alt = NONE
altern_alt_coast = 1

case 1: alt_profile = LEVEL

(a) rpt_alt > (trk_alt+1) (WAY UP)

alt_profile=UP
alt_rate=COMPUTED,
trk_alt=COMPUTED

(b) rpt_alt = (trk_alt+1) (UP)

alt_profile = UP
alt_rate = 0
trk_alt = rpt_alt

(c) rpt_alt < (trk_alt-1) (WAY DOWN)

alt_profile = DOWN
alt_rate=COMPUTED,
trk_alt=COMPUTED

(d) rpt_alt = (trk_alt-1) (DOWN)

alt_profile = DOWN
alt_rate = 0
trk_alt = rpt_alt

case 2: alt_profile = UP AND alt_rate = 0

(a) rpt_alt > trk_alt (UP)
alt_profile = UP
alt_rate = COMPUTED
trk_alt = COMPUTED

(b) rpt_alt ≤ trk_alt (LEVEL or DOWN)

alt_profile = LEVEL
alt_rate = 0
trk_alt = rpt_alt

case 3: alt_profile = UP AND alt_rate > 0

(a) rpt ≥ trk_alt (LEVEL or UP)

alt_profile = UP
alt_rate = COMPUTED
trk_alt = COMPUTED

(b) rpt < trk_alt (DOWN)

alt_rate = COMPUTED
if(computed alt_rate = 0)
alt_profile = LEVEL
else if(computed alt_rate > 0)

alt_profile = UP
else
alt_profile = DOWN

if(computed alt_profile = DOWN)
alt_rate = 0
trk_alt = rpt_alt
else
trk_alt = COMPUTED

case 4: alt_profile = DOWN AND alt_rate = 0

(a) rpt_alt < trk_alt (DOWN)

alt_profile = DOWN
alt_rate = COMPUTED
trk_alt = COMPUTED

(b) rpt_alt ≥ trk_alt (LEVEL or UP)

alt_profile = LEVEL
alt_rate = 0
trk_alt = rpt_alt

case 5: alt_profile = DOWN AND alt_rate < 0

(a) rpt_alt ≤ trk_alt (LEVEL or DOWN)

alt_profile = DOWN
alt_rate = COMPUTED
trk_alt = COMPUTED

(b) rpt_alt > trk_alt (UP)

alt_rate = COMPUTED
if(computed alt_rate = 0)
alt_profile = LEVEL
else if(computed alt_rate > 0)
alt_profile = UP
else
alt_profile = DOWN
if(computed alt_profile = UP)
alt_rate = 0
trk_alt = rpt_alt
else
trk_alt = COMPUTED

In the above description, alt_rate is computed as follows:

last_alt = trk_alt - (alt_rate * alt_coast) + (sign(trk_alt) * 0.45)
alt_rate = (rpt_alt - last_alt) / alt_coast

In the above description, trk_alt is computed as follows:

trk_alt = rpt_alt + computed alt_rate + (sign(rpt_alt) * 0.5)

If report altitude agrees with the track altitude, and the report altitude has confidence 2, the track altitude is coasted if it is a known Flight Level, and the altern_alt_coast is incremented if there is an altern_alt for the track, as shown earlier:

trk_alt = trk_alt + alt_rate + (sign(trk_alt) * 0.5)
alt_coast = alt_coast + 1
if altern_alt exists
altern_alt_coast = altern_alt_coast + 1

If the report altitude does not agree with the track altitude, then the track's altern_alt is tested, if there is any. If the report and track altern_alt both have confidence 3, then they agree if they are within 5 Flight Level (or both Brackets-Only). If either the report or track altern_alt have confidence 2, then they agree if they are within 2 Flight Level (or both Brackets-Only).

If the report agrees with the altern_alt, and either the report altitude has confidence 3, or the report has confidence 2 and the altern_alt has confidence3, then the report altitude replaces the track altitude, as follows:

alt_rate = (rpt_alt - altern_alt) / altern_alt_coast
alt_profile = sign(alt_rate)
trk_alt = rpt_alt + alt_rate + sign(rpt_alt) + 0.5
alt_coast = 1
altern_alt = NONE
altern_alt_coast = 1

If the report agrees with the altern_alt, the report altitude has confidence 2, and the altern_alt has confidence 2, then the track altitude is coasted, and the report altitude replaces the altern_alt, as follows:

trk_alt = trk_alt + alt_rate + (sign(trk_alt) * 0.5)
alt_coast = alt_coast + 1
altern_alt = rpt_alt
altern_alt_coast = 1
altern_alt_conf = rpt_alt_conf

If the report altitude disagrees with the altern_alt, and either the report altitude has confidence 3 or the altern_alt has confidence <3, then the track altitude is coasted, and the report altitude replaces the altern_alt, as above.

Otherwise, the track altitude is coasted, and the altern_alt remains the same.

6.0 SYSTEM PROCEDURES

6.1 INITIALIZATION/RESET

Initialization/Reset is the process by which the 9-PAC BTD task is prepared for execution by initializing various data structures. This process occurs at system startup, as well as whenever the beacon reply data stream may have been corrupted or interrupted. The signal for possible trouble is whenever the sequence of beacon reply sweeps has a large azimuth jump, indicating either a temporary beacon antenna failure or a backwards movement of the antenna. Also, the ASP may reset the beacon ring buffer, in which case the BTD task will be reset.

The following tasks are performed:

a) Beacon Reply Buffer is flushed;

b) Beacon Track File is flushed;

c) Beacon Reflector Sample Database is flushed;

d) All ACTIVE and MATURE Reflectors are removed from the Beacon Dynamic Reflector Database; only PERMANENT reflectors are saved.

APPENDIX A

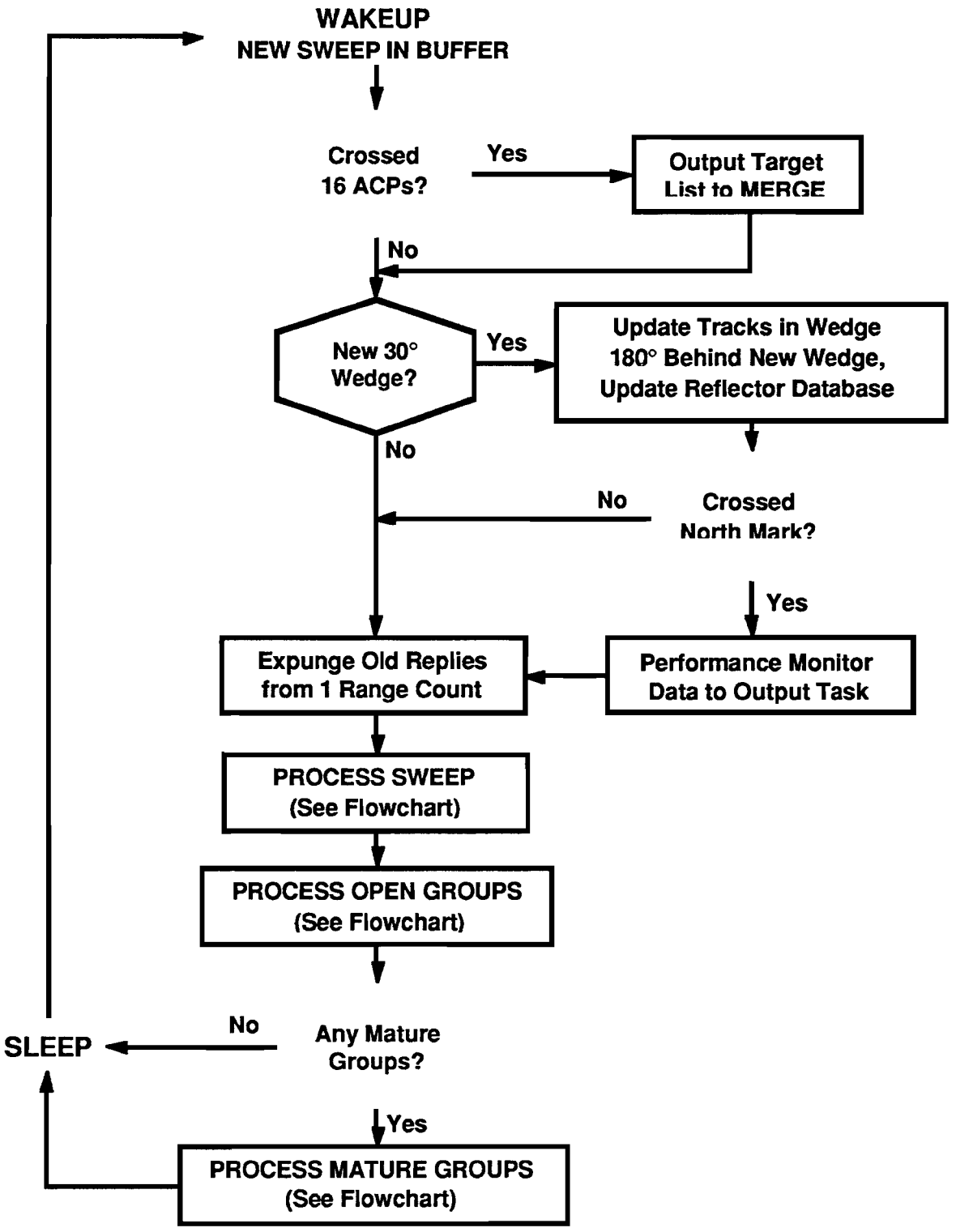


Figure 1. BTD Processing Task.

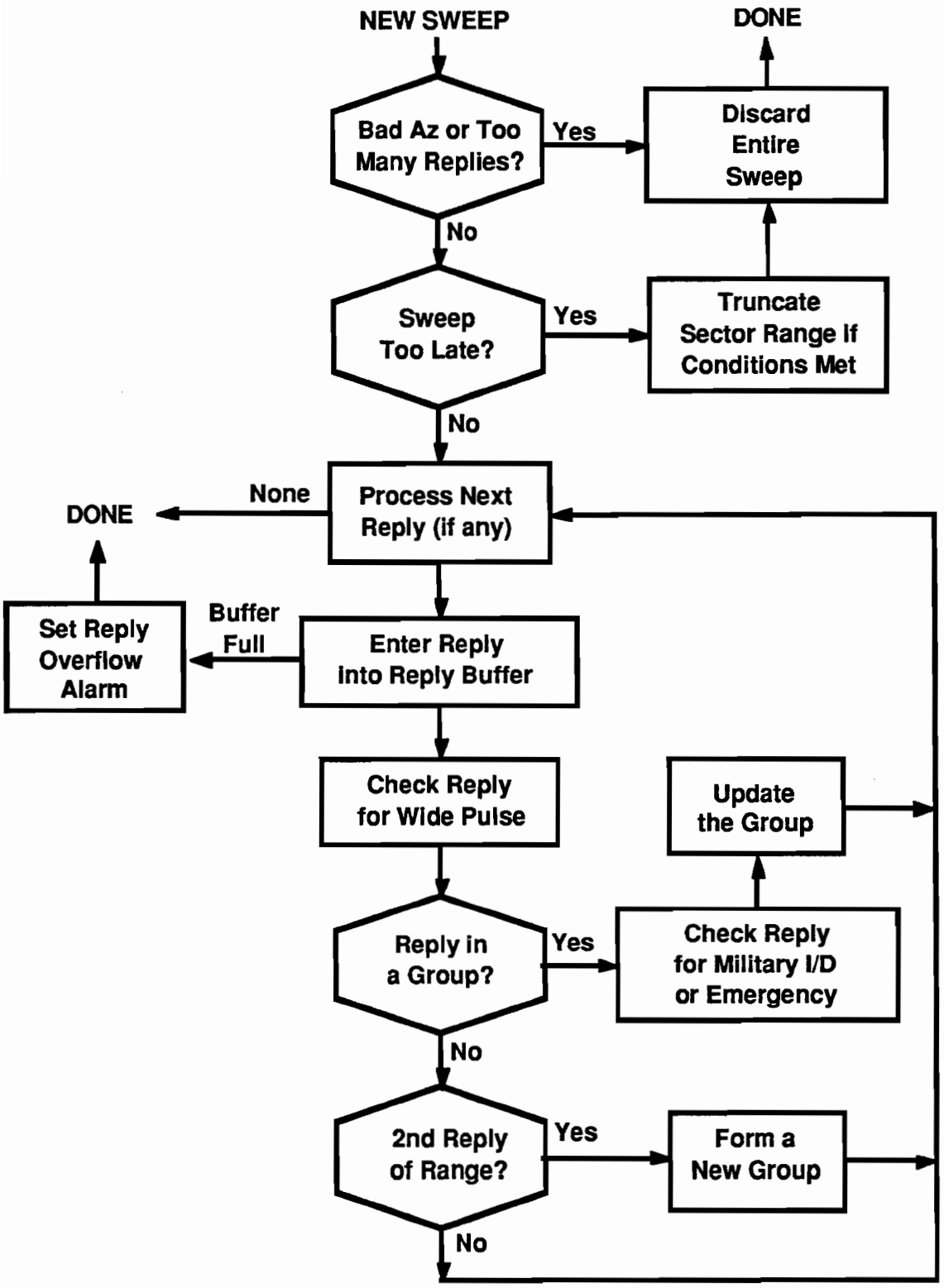


Figure 2. Process Sweep Routine.

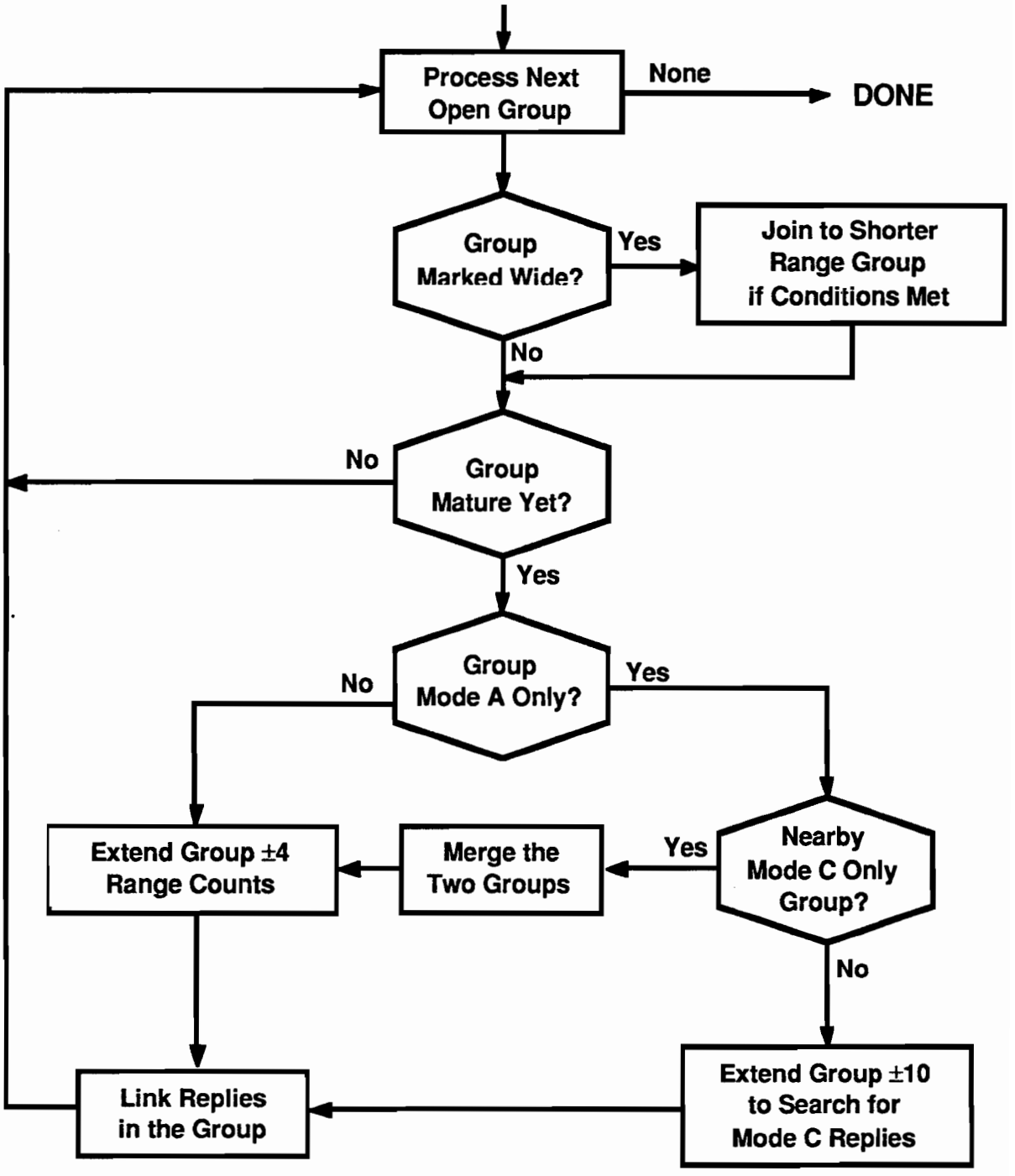


Figure 3. Process Open Groups Routine.

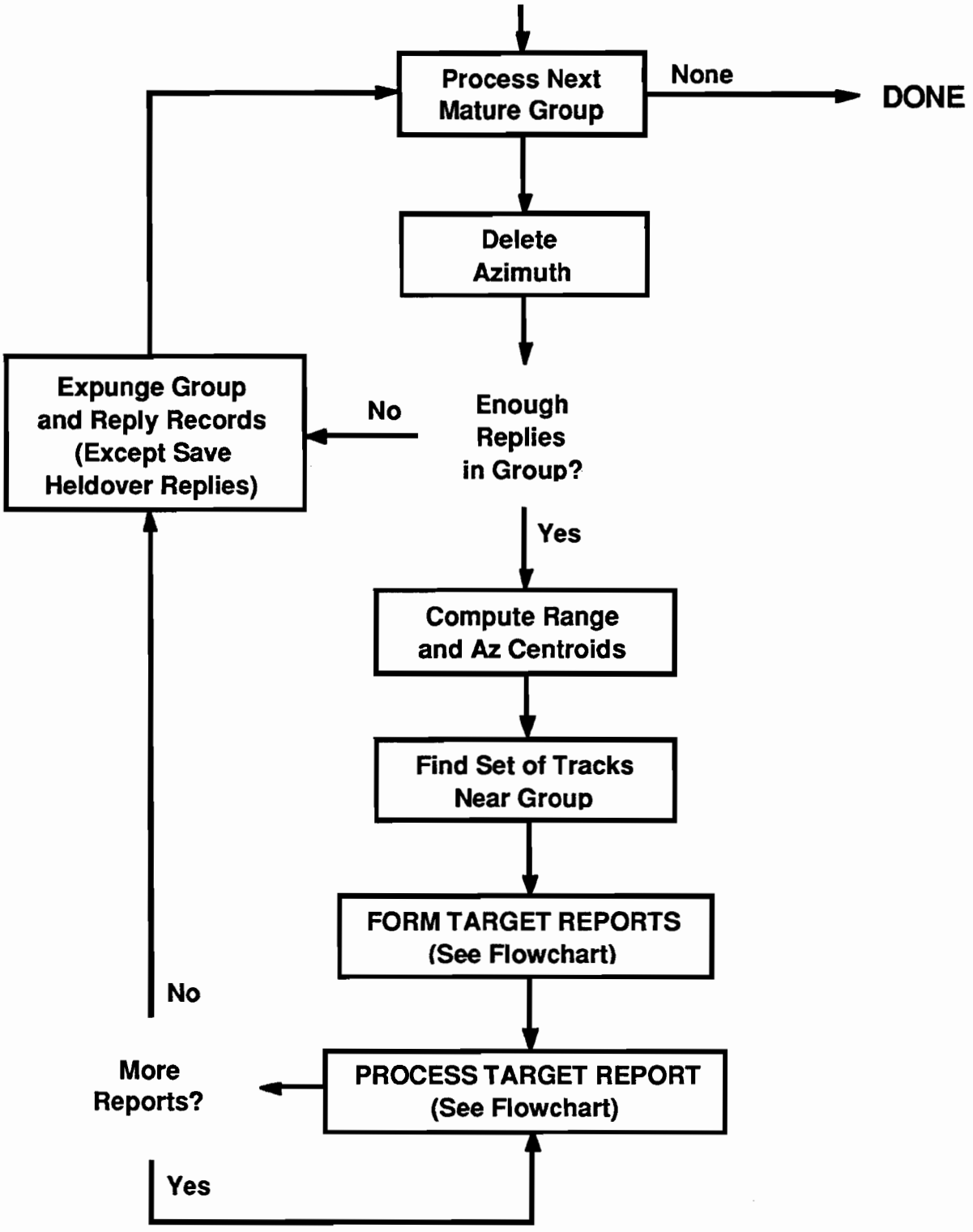


Figure 4. Process Mature Groups Routine.

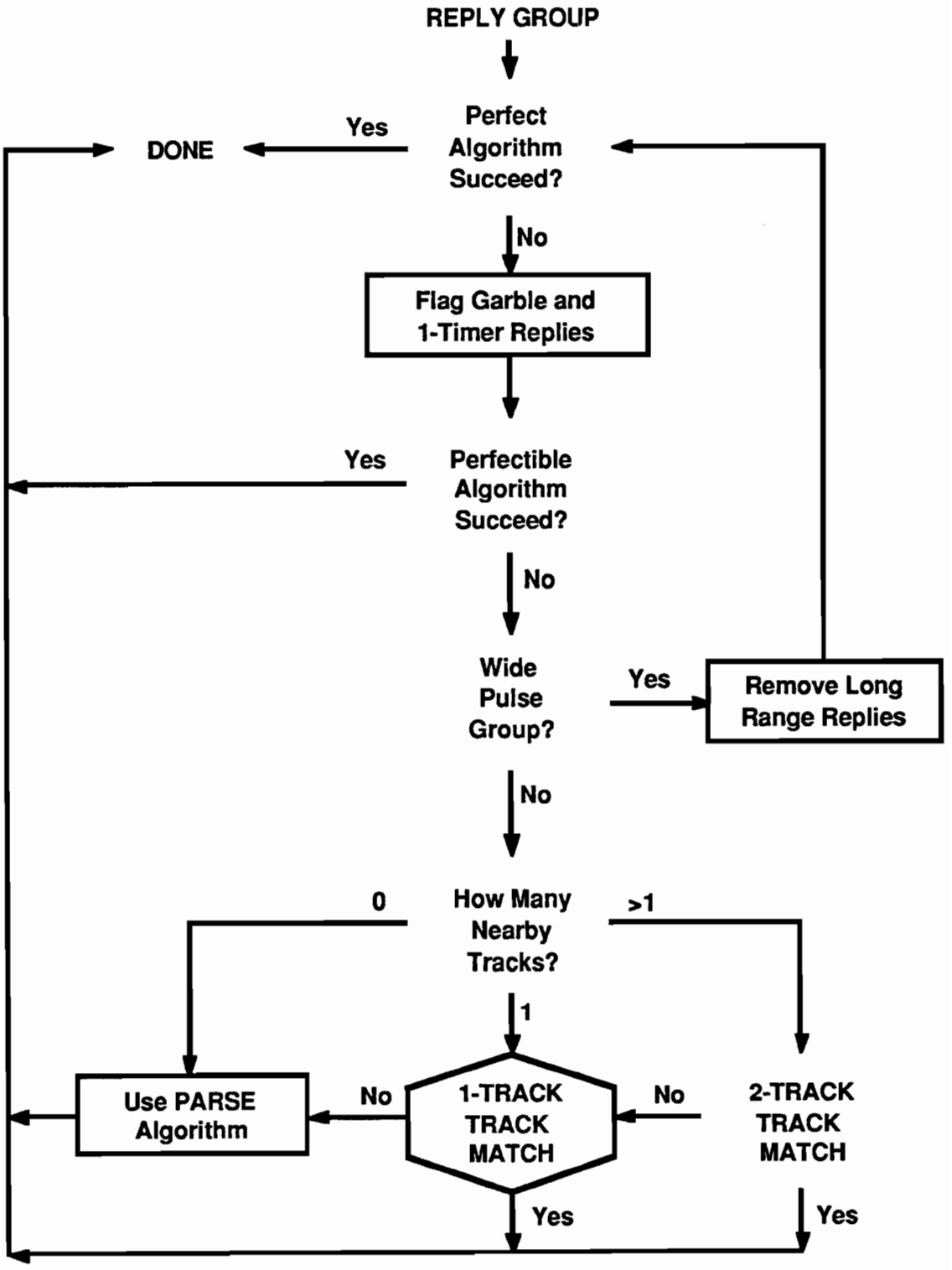


Figure 5. Form Target Report Routine.

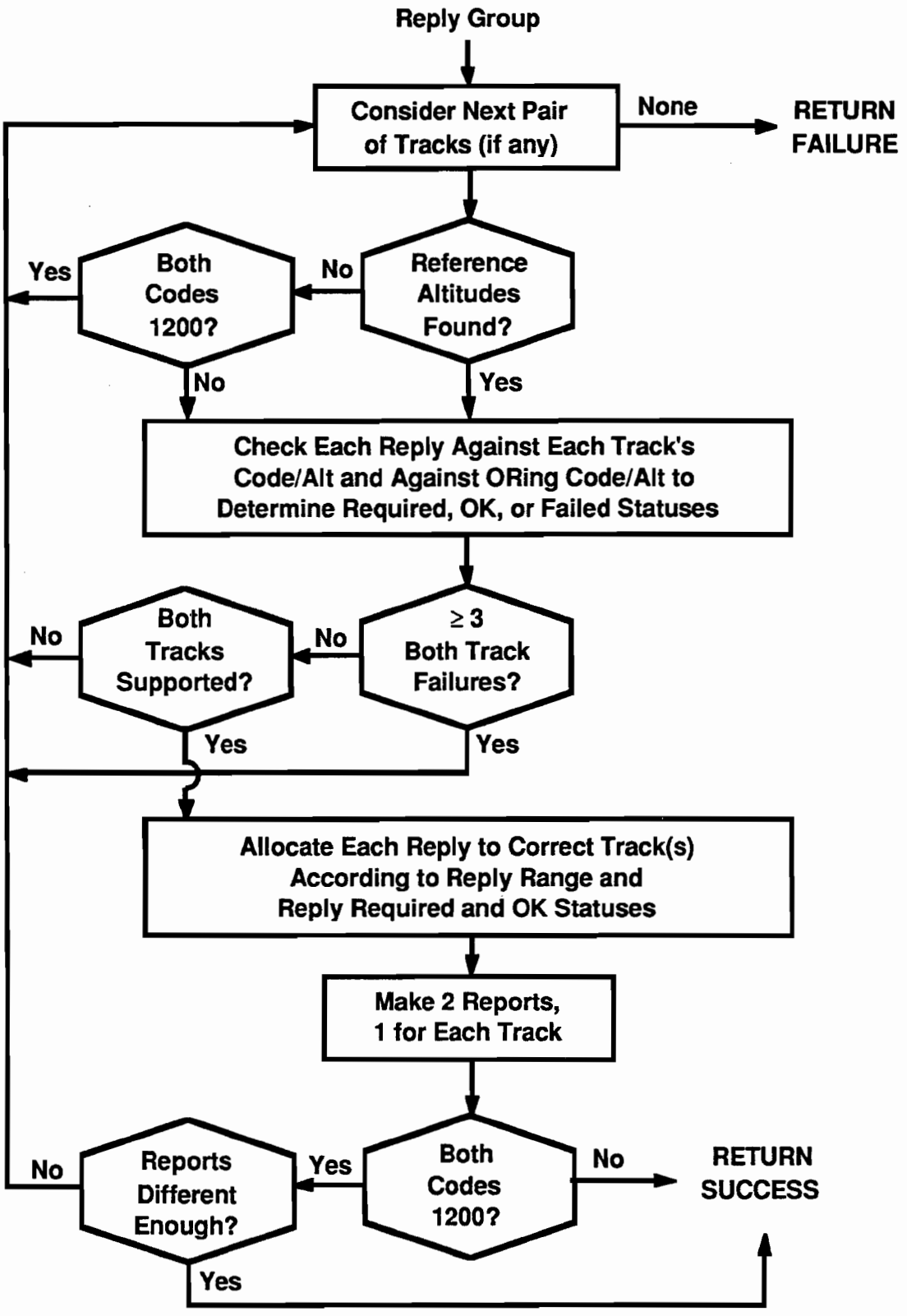


Figure 6. 2-Track Track Match Algorithm.

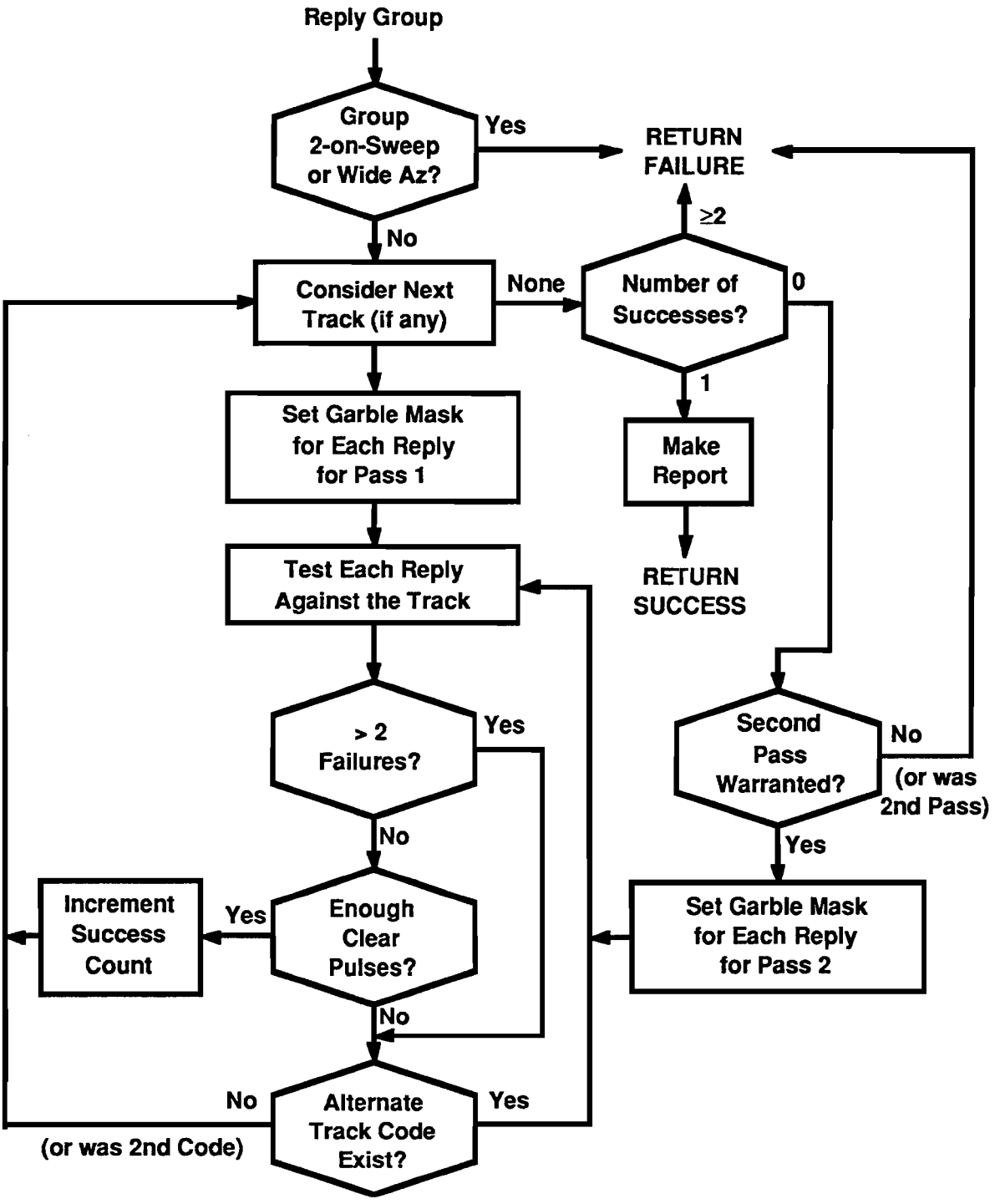


Figure 7. 1-Track Track Match Algorithm.

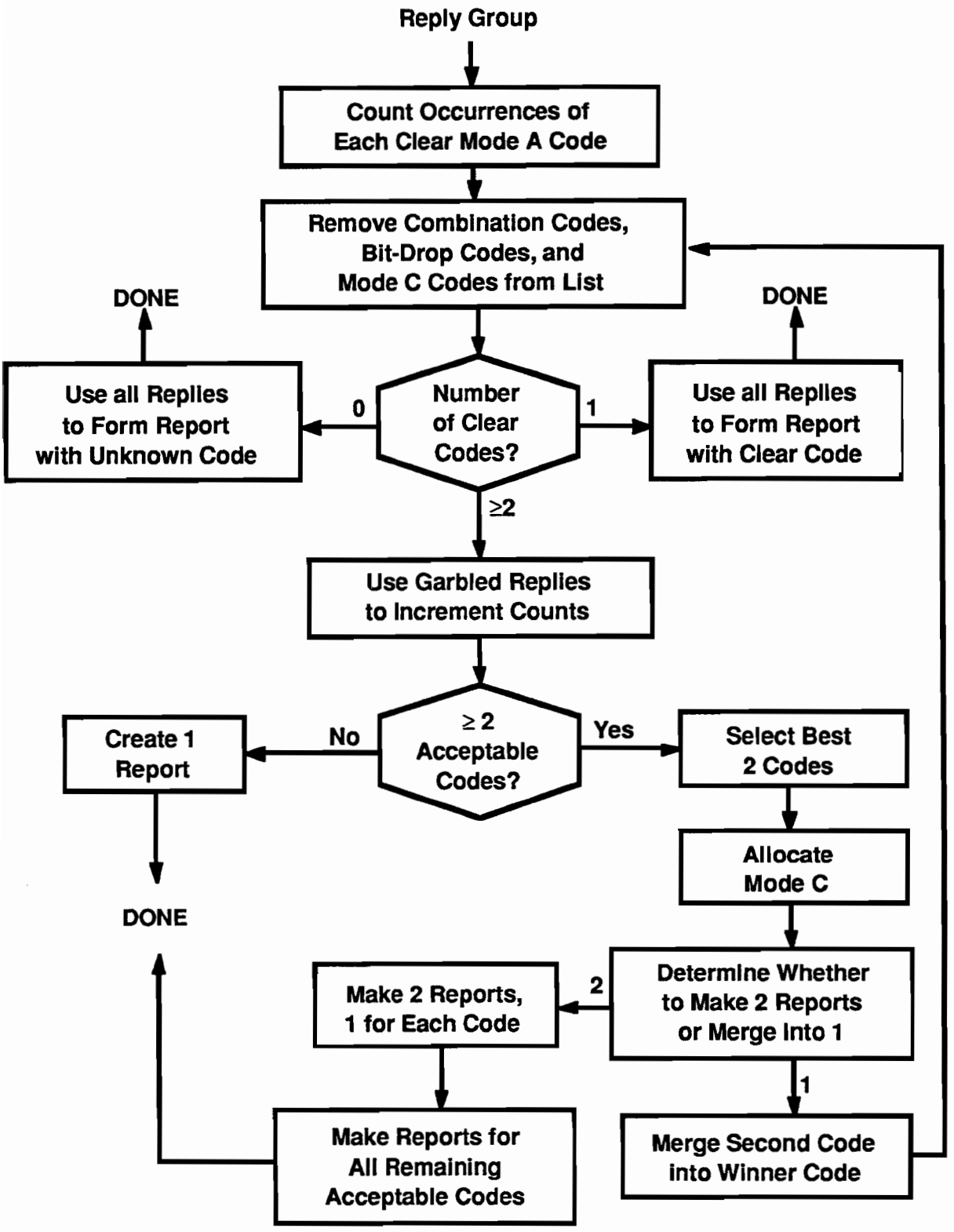


Figure 8. Parse Algorithm.

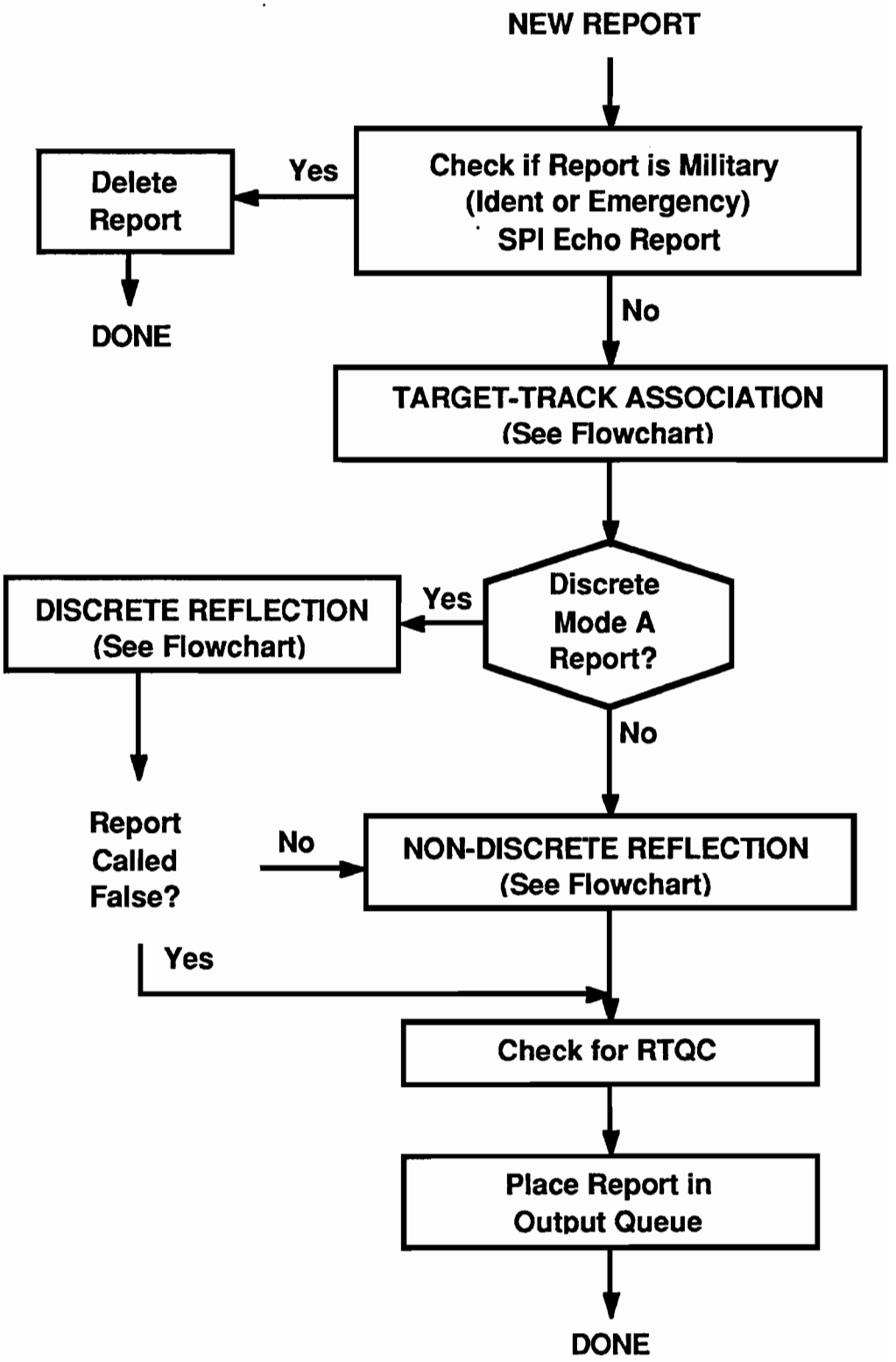


Figure 9. Process Target Report Routine.

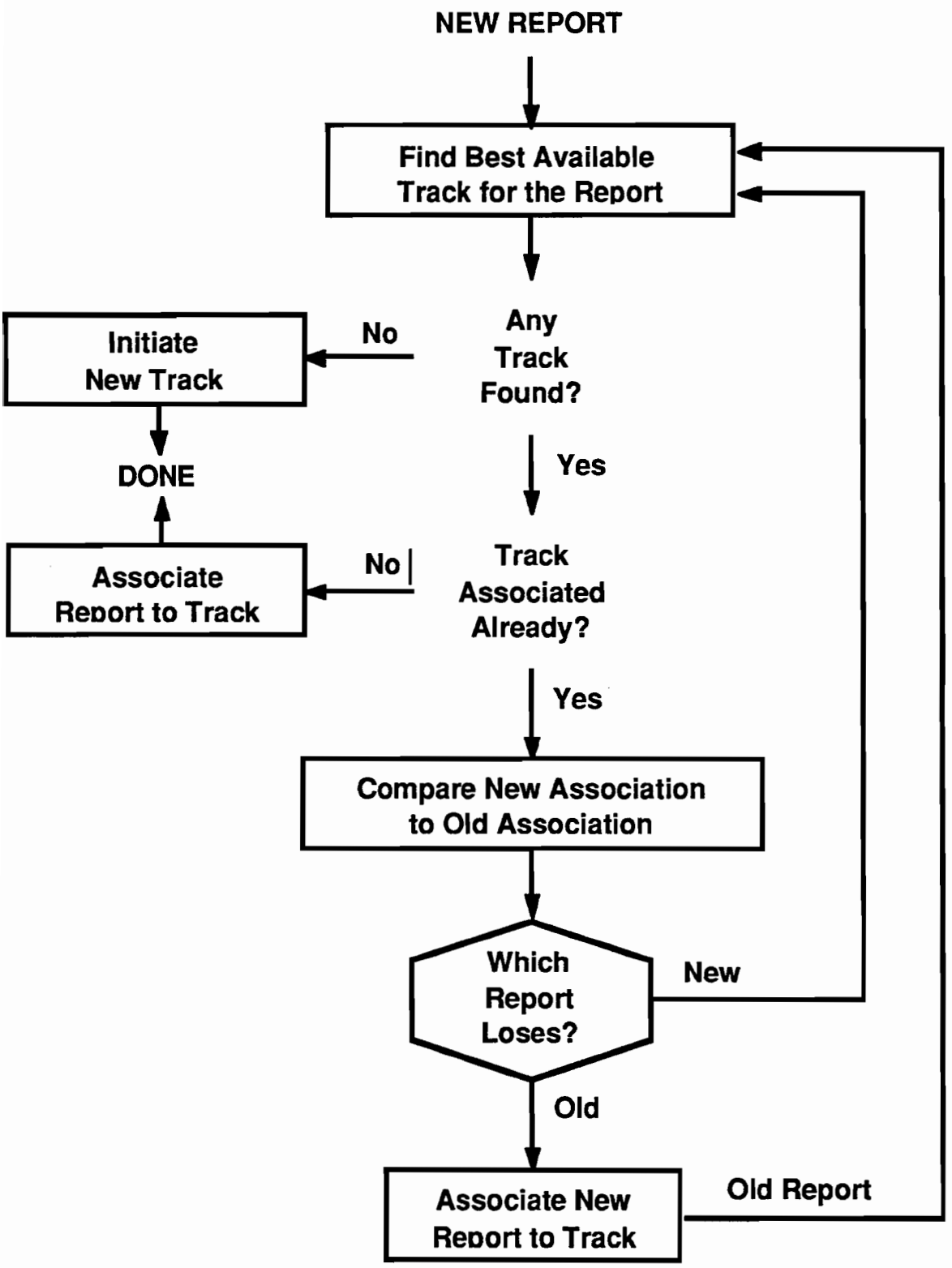


Figure 10. Target-Track Association.

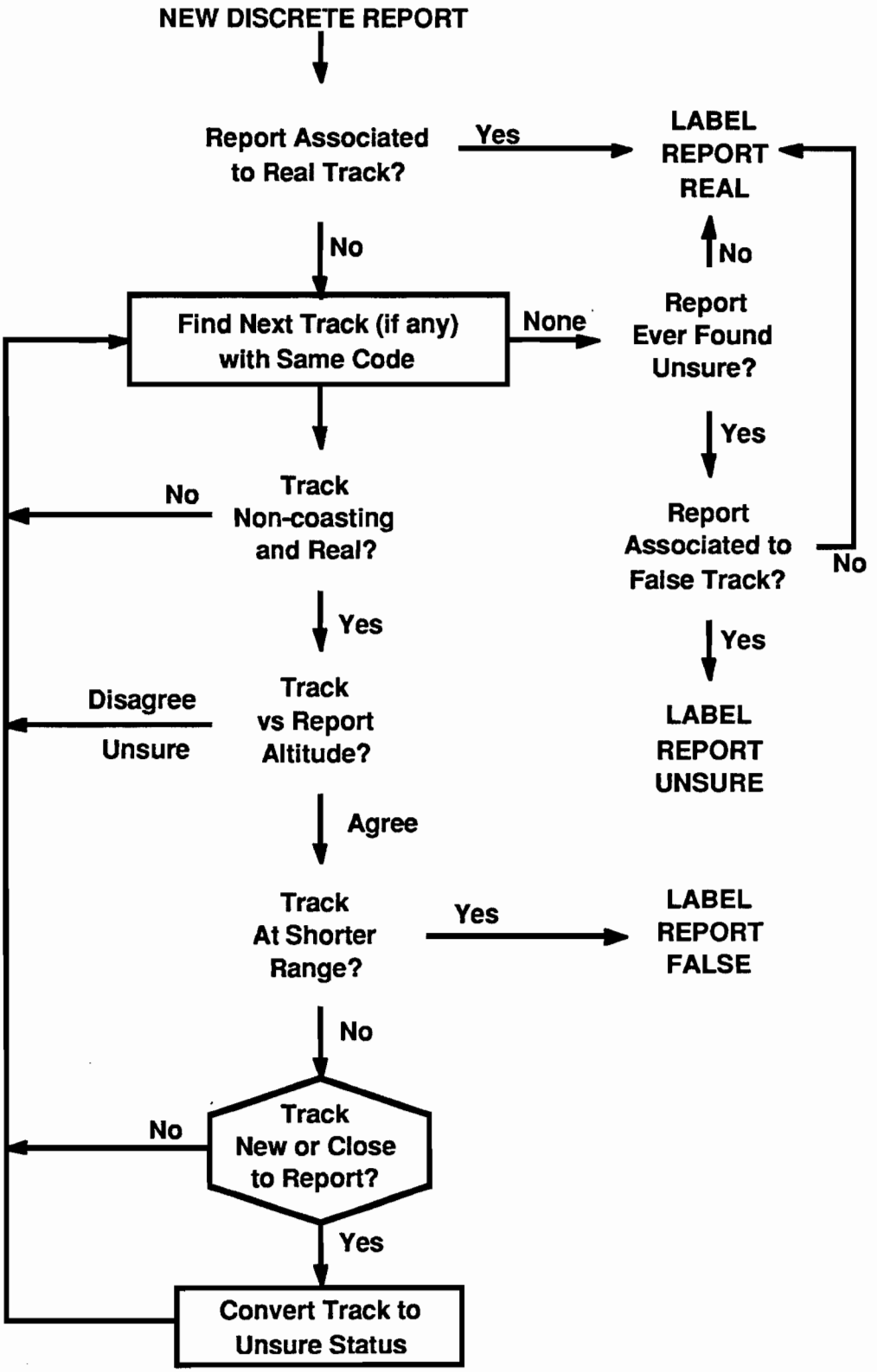


Figure 11. Discrete Reflection Processing.

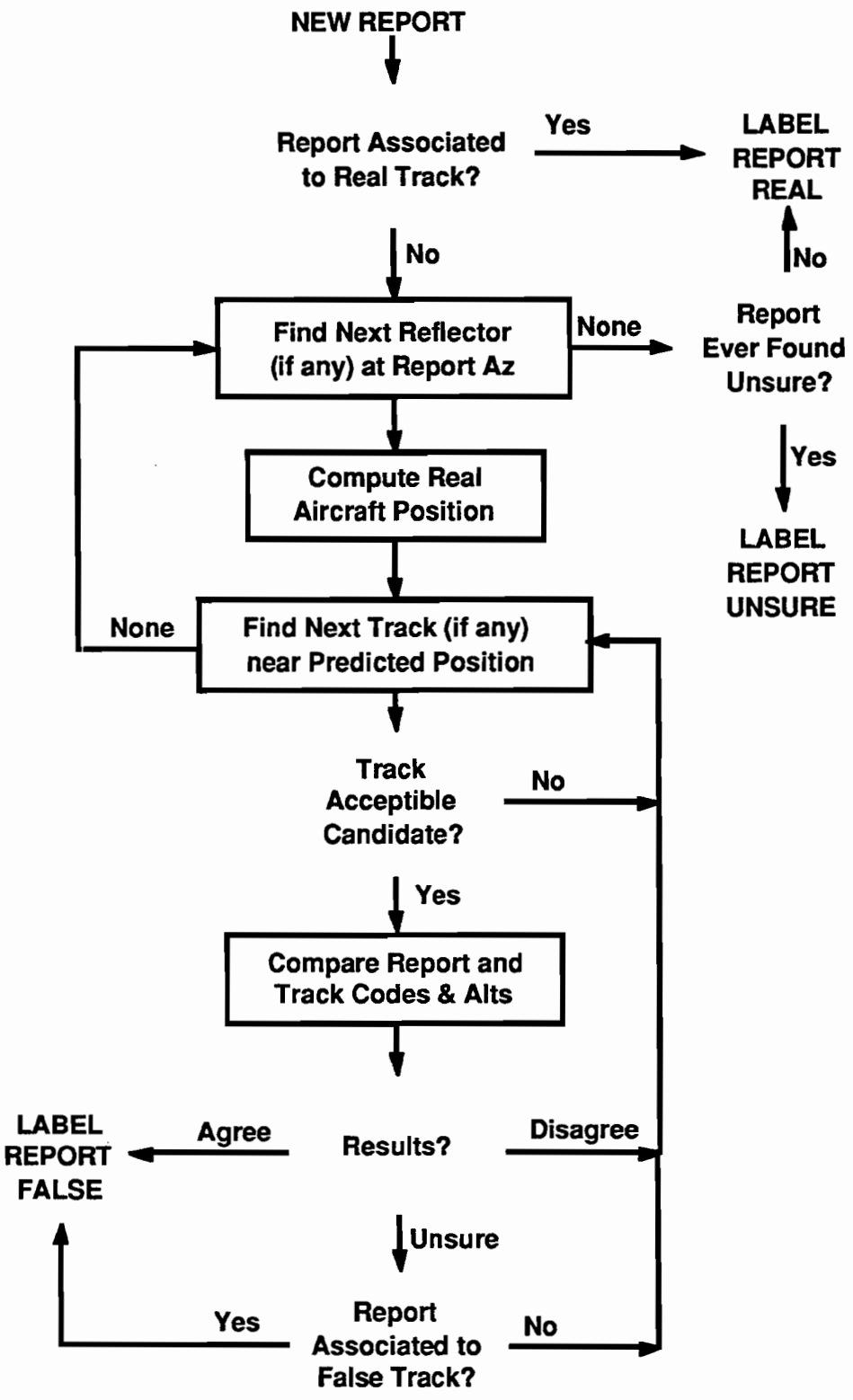


Figure 12. Non-Discrete Reflection Processing.

APPENDIX B

Variable Site Parameters (VSP)

The following is a list and description of the VSP settings for the 9-PAC Beacon Target Detection (BTD) task. These are provided to the 9-PAC by the ASP (using the RMS screens) at system startup. For each VSP, the valid range of values is provided, along with a suggested default value setting.

BTD VSPs (total of 42 words)

Name:	MINREP
Descr:	Minimum number of replies needed to declare a target. This VSP covers all interrogation modes.
Values:	3 through 6
Default:	4
Size:	1 word

Name:	MIN3A
Descr:	Minimum number of Mode 3/A replies needed to declare a target.
Values:	2 through 6
Default:	3
Size:	1 word

Name:	TGTRUN
Descr:	Nominal runlength assumed for any beacon target (ACPs).
Values:	0 through 111 ACPs
Default:	66
Size:	1 word

Name:	REFERRO
Descr:	Reflector merging orientation (direction it faces) error. This parameter (along with REFERRR and REFERRA and REFEXTA) will be used when deciding whether to add another reflector to the database. These parameters will be used to prevent multiple database entries due to small errors in reflector calculations.
Values:	0 through 333 ACPS
Default:	111
Size:	1 word

Name:	REFERRR
Descr:	Reflector merging range error, expressed in LSB 1/64 nmi.
Values:	0 through 32 (64ths of nmi)

Default: 13 (64ths of nmi)
Size: 1 word

Name: REFERRA
Descr: Reflector merging azimuth error.
Values: 0 through 333 ACPs
Default: 111
Size: 1 word

Name: REFEXTA
Descr: Reflector azimuth extension increment. This determines the azimuth extent covered by a reflector.
Values: 0 through 56 ACPs
Default: 22
Size: 1 word

Name: REFNOMR
Descr: Nominal reflector-based false target range box, expressed in LSB 1/64 nmi. This parameter, and REFNOMA, will be used by BTD to decide if a non-discrete target could have been generated by a reflector (something in the dynamic reflector database).
Values: 0 through 64 (64ths of nmi)
Default: 16 (64ths of nmi)
Size: 1 word

Name: REFNOMA
Descr: Nominal reflector-based false target azimuth box.
Values: 0 through 222 ACPs
Default: 45
Size: 1 word
Name: BAZBIAS
Descr: Beacon Azimuth Bias. This is sent from RMS in the form of a left justified 12-bit azimuth. The ASP will right justify this parameter before sending it to the 9-PAC board.
Values: 0 through 4095 ACPs
Default: 0
Size: 1 word

Name: BRNGBIA
Descr: Beacon Range Bias. This is sent from RMS as an unsigned integer with LSB 1/64 nmi. The ASP will send it on as is.
Values: Any valid range, in 1/64ths of nmi

Default: 0
Size: 1 word

Name: RNGOENA
Descr: Range Offset Enable (+0.5 nmi). RMS sends all ones to add a half nmi to the target's final range centroid. All zeroes means leave it alone. The ASP will transform these values to what C language prefers (1 to add 0.5 nmi, 0 to disable).
Values: 0 or 1
Default: 0
Size: 1 word

Name: RTQCRNG
Descr: RTQC Range (for detection of). This is sent from RMS as an unsigned integer with LSB 1/64 nmi. The ASP will send it on unchanged.
Values: Any valid range, in 64ths of nmi
Default: N/A
Size: 1 word

Name: RTQCAZ
Descr: RTQC Azimuth (for detection of). This is sent from RMS in the form of a left justified 12-bit azimuth. The ASP will right justify this value before sending it to the 9-PAC.
Values: 0 through 4095 ACPs
Default: N/A
Size: 1 word

Name: VSPV
Descr: Validation Upgrade Threshold. This is used to validate X-bits and SPI-flags. It is also used to validate codes, but only when BTD Track File information could not be used.
Values: 1 through 6
Default: 2
Size: 1 word

Name: RMATURE
Descr: Number of real report correlations needed to declare a real BTD track. Any report correlating to a real track can never be declared a false target.
Values: 0 through 5
Default: 3
Size: 1 word

Name: FMATURE
Descr: Number of false report correlations needed to declare a false BTD track.
Values: 0 through 5
Default: 2
Size: 1 word

Name: DRPCNT
Descr: Consecutive coasts needed to drop a mature (2 or more hits) BTD track.
Values: 3 through 10
Default: 5
Size: 1 word

Name: DRPCNTI
Descr: Consecutive coasts needed to drop an immature (newly created) BTD track.
Values: 1 through 5
Default: 2
Size: 1 word

Name: DISCALT
Descr: Altitude window for false target testing on discrete Mode 3/A targets. This parameter is expressed as an unsigned integer with LSB 100 feet.
Values: 0 through 500
Default: 2
Size: 1 word

Name: NONDALT
Descr: Altitude window for false target testing on non-discrete Mode 3/A targets. This parameter is expressed as an unsigned integer with LSB 100 feet.
Values: 0 through 500
Default: 1
Size: 1 word

Name: NONDCNT
Descr: Number of entries in non-discrete Mode 3/A code list (below). Note that by default, any code beginning with the octal digits 12 (e.g., 1200, 1201) or ending with the octal digits 00 (e.g., 1200, 5500) is always considered non-discrete.
Values: 0 through 20
Default: 0
Size: 1 word

Name:	NONDLIS
Descr:	List of discrete Mode 3/A beacon code values to actually treat as non-discrete (for example 1201 in LAX). The MIP will use default values of all zeroes (empty EEPROM).
Values:	any valid Mode 3/A code (0 through 7777 octal)
Default:	all 0's
Size:	20 words

APPENDIX C

Performance Monitor (PM)

The following is a list and description of the performance counts and alarms set by the BTD algorithms. These are sent to the ASR-9 ASP once every scan.

BTD Performance Counts (total of 28 words)

Name: TTGCNT
Descr: Test target (reply) detection count.
Size: 1 word

Name: PMCMPCT
Descr: Total number of targets declared.
Size: 1 word

Name: PMOUT
Descr: Total number of targets output to MERGE.
Size: 1 word

Name: PMRPYCT
Descr: Total number of replies received.
Size: 1 word

Name: FRTCNT
Descr: Fruit reply deletion count.
Size: 1 word

Name: PMMOD3V
Descr: Mode 3/A code validation count.
Size: 1 word

Name: PM3XVC
Descr: Mode 3/A X-bit validation count.
Size: 1 word

Name: PMMODCV
Descr: Mode C code validation count.
Size: 1 word

Name: PMMOD2V
Descr: Mode 2 code validation count.
Size: 1 word

Name: PM2XVC
Descr: Mode 2 X-bit validation count.
Size: 1 word

Name: PMSPIVC
Descr: Count of SPI validations.
Size: 1 word

Name: PMGRPCT
Descr: Total number of reply groups started.
Size: 1 word

Name: PMMULCT
Descr: Number of group declaring multiple targets (close in range).
Size: 1 word

Name: WEAKCNT
Descr: Number of groups rejected as too weak to make a target.
Size: 1 word

Name: PERFECT
Descr: Number of reply groups handled by Perfect algorithm.
Size: 1 word

Name: PERFIBL
Descr: Number of reply groups handled by Perfectible algorithm.
Size: 1 word

Name: TRKMAT
Descr: Reply groups handled by Single-Track Track Match alg.
Size: 1 word

Name: TRKMAT2
Descr: Reply groups handled by Two-Track Track Match alg.
Size: 1 word

Name: PARSE
Descr: Reply groups handled by Parse algorithm declaring 1 target.
Size: 1 word

Name: PARSE2
Descr: Reply groups handled by Parse alg. declaring >1 targets.
Size: 1 word

Name: WPULCNT
Descr: Number of wide-pulse transponder targets.
Size: 1 word

Name: MULTTRK
Descr: Number of reply groups with multiple BTD tracks nearby.
Size: 1 word

Name: FALSECT
Descr: Number of beacon reflection false targets.
Size: 1 word

Name: RFLPERM
Descr: Total number of permanent reflectors.
Size: 1 word

Name: RFLMAT
Descr: Total number of mature reflectors.
Size: 1 word

Name: RFLACT
Descr: Total number of active (immature) reflectors.
Size: 1 word

Name: TRKCNT
Descr: Total number of BTD Tracks.
Size: 1 word

Name: CSTCNT
Descr: Number of Coasting BTD Tracks.
Size: 1 word

BTD Performance Alarms (total of 3 words)

Name: PRTOVFL
Descr: BTD PRT Overload Alarm (> 42 replies on a sweep)
Size: 1 word

Name: ATFOVFL
Descr: BTD Report Overload Alarm
Size: 1 word

Name: VARALM1
Descr: BTD Azimuth Variance Alarm (not monotonic)
Size: 1 word

APPENDIX D

Glossary

Bit Difference:	The bit difference of two reply codes is given by the number of bits (1's) set in the exclusive or-ing of the two codes.
Imperfect Superset:	Reply code A is an Imperfect Superset of reply code B if every bit position set to 1 in B is also set to 1 in A, except for NDROPS permitted bit-drops in code A. Thus, the formula is the following: (Number of Bits(~A & B) <= NDROPS). NDROPS can be either 1 or 2.
Number of Bits:	The number of bits set for a reply code is just the number of bits set to 1.
True Superset:	Reply code A is a True Superset of reply code B if every bit position set to 1 in B is also set to 1 in A. That is, ((A & B) == B)