AD-A071 224 NAVY ELECTRONICS LAB SAN DIEGO CA F/G 17/1

A COMPUTER TECHNIQUE IMPLEMENTING THE NEL SONAR CLASSIFICATION --ETC(U)

MAY 61 W A HILLIX

UNCLASSIFIED NEL-TM-472 NL

| OF |

AD A071224

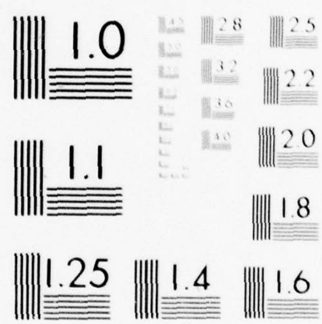
END

DATE

FILMED

8-79

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

UNCLASSIFIED

~~CONFIDENTIAL~~

MOST Project 3

NEL/Technical Memorandum 472

LEVEL II

S-F001 03 03, Task 8132

PROBLEM/NEL E1-11

Technical Memorandum No. TM-472

10 May 1961
date

NEL-TM-472

28p.

F00103

SF00103 03

A COMPUTER TECHNIQUE IMPLEMENTING THE NEL SONAR CLASSIFICATION SYSTEM, WITH AN APPLICATION TO NELIAC. (U)

W. A. Hillix — 2140
author — *code*

This document contains information affecting the national defense of the United States within the meaning of the Espionage Laws, Title 18, U.S.C., Sections 793 and 794. The transmission or the revelation of its contents, in any manner to an unauthorized person, is prohibited by law.

Extracts from this publication may be made to facilitate the preparation of other Department of Defense Publications. It is forbidden to make extracts for any other purpose without the specific approval of the Chief of the Bureau of Ships, except as provided for in the U. S. Navy Security Manual for Classified Matter.

U. S. Navy Electronics Laboratory, San Diego 52, California

620622-0129

The opinions expressed herein are those of the author(s) and are not necessarily the official views of the Navy Electronics Laboratory. If cited in the scientific literature, this document should be described as an unpublished memorandum.

253 550

UNCLASSIFIED

~~CONFIDENTIAL~~

DOWNGRADED AT 3-YEAR INTERVALS
DECLASSIFIED AFTER 12 YEARS
DOD DIR 5200.10

DISCLAIMER NOTICE

A COMPUTER TECHNIQUE IMPLEMENTING THE NEL SONAR CLASSIFICATION SYSTEM, WITH AN APPLICATION TO NELIAC

INTRODUCTION

The NEL Sonar Contact Classification System (Gales and Eady, 1958) has thus far been presented for fleet use only in the form of a tabular flipchart. Future availability of digital computers in some fleet contexts will make it advisable to have programs ready to effect automatic contact classification, given the inputs used by the NEL System. Automatic classification will reduce errors, increase speed of classification, and make it easy to gather data for the evaluation of system performance.

Although a previous program has been developed for the NEL System (Mooney, 1959), this program does not differentiate between non-submarine configurations in the tables and configurations which, though mathematically possible, are not included in the NEL tables.* In many cases, it might be a serious mistake if a contact which should be regarded as unknown were dismissed as a non-submarine. The present program duplicates the previous program functionally by using 26 cells of storage (where the previous program required 684). The distinction between non-submarine and unknown cases is preserved by using 26 additional cells. The actual classification in the present program takes place by doing either six basic instructions (if the input configuration falls into the submarine category) or twelve instructions (if the configuration represents a non-submarine or an unknown category). As usual, preparatory instructions are also involved; for details, see the NELIAC program in the appendix. The previous program

*LT Tom McDonald and LT Jerry Edwards have both written effective programs, but neither program has been officially reported.

required a search of a table 684 cells in length, and, therefore, would require about 342/9 - 38 times as many basic instructions to be executed, on the average, as the present program.

The present program has also been modified in form (though exactly the same logic is involved) for use in the NELIAC compiler, with a reported saving in compiling time of one third. It is, therefore, felt that the general method may be useful in still other related applications, where it may be a useful substitute for table lookup.

ESSENTIAL FEATURES OF THE NEL SYSTEM

The NEL System operates on a principle of combining sonar information from the audio display, the PPI scope, and the tactical range recorder. The classification of a contact depends on the total pattern, or configuration, of information obtained from the three displays, and thus includes a correlative power not inherent in previous classification procedures.

Table 1 lists the types of information which operators using the system are required to judge. From left to right, the table lists the sonar displays, the variables being judged on each display, and the values the judgments may have (for a detailed description of the variables and pictorial illustration of scale values, refer to Gales and Eady, 1958). The rules that limit the selection and combination of values are listed at the bottom of the table. Within these rules, there are 3072 different configurations of display values. Many of these are probably not realizable because the physical properties of objects producing sonar contacts are such that many configurations do not occur. To date, 1079 configurations of the 3072 possible are classified by the NEL System as being submarine or non-submarine.

Table 2 shows how the NEL System classifies each of these 1079 configurations. Since many of the configurations leading to the same decision have elements in common, it is possible to list the 1079 cases with their associated decisions in only 56 lines, each line representing one cluster of configurations. Table 2 differs slightly in format from what might be expected on the basis of table 1; space is economized by listing the ranges of axis angle and trace length that are acceptable, rather than giving the individual categorical values that are acceptable. In addition, the "none" category of axis angle is understood to be applicable in any case where a non-elongated pip shape is one of the values allowable within the configuration.

All of the possible legal input configurations that can be formed from the values listed on a given line lead to the same classification. On line 1, for example, we see that, regardless of doppler value, PPI shape, or axis angle, the combination of values of 3 on leading edge alignment, 3 on trailing edge alignment, and 10-29 yard trace length leads to a classification of non-submarine. In particular, the configuration up doppler, non-elongated pip, no axis angle, 3 leading edge alignment, 3 trailing edge alignment, and 10-29 yard trace length is one of the configurations in the cluster represented on line 1, and is classified non-submarine.

Table 2, as presented, makes no special provision for preventing illegal entries. For example, a configuration which would be a possible, though not a legal, member of the cluster on line one would be down doppler, double pip, 0-19 axis angle, 3 leading edge alignment, 3 trailing edge alignment, and 10-29 yard trace length. This combination of double pip with 0-19 degree axis angle is not allowable, as indicated by the rules accompanying table 1. If classifications are being made automatically, special checks must be incorporated to see that such illegal configurations are detected; the present program incorporates these checks.

If classification were to be accomplished by the human operator, table 2 would not be in acceptable form for his use. Locating a given configuration would be too time-consuming. The flipcharts that have been used in the past have accordingly not been presented in this reduced form. One of the advantages of automatic classification is that this reduced form can be used easily within the computer program.

Table 3 presents part of the same information as table 2, but in a form which can be used by the computer. Table 3 is limited to the 28 non-submarine clusters, since it is presented only for illustrative purposes. The presence of a value in a cluster is indicated by putting a 1 in the appropriate column. In order to do this, it is necessary to list each value of each variable explicitly. Submarine and reentry clusters would be handled in exactly the same way. The reentry cases are those cases in which highlights have been judged in connection with an axis angle of 20° or greater; edge alignment should replace the highlight judgment in all such cases.

A computer program could classify automatically with information in the form indicated in table 3. The general method would be to compare the configuration entered by the classification keyset with each of the clusters in succession until that cluster was found to which the configuration belonged. This is wasteful because most of the comparisons made will be made with clusters to which the configuration does not belong.

Table 4 presents the same information as table 3 in another form. Table 4 is constructed directly from table 3 by interchanging rows and columns; the rows in table 3 are the columns of table 4. Each row in table 4 represents the role of a particular value of a particular variable, as indicated by the row labels in the rightmost column. The columns represent the values acceptable in a particular cluster. Each row of this matrix fits conveniently into a 30-bit computer word, with two positions left over. Twenty-six computer words are needed to represent the non-submarine clusters. Twenty-six more are needed for the submarine and re-entry clusters. These words are presented in octal form in the dimensioning statement in the appendix; the order in which the words are presented is different because of the necessity to make the table compatible with the numbers assigned to the values by the keyset, and so that no blank words will be left within the computer.

Data in the form indicated in table 4 can be used to determine the class membership (submarine or non-submarine) of a configuration with a minimum number of operations. The classification keyset provides a number for each variable; the number represents the operator's judgment of the value of that variable. The computer uses these numbers as indices to select the computer words which represent those values. If, for example, the classification keyset operator presses the keys for up doppler, elongated pip, 0-19 axis angle, 3 leading edge alignment, 3 trailing edge alignment, and 10-29 yard trace length, the computer will select the corresponding words to work with, as follows:

011 010 000 000 000 000 111 000 000 000 (up doppler)
011 000 000 111 111 011 110 111 110 100 (elongated pip)
010 111 111 110 000 111 100 110 000 110 (0-19 axis angle)
010 000 011 110 110 110 100 100 111 000 (3 leading edge alignment)
010 000 010 101 001 101 111 010 100 110 (3 trailing edge alignment)
011 111 000 000 000 000 000 000 000 000 (10-29 yard trace length)

The six words above are identical to the corresponding rows in table 4, except that the left-most and right-most bit positions, which do not represent any cluster, are indicated above and not in table 4.

In table 4, it will be recalled, each column or bit position, represented a cluster. That is, the column representing a cluster has ones in all those rows representing values acceptable in the cluster. Putting it the other way around, every computer word representing a value which is part of a cluster will have a one in the bit position representing that cluster. However, if any one of a set of values is not part of a cluster, that value will have a zero in the bit position representing the cluster.

In the six words above, there is a one in each of the words in the second bit position from the left (position 28). This bit position represents cluster one (line one in table 2). No other bit position contains a one in every word, because the values represented are not all a part of any other cluster. It is always true that any six values belong to no more than one cluster.

The logical product instruction, which is a part of the repertoire of most digital computers, is a very convenient method of checking whether all words in any given set have a 1 in a common bit position, and, therefore, all belong to a cluster. In the present example, the logical product of the up doppler and elongated pip words is formed; then the logical product of the previous logical product and the 0-19 axis angle word is formed; and so on through the remaining three words. Since there will be a 1 in any given bit position of a logical product only if there was a 1 in that bit position in both the constituent words, the five successive logical products obtained in the above example will be as follows:

```
011 000 000 000 000 000 110 000 000 000   (1)
010 000 000 000 000 000 100 000 000 000   (2)
010 000 000 000 000 000 100 000 000 000   (3)
010 000 000 000 000 000 100 000 000 000   (4)
010 000 000 000 000 000 000 000 000 000   (5)
```

Thus, at each stage, ones remain in those bit positions representing clusters to which the values entering the logical product all belong. When the logical product of all six value words has been obtained, it will contain a one if and only if all six values were part of one cluster. The program has the submarine clusters represented in one group of words, and the non-submarine clusters represented in another group of words. The program first forms the logical product of the six words selected from the submarine cluster portion. Then it checks to see if there is a one remaining in the logical product. If a one remains, the configuration is classified submarine. If no one remains, the program selects the six words representing the six values from the non-submarine portion of the computer storage. The logical product of these is formed and checked to see whether a one remains; if so, the contact is classified non-submarine. If not, the contact is classified unknown.

The foregoing outlines the logic involved. For simplicity of exposition, the reentry condition was ignored; however, the submarine words actually contain 4 bit positions representing reentry clusters. If a one occurs in a logical product obtained from "submarine" words, submarine classifications have to be distinguished from reentry classifications via a comparison statement. Since the four least significant bit positions (rightmost) are used for configurations indicating that edge alignment judgments should replace a highlight entry, the final value of the logical product will be no greater than 8 if a reentry condition is indicated; otherwise, a submarine classification is indicated.

APPLICATION OF THE LOGICAL PRODUCT METHOD TO NELIAC

LT T. McDonald, who helped the author get the classification program into acceptable NELIAC, also suggested to the author that a similar method might be applicable to the problem of selecting the correct subroutines to jump to in the NELIAC compiling process. The author worked out the general method for making this application, and LTs A. E. Lemay and J. E. White wrote the routine. They report that the compiler with the routine incorporated ran about 1/3 faster than it previously had, and had some other desirable features.

The NELIAC application is really just another instance of a classification problem. The compiling program examines the current operator symbol in combination with the next operator symbol, and then "classifies" the combination according to which subroutine will be needed to generate the machine instructions for carrying out the indicated operation. Table 5 shows the legal current operators in the left-hand column, the legal next operators in the top row, and the numbers of the desired subroutines as entries in the table.

In the compiler application, there are 20 possible current operators and 24 possible next operators for a total of 480 combinations of current operator and next operator. This number is immediately reduced to 437 (19 x 23), since the "or" and "and" symbols are treated as identical for purposes of selecting a subroutine. The particular combination of current operator and next operator determines which of 27 subroutines is to be used, if the combination is legal; if the combination is not legal, a **fault** condition should be indicated. This gives a total of 28 outputs needed, as contrasted with four final outputs for the classification program. The small number of outputs in the classification program made it easiest to distinguish between outputs by simply knowing at which stage in the program the output occurred and by asking one special question to distinguish reentry cases. In the compiler, the larger number of outputs made it necessary to determine which bit position was filled when the output occurred.

A problem occurred in the writing of the NELIAC routine which had not occurred in the classification routine. Some of the logical products performed between the current operator words and the next operator words led to an ambiguity, that is, more than one bit position contained a 1. Since each bit position represents a separate output, it would not be possible to determine which output was correct without special questions which would slow the routine. LTs White and Lemay solved the problem ingeniously by renaming the outputs which led to ambiguity, that is, by shifting troublesome outputs to a new bit position. For example, subroutine 10 as it occurred in some positions was renamed 27. This routine then had two names at an intermediate stage in the program. Another look at table 3 will show what was involved more clearly. Consider the combination of comma as current operator with colon as next operator. For this combination the correct subroutine is number 29. However, the comma shares subroutine 10 with several next operators, and subroutine 10 is also used when the current operator is plus and the next operator colon. Therefore, the words in the computer for a current operator of comma and the next operator of colon have two bit positions in common: 10 and 29. By renaming the case where the plus and colon use subroutine 10, calling this case 27, the ambiguity is eliminated. Several other occurrences of 10 were also renamed 27, and other ambiguities were eliminated in analogous fashion; all ambiguities are indicated in the table by stars. The jump table is then constructed so that 10 and 27 are recombined and, whether bit position 10 or 27 is filled, the same subroutine is jumped to.

GENERAL USEFULNESS OF THE METHOD

Because of the large number of outputs, the number of ambiguities was small, and it was possible to proceed by renaming a few cases. If the number of outputs had been smaller, or if the table outputs had clustered in less orderly fashion, it might have been necessary to construct a minimal table via the same type of procedures that were used for the NEL Classification System. That is, the known configurations could be listed separately at first, and then combined as far as possible without introducing any ambiguity. The lines in such a minimal table would represent the intermediate outputs necessary to avoid ambiguity. Several lines might represent the same final outcome; in the case of the NEL System tables, as many as 28 lines represent a single final outcome; in the NELIAC compiler, only two names needed to be used for any single subroutine, so only two intermediate outputs are combined for any final output.

Both the NEL System and the NELIAC tables lend themselves well to the logical product method partly because there is in each case a single output for a large proportion of the possible configurations. In the first case, 1993 configurations out of the 3072 legal configurations lead to an "unknown" output. In the compiler case, 200 of 437 combinations are illegal. In both cases, this commonest output could be assigned the condition, logical product = 0. The ability to treat the commonest output in this economical fashion is another advantage of the logical product method. If the commonest output had to be treated exactly like the other outputs, many more ambiguities would arise, and the minimal table would have to be much larger.

The usefulness of the logical product method as a substitute for table search methods in these two applications leads one to inquire when, in general, such a method would be useful and when it would not. The strength of the logical product method is that it takes a short time to "search" the input combinations and determine whether a legal output exists. Then the outputs must be "searched" sufficiently to determine which final output has resulted. In the NELIAC routine, this search is carried out by successive shifting, counting the shifts, and choosing the output as determined by the shift counter when a "1" appears in the rightmost position. In the classification routine, the stage in the program at which an output was obtained was sufficient to specify the output. In either case, the logical product method involves an output, rather than an input, search. Thus, in general, if there are fewer outputs than inputs, it would seem preferable to use an output search. In addition, instructions are available that make this output search very fast, at least where the number of outputs is $\leq$ the number of bits in a computer word. If a computer word specifies 30 outputs according to which of its 30 bit positions contain a 1, it is quite fast to initiate a repeat of a right shift instruction for 30 times with a skip condition when the register turns to 0; the number of shifts required to turn the register to 0 is available in the repeat index. No special comparison instructions are required.

The weakness of the logical product method is that special precautions are necessary in order to avoid ambiguities. As we have seen, if ambiguities occur, the number of intermediate outputs must be enlarged in order to eliminate the ambiguities. In the classification program, there were actually 56 intermediate outputs corresponding to the lines of the table. These 56 were recombined partially to give only three final outputs, plus the fourth when none of the 56 positions contained a 1. The technique of multiplying logical products is useful only if the number of <u>intermediate</u> outputs is relatively small. This means that appreciable numbers of input configurations must cluster on the lines of a table, where each line represents an output; or that in a table like the NELIAC distribution table, the outputs must cluster on rows, or in columns, or in both. Where these clustering effects are observed, it seems safe to say that table storage methods should be replaced by methods which find an output by obtaining logical products of words, where each word describes the role of an individual input value. Of course, there are other factors to consider in a specific case; for example, does the number of intermediate outputs nearly equal some multiple of the number of bits in a computer word? If it does, each performance of a logical product produces more information, and storage space is better used. If it does not, a method like table storage which is abstractly inferior may actually be better.

One application which would be expected to show the characteristics associated with efficient use of the logical product method would be the making of tactical decisions in computer-aided weapons systems operating in real time. In such cases, decisions would be likely to depend upon the existing values of several variables. The number of possible decisions would probably be quite limited compared to the number of configurations of values of varisbles. These are exactly the requirements for situations where the present method might be helpful.

SUMMARY

A computer technique is described which efficiently implements the NEL Sonar Contact Classification System. The correct output is determined by a logical product method rather than by the usual table storage and look up methods. The computer gets its inputs from a keyset designed specifically for sonar classification. The contact classification given by the computer will be identical to that made from tables based on the current NEL Sonar Contact Classification System for scanning sonars.

A logically identical technique turned out to have an application within the NELIAC compiling program, where it was reported to save 1/3 of the compiling time. The success of the method in these two applications led to some examination of the general usefulness of the method as a

substitute for table storage and table look up in computer programming. One type of problem where this method should be efficient is the making of decisions in real time systems, where the decisions are based on the existing values of several variables.

REFERENCES

Gales, R. S. and Eady, H. R., "Human Factors in Sonar Detection and Classification", J. Underwater Acoustics, July 1958.

Mooney, H. L., "Automation of Sonar Echo Classification with the NTDS Unit Computer", NEL TM-366, 13 October 1959.

TABLE 1. INPUT VARIABLES TO THE NEL CLASSIFICATION SYSTEM

Display	Variable*						
Audio	Doppler	Up	Down	None			
PPI	Pip Shape	Non-Elongated	Elongated	Elongated, with wake	Double Pip**		
	Axis Angle	None***	0-19	20-59	60-74	75-84	85-90
Tactical Range Recorder	Leading Edge Alignment	Very Good (1)	Good (2)	Fair to Poor (3)	Wake**** Effect (W)		
	Trailing Edge Alignments	Very Good (1)	Good (2)	Fair to Poor (3)	Wake Effect (W)		
	Highlights*****	Present	Absent				
	Trace Length	10-29	30-69	70-129	130-over		

* A single value must be chosen for each variable in order for the NEL system to classify a contact.

** The "double pip" is a special category allowed only when the axis angle is judged to be between 85 and 90 degrees.

*** If the pip is non-elongated, it is impossible to judge the angle between the pip and the bearing line to the contact; hence, "none" must be used when the pip is non-elongated.

**** Since a submarine could not conceivably produce wake on both ends, wake effect can be judged on only one side of the trace, and the "W,W" combination is illegal.

***** If there is a highlight effect, this judgment should replace the judgment of alignment on both leading and trailing edges; in this case, the configuration will contain only 5 values rather than 6.

TABLE 2. MINIMAL REPRESENTATION OF THE NEL CLASSIFICATION SYSTEM

CLUSTER	DOPPLER	PPI SHAPE	AXIS ANGLE	ALIGNMENT	TRACE LENGTH	CLASS
1.	U D N	NE E EW DP	0-90	3-3	10-29	N
2.	U D N	E EW	20-59	1,2,-1,2	10-29	N
3.	N	NE	-	2-2	10-29	N
4.	U D	NE	-	1,2-1,2	10-29	N
5.	N	EW	0-19	1,2-1,2	10-29	N
6.	N	NE	-	2-2	30-129	N
7.	N	NE	-	1,2,3-3,W	30-129	N
8.	N	NE	-	3,W-1,2	30-129	N
9.	N	E	0-59	1,2,3-3,W	30-129	N
10.	N	E	0-59	3,W-1,2	30-129	N
11.	N	E	60-90	1,2-3,W	30-69	N
12.	N	E	60-90	3,W-1,2	30-69	N
13.	N	E EW	60-90	1,2,3,W-1,2	70-129	N
14.	N	E EW	60-90	1,2-3,W	70-129	N
15.	N	EW	0-59	1,2,3,W-1,2, 3,W	30-129	N
16.	N	NE E EW DP	0-90	1,2,3,W-1,2	130-over	N
17.	N	NE E EW DP	0-90	1,2-3,W	130-over	N
18.	U	NE E EW	0-90	W,3-1,2,3,W	30-over	N
19.	U	E	60-90	1,2-1,2,3,W	130-over	N
20.	U	EW	75-90	1,2-1,2,3,W	130-over	N
21.	D	NE E EW	0-59	2,3-W	30-over	N
22.	D	NE E EW	0-90	1-3,W	30-over	N
23.	D	E EW	60-90	2-W	30-over	N
24.	D N	E EW	60-90	3,W-3,W	30-over	N

TABLE 2 (Cont'd)

CLUSTER	DOPPLER	PPI SHAPE	AXIS ANGLE	ALIGNMENT	TRACE LENGTH	CLASS
25.	D	E	60-90	1,2,3,W-1,2	130-over	N
26.	D	EW	75-90	1,2,3,W-1,2	130-over	N
27.	N	NE E	0-19	H	130-over	N
28.	N	EW	0-59	H	30-over	N
29.	N	NE	-	1-1,2	10-29	S
30.	N	NE	-	2-1	10-29	S
31.	N	E	0-19	1,2-1,2	10-29	S
32.	N	DP	85-90	1,2-1,2	10-29	S
33.	N	NE	-	1,2-1	30-129	S
34.	N	NE	-	1-2	30-129	S
35.	N	E	0-59	1,2-1,2	30-129	S
36.	N	E	60-90	1,2-1,2	30-69	S
37.	N	EW	60-90	1,2-1,2,3,W	30-69	S
38.	N	EW	60-90	3,W-1,2	30-69	S
39.	N	NE E	0-19	H	30-129	S
40.	U D N	E EW	60-90	1,2-1,2	10-29	S
41.	U D	E EW	0-19	1,2-1,2	10-29	S
42.	U	NE E EW	0-59	1,2-1,2,3,W	30-over	S
43.	U	E	60-90	1,2-1,2,3,W	30-129	S
44.	U	EW	60-74	1,2-1,2,3,W	30-over	S
45.	U	EW	75-90	1,2-1,2,3,W	30-129	S
46.	D	NE E EW	0-59	1,2,3,W-1,2	30-over	S
47.	D	NE E EW	0-59	2,3,W-3	30-over	S
48.	D	E	60-90	1,2,3,W-1,2	30-129	S

TABLE 2 (Cont'd)

CLUSTER	DOPPLER	PPI SHAPE	AXIS ANGLE	ALIGNMENT	TRACE LENGTH	CLASS
49.	D	EW	60-74	1,2,3,W-1,2	30-over	S
50.	D	EW	75-90	1,2,3,W-1,2	30-129	S
51.	U D	NE E	0-19	H	30-over	S
52.	U D	EW	0-19	H	30-over	S
53.	U D N	E	20-90	H	30-129	Not a high-light case, enter edge alignment
54.	U D N	EW	60-90	H	30-129	"
55.	U D	EW	20-59	H	30-over	"
56.	N	DP	85-90	H	10-29	"

CONFIDENTIAL

TABLE 3. NON SUBMARINE PORTION OF TABLE 2 IN REVISED FORMAT

Line	Doppler			PPI Shape				Axis Angle						LEA					TEA				Trace Length			
	U	N	D	E	EW	NE	DP	None	0-19	20-59	60-74	75-84	85-90	1	2	3	W	H	1	2	3	W	10-29	30-69	70-129	130-
1	1	1	1	1	1	1	1	0	1	1	1	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0
2	1	1	1	0	1	1	0	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0
3	0	1	0	1	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	1	0	0	1	0	0	0
4	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	1	0	0	0
5	0	1	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	1	1	0	0	1	0	0	0
6	0	1	0	1	0	0	0	1	1	1	1	1	1	0	1	0	0	0	0	1	0	0	0	1	1	0
7	0	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	0
8	0	1	0	1	0	0	0	1	1	1	1	1	1	0	0	1	1	0	1	1	0	0	0	1	1	0
9	0	1	0	0	1	0	0	0	1	1	0	0	0	1	1	1	0	0	0	0	1	1	0	1	1	0
10	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	0
11	0	1	0	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	0	1	0	0
12	0	1	0	0	1	0	0	0	0	0	1	1	1	0	0	1	1	0	1	1	0	0	0	1	0	0
13	0	1	0	0	1	1	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0	1	0
14	0	1	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	0	0	1	0
15	0	1	0	0	0	1	0	0	1	1	0	0	0	1	1	1	1	0	1	1	1	1	0	1	1	0
16	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	1
17	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	1	1	0	0	0	1
18	1	0	0	1	1	1	0	0	1	1	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1
19	1	0	0	0	1	0	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0	1
20	1	0	0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0	1
21	0	0	1	1	1	1	0	0	1	1	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	1
22	0	0	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0	0	1	1	0	1	1	1
23	0	0	1	0	1	1	0	0	0	0	1	1	1	0	1	0	0	0	0	0	0	1	0	1	1	1
24	0	1	1	0	1	1	0	0	0	0	1	1	1	0	0	1	1	0	0	0	1	1	0	1	1	1
25	0	0	1	0	1	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	0	0	0	0	0	1
26	0	0	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	1	1	0	0	0	0	0	1
27	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	1
28	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1

CONFIDENTIAL

TABLE 4. NON SUBMARINE MATRIX IN FORM USED IN COMPUTER

COMPUTER WORD NUMBER	BIT POSITION*										VARIABLE
	27	24	21	18	15	12	9	6	3	1	
1	11	010	000	000	000	000	111	000	000	00	(Up doppler)
2	11	101	111	111	111	111	000	000	100	11	(No doppler)
3	11	010	000	000	000	000	000	111	111	00	(Down doppler)
4	10	110	111	000	000	011	100	110	000	10	(Not elongated pip)
5	11	000	000	111	111	011	110	111	110	10	(Elongated pip)
6	11	001	000	000	011	111	101	111	101	01	(Elongated with wake pip)
7	10	000	000	000	000	011	000	000	000	00	(Double pip)
8	00	110	111	000	000	000	000	000	000	00	(No axis angle)
9	10	111	111	110	000	111	100	110	000	11	(0-19 axis angle)
10	11	110	111	110	000	111	100	110	000	01	(20-59 axis angle)
11	10	110	111	001	111	011	110	011	110	00	(60-74 axis angle)
12	10	110	111	001	111	011	111	011	111	00	(75-84 axis angle)
13	10	110	111	001	111	011	111	011	111	00	(85-90 axis angle)
14	01	011	010	101	011	111	011	010	011	00	(1 leading edge alignment)
15	01	111	110	101	011	111	011	101	011	00	(2 leading edge alignment)
16	10	000	011	110	110	110	100	100	111	00	(3 leading edge alignment)
17	00	000	001	010	110	110	100	000	111	00	(Wake on leading edge)
18	00	000	000	000	000	000	000	000	000	11	(Highlights)
19	01	011	001	010	110	110	111	000	011	11	(1 trailing edge alignment)
20	01	111	101	010	110	110	111	000	011	11	(2 trailing edge alignment)
21	10	000	010	101	001	101	111	010	100	11	(3 trailing edge alignment)

*Bit Positions 0 and 29 are not used; **the numbers are for the least significant of each set of three bits.**

TABLE 5. NELIAC DISTRIBUTION TABLE

	,	;	.	:	(	)	[	]	|	=	≠	>	<	→	+	-	X	/	⋕	⊥	⊕⊠	¢	◊
,	4	4	3	29	6		23		4	10	10	10	10	10	11	11	13	17	4	10			4
;	4	4	3	29	6		23		4	10	10	10	10	10	11	11	13	17	4	10			4
.	4	4	3	29	6		23		4	10	10	10	10	10	11	11	13	17	4	10			4
:	4	4	3	5	6		23		4	10	10	10	10	10	11	11	13	17	4	10	*10		4
(					6	20	23							20	7	7	13	17					
)				9		8	23			9	9	9	9	9	8	8	14	18			9	21	
[								25							24	24							
=				1	1		23							1	1	1	1	1			1		
≠				1	1		23							1	1	1	1	1			1		
>				1	1		23							1	1	1	1	1			1		
<				1	1		23						*1	1	1	1	1	1			1		
→	19	19	19	19	26	19	23		19	19	19	19	19	19	19	19	19	19			19		19
+				*10	6	*10	23	25		10	10	10	10	10	11	11	13	17		10	*10	21	
-				12	6	12	23	25		12	12	12	12	12	12	12	13	17			12	21	
X				15	6	15	23			15	15	15	15	15	15	15	15	15			15		
/				16	6	16	23			16	16	16	16	16	16	16	16	16			16		
⋕	*29	2	2																				
⊕⊠					6		23			10	10	10	10	10	11	11	13	17		10			
◊	22	22	22																				

4

(COMMENTS: THE AMPLIFYING WORD FROM THE KEYSET WILL CONTAIN THE DOPPLER ENTRY IN BITS 27→29, PIP SHAPE IN 24→26, AXIS ANGLE IN 21→23, LEADING EDGE ALIGNMENT IN 18→20, TRAILING EDGE ALIGNMENT IN 15→17, AND TRACE LENGTH IN 12→14. ENTRIES IN THE SONAR CLASSIFICATION WORD HAVE THE FOLLOWING MEANINGS: 0 = UNKNOWN, 1= ILLEGAL, 2 = NONSUB, 3 = SUB, 7 = INSUFFICIENT DATA. THE DIMENSIONING STATEMENT IS SET UP SO THAT A COMPACT TABLE WILL RESULT. THEREFORE, THE LAST VALUE FOR A PARTICULAR VARIABLE, LIKE SUB DOPPLER, IS LISTED IN THE DIMENSIONING STATEMENT AS THOUGH IT WERE THE FIRST VALUE OF THE SUCCEDING VARIABLE. THE ORDER OF WORDS IN THE DIMENSIONING STATEMENT IS: DOPPLER, NONE, UP, DOWN; PIP SHAPE, NOT ELONGATED, ELONGATED, ELONGATED WITH WAKE, DOUBLE PIP; AXIS ANGLE, IN ORDER OF SIZE; LEADING EDGE ALIGNMENT, 1, 2, 3, W; HIGHLIGHTS; TRAILING EDGE ALIGNMENT, 1, 2, 3, W; AND TRACE LENGTH, IN ORDER OF SIZE. THE KEYSET ENTERS NUMBERS FOR EACH VARIABLE WHEN AN ENTRY IS MADE, EXCEPT THAT HIGHLIGHTS ARE HANDLED AS THE LAST VALUE OF LEADING EDGE ALIGNMENT. THE SIZE OF THE NUMBERS ENTERED CORRESPONDS TO THE ORDER OF THE VALUES IN THE TABLE IN COMPUTER STORAGE.)

SUB DOPPLER (3) = , ⊦ 37770 00015, ⊦ 00007 70156,
SUB PIP (4) = ⊦ 00006 07756, ⊦ 31411 06100,
⊦ 04317 47110, ⊦ 00067 36646, SUB AXIS ANGLE (6) = ⊦ 02000 00001,
⊦ 31400 00000, ⊦ 35613 06140, ⊦ 61601 06012, ⊦ 61564 61414, ⊦ 31564 51214,
SUB LEADING EDGE ALIGNMENT (5) = ⊦ 33564 51215, ⊦ 27747 75600,
⊦ 17347 77600, ⊦ 00020 07600, ⊦ 00020 07600, SUB TRAILING EDGE
ALIGNMENT (4) = ⊦ 00010 00157, ⊦ 37377 75740, ⊦ 26777 75740, ⊦ 00051 72140,
SUB TRACE LENGTH (5) = ⊦ 00051 70140, ⊦ 36006 00001, ⊦ 01771 77756,
⊦ 01611 77756, ⊦ 00001 26542, NONSUB DOPPLER (3) = , ⊦ 35777 70046,
⊦ 32000 07000, NONSUB PIP (4) = ⊦ 32000 00770, ⊦ 26700 34604,
⊦ 30077 36764, ⊦ 31003 75752, NONSUB AXIS ANGLE (6) = ⊦ 20000 30000,
⊦ 06700 00000, ⊦ 27760 74606, ⊦ 36760 74602, ⊦ 26717 36360,
⊦ 26717 37370, NONSUB LEADING EDGE ALIGNMENT (5) = ⊦ 26717 37370,
⊦ 13253 73230, ⊦ 17653 73530, ⊦ 20366 64470, ⊦ 00126 64070,
NONSUB TRAILING EDGE ALIGNMENT (4) = ⊦ 00000 00006,
⊦ 13126 67036, ⊦ 17526 67036, ⊦ 20251 57246,
NONSUB TRACE LENGTH (5) = ⊦ 00251 57746, ⊦ 37000 00000, ⊦ 00774 44742,
⊦ 00763 44742, ⊦ 0000 37776, LOGICAL PRODUCT;

NEL CLASSIFY:

|SAVE INDEX REGISTER VALUES, AMPLIFYING WORD [j] → AMPLIFYING WORD, SET UP INDICES, CHECK ENTRIES FOR LOGIC, CHECK SUBMARINE WORDS, CHECK NON SUBMARINE WORDS, EXIT CLASSIFICATION:,|,

SAVE INDEX REGISTER VALUES:

|i → SAVE I, j → SAVE J, k → SAVE K, l → SAVE L, m → SAVE M, n → SAVE N,|,

SET UP INDICES:

|AMPLIFYING WORD (27→29) → i, AMPLIFYING WORD (24→26) → j, AMPLIFYING WORD (21→23) → k, AMPLIFYING WORD (18→20) → l, AMPLIFYING WORD (15→17) → m, AMPLIFYING WORD (12→14) → n,|,

CHECK ENTRIES FOR LOGIC:

| IF l = 5: | IF m = 0: | IF i = 0 ⊕ j = 0 ⊕ k = 0 ⊕ n = 0: INSUFFICIENT DATA. IF NOT, ;| ; IF NOT, ILLEGAL.| ; IF NOT, | IF i = 0 ⊕ j = 0 ⊕ k = 0 ⊕ l = 0 ⊕ m = 0 ⊕ n = 0: INSUFFICIENT DATA. IF NOT,;|; IF l = 4 ⊠ m = 4: ILLEGAL. IF NOT,; IF j = 1 ⊠ k ≠ 1: ILLEGAL. IF NOT,; IF j = 4 ⊠ k ≠ 6: ILLEGAL. IF NOT, ; |,

CHECK SUBMARINE WORDS:

| ⊥ 10031 SUB DOPPLER, ⊥ 40032 SUB PIP, ⊥ 03000 30, ⊥ 40033 SUB AXIS ANGLE, ⊥ 03000 30, ⊥ 40034 SUB LEADING EDGE ALIGNMENT, ⊥ 03000 30, ⊥ 40035 SUB TRAILING EDGE ALIGNMENT, ⊥ 03000 30, ⊥ 40036 SUB TRACE LENGTH, ⊥ 15030 LOGICAL PRODUCT, IF LOGICAL PRODUCT ≠ 0: | IF 16 < LOGICAL PRODUCT: SUBMARINE. IF NOT, ILLEGAL.|, IF NOT, ; |,

CHECK NON SUBMARINE WORDS:

| | 10031 NONSUB DOPPLER, | 40032 NONSUB PIP, | 03000 30, | 40033 NONSUB AXIS ANGLE, | 03000 30, | 40034 NONSUB LEADING EDGE ALIGNMENT, | 03000 30, | 40035 NONSUB TRAILING EDGE ALIGNMENT, | 03000 30, | 40036 NONSUB TRACE LENGTH, | 15030 LOGICAL PRODUCT, IF LOGICAL PRODUCT ≠ 0: NON SUBMARINE. IF NOT, UNKNOWN. |,

ILLEGAL:

RESTORE INDEX REGISTER VALUES, 1 → CLASSIFICATION [j], EXIT CLASSIFICATION.

INSUFFICIENT DATA:

RESTORE INDEX REGISTER VALUES, 7 → CLASSIFICATION [j], EXIT CLASSIFICATION.

SUBMARINE:

RESTORE INDEX REGISTER VALUES, 3 → CLASSIFICATION [j], EXIT CLASSIFICATION.

NON SUBMARINE:

RESTORE INDEX REGISTER VALUES, 2 → CLASSIFICATION [j], EXIT CLASSIFICATION.

UNKNOWN:

RESTORE INDEX REGISTER VALUES, 0 → CLASSIFICATION [j], EXIT CLASSIFICATION.

RESTORE INDEX REGISTER VALUES:

|SAVE I → i, SAVE J → j, SAVE K → k, SAVE L → l, SAVE M → m, SAVE N → n,|..