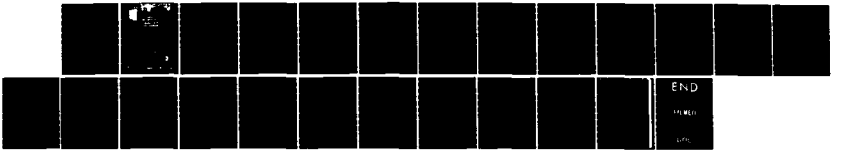
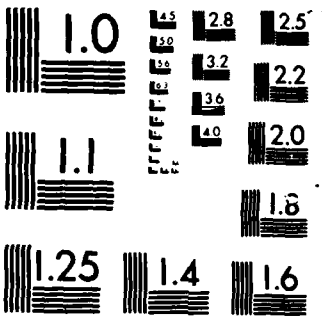
AD-A163 745 MODE S DATA LINK PHASE CODING(U) ROYAL SIGNALS AND 1/1
RADAR ESTABLISHMENT MALVERN (ENGLAND)
P T HUMPHREY ET AL. JUL 85 RSRE-MEMO-3859 DRIC-BR-98168
UNCLASSIFIED F/G 17/2 NL





MEMORANDUM No. 3859

MODE S DATA LINK PHRASE CODING

Authors: P T Humphrey and R J Larkin

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3859

Title: MODE S DATA LINK PHRASE CODING

Authors: P T Humphrey, AD4 Division, RSRE
R J Larkin, Gela Software Ltd.

Date: July 1985

SUMMARY

This Memo suggests improvements to a proposed method of phrase coding for the Mode S Data Link and describes some testing of the phrase language.

RSRE Task: 165

CAA R and D PROGRAMME ITEM: 2.2.1.1
(RELATED ITEM: 2.5.1.2)

CAA SPONSOR: Chief Scientist

CAA LIAISON OFFICERS: Dr J H Crowther, RD1
Mr P A Platt, Tels C3

RSRE MEMORANDUM 3859

MODE S DATA LINK PHRASE CODING

P T HUMPHREY, R J LARKIN

LIST OF CONTENTS

TABLES

FIGURES

1 INTRODUCTION.

The new form of the selective address secondary surveillance radar (SSR), Mode-S, has the capability of passing data both up and down between the ground and individual aircraft. In each interrogation of an aircraft and in its reply, 56 bits of data can be passed and several interrogations or replies can be associated to pass longer messages. With this limited data transfer capacity an efficient method of coding is required.

Mode-S is at present passing through the process of definition and approval by the International Civil Aviation Organisation in an expert Panel, the SSR Improvements and Collision Avoidance Systems Panel (SICASP). The coding scheme chosen for passing ATC tactical and strategic messages will use word and phrase dictionaries which will be defined in ICAO Standards and Recommended Practices (SARPS) and be useable worldwide. With many airlines, ATC authorities and manufacturers of equipment involved, an underlying concern in the proceedings of SICASP has been the need to devise a system which is certifiable and which can be implemented in a range of avionic designs.

SARPS define the signals-in space but avoid defining the computer high level language or the hardware in order to permit technical growth and development. The intention is to allow a manufacturer of avionics to continue to design his equipment as he prefers to do, to use whatever high level language he chooses and yet ensure that the dictionaries and any coding/decoding logic is implemented in a uniform way with a defined response time.

This paper seeks to add improvements to the method of phrase coding suggested in Ref.2 which has arisen out of the experience of building an experimental Mode-S data link at RSRE. The comments and the additions which are recommended are based on the current stage of Mode-S definition which is represented by Ref.1.

2 THE DICTIONARY.

The messages which may be passed on an ATC data link may include alpha-numeric text and numerical data. Each message is called a phrase and is defined in a dictionary and referred to by a dictionary entry number. There may also be a number of dictionaries which may refer to different applications or to areas of the world, each referred to by a dictionary number.

The dictionaries of phrases will be approved by ICAO so that they may be universally applicable in any ATC region which chooses to incorporate a data link for ATC purposes. This

means that each phrase will go through a process of approval by ICAO and implementation by a manufacturer. The implementation must be error free and so it is proposed that the paper definition which is approved should also be the coding definition for the computer programmer. The code which is produced should be in a form which can be implemented in any computer in a standard way.

The paper definition must be easily readable and unambiguous but understandable by a computer. In Section 3, the programming language which will permit this is described together with its formal specification. It is suggested that this renders the phrases readily understandable, more so than a language like BASIC, so that each phrase can be discussed by non-computer specialists.

A list of phrases for experimental use has been drawn up and is shown in Table 1 in plain English. It has arisen out of studies at RSRE of computer assistance for the air traffic controller who has to manage arrival traffic and out of other studies of the use of a data link for passing aircraft and meteorological parameters. The list is not meant to be complete.

3 THE PHRASE PROGRAMMING LANGUAGE.

3.1 DESIGN APPROACHES

As outlined in Ref.2 there are at least three approaches to the design of a phrase language processor:

a) conventional compiler

b) conventional interpreter

c) hybrid translator/interpreter

Approach a) may be dismissed on two counts. Firstly, each individual phrase would be compiled into an object routine that would be stored in a phrase program dictionary and executed as a subroutine. This would inevitably lead to very large programs. Secondly, The program logic to process the phrases would not be resident in the target processor. This would obviously increase the problems of control and certification.

Approach c) was the design chosen in the referenced paper, this was an obvious choice for that particular application bearing in mind the outlined advantages. These are that the source language could be parsed off-line; the pseudo-code (p-code) instructions could be designed for maximum efficiency when accessed by the interpreter in the target machine; and the

branch destinations could be pre-computed. Finally the translator is obviously portable since it only relies on a p-code interpreter for each particular target machine.

Approach b) was chosen for the design of the language processor described in this paper. The design is similar to approach c) as far as interpreting the source phrase code within the target processor. The source phrase language is defined with 5 operators that each take one or two parameters plus the 'END()' operator. This keeps the language consistent and it is therefore easy to interpret at run time. The criticisms, by the referenced paper, on this approach are duely noted but because of the compact nature of the source language the parsing, syntax checking and branching does not require 'excessive execution time', certainly for experimental purposes. The other details for this approach are outlined below:

i).A single source language can be used to implement the chosen design. In this case CORAL 66 was used because of the authors familiarity with the language and compatibility with other programs.

ii) It was important to complete the project with a working demonstration of the language processor as quickly as possible.

iii) The speed of interpreting the source phrase language was not of major importance since the completed project will be a ground based research demonstration of a data link,including the coding rules, on a VAX 11/780.

iv) A future development could introduce the intermediate step of p-code to produce a translator/interprter hybrid system quite easily.

3.2 PHRASE PROGRAM SOURCE LANGUAGE

This section will highlight some improvements to the Phrase Program Source Language defined in Ref.2 and then the improved version will be described concluding with a formal definition of it.

3.2.1 Suggested Improvements - The referenced paper seems to assume that a phrase program should be presented in the form of a BASIC type language. It is suggested that it would be preferable to define the Source Language, as much as possible, in plain English in order to improve its legibility. To

accomplish this the language structure should be based on a high level language.

To illustrate the possible improvements the example phrase program source file which is quoted in Ref.2 will be studied.

```
10    FIX 150                                ;CLEARED TO ... .
20    VAR (1-6) POSBIN
30    CTL (7) POSBIN
      ON (1) GOTO 40
      ON (0) GOTO 60
40    FIX 160                                ;LEFT
50    GOTO 70
60    FIX 170                                ;RIGHT
70    FIX 180                                ;WIND
80    VAR (8-16) POSBIN
90    FIX 190
100   VAR (17-23) POSBIN
110   CTL (24) POSBIN
      ON (1) GOTO 120
      ON (0) GOTO 140
120   FIX 200                                ;GUSTING
130   VAR (25-31) POSBIN
140   END
150   DATA " CLEARED TO START ENGINES EXPECT RUNWAY "
160   DATA " LEFT "
170   DATA " RIGHT "
180   DATA " WIND "
190   DATA " / "
200   DATA " GUSTING "
```

Comments on this are as follows.

i) The FIX() operator references a line number to access the ASCII string. An obvious improvement would be to define the string as part of the operator, substituting this for the line number. This would reduce program searching and improve phrase legibility. Another associated improvement is that comments become redundant since they were used to show the string being stored at the line number being referenced.

ii) The phrase program example makes extensive use of the GOTO statement. It is considered 'bad practice' to use this statement in programing and so it would be better to remove it from the Source Program Language. This may be accomplished by introducing the high level language 'case statement'. This case statement could also cover the ELSE structure thus also making the ELSE statement redundant.

iii) The improvements in i) and ii) would mean that line numbers become unnecessary so that the phrase programs may be written in free format.

Another set of improvements to the phrase program may be made to the consistency and compactness of the source language. This improves not only their legibility but also the ease with which the phrases may be processed.

i) It appears that unnecessarily long, 5 and 6 character, words are used to define simple data types, ie POSBIN, SIGNED, ALPHA and ASCII. It would be better to use single character identifiers,

ie. U = Unsigned, S = Signed, A = Alpha, @ = ASCII.

This improves readability and it reduces phrase processing time considerably.

ii) The bit numbering of the VDF (Variable Data Field) and CDF (Control Data Field) are defined as

(<firstbit> - <lastbit>).

However, since a message is always processed sequentially the position of the field becomes redundant if the field size is defined instead and a running total of the start-bit position is kept.

iii) Each of the operators VAR(), CNT() and ON() can accept either 2 or 3 parameters depending on whether a single value or range of values is being specified. Also most of the ten operators take different types of parameter. It would be more convenient if each operator was to take the same number and type of parameter. The language could be defined so that the only parameters to be accepted by an operator would be (<value><datatype>).This would improve legability as well as facilitating with the process of interpretation.

3.2.2 Improved Language Definition - The definition of the improved Phrase Program Source Language contains all of the above recomendations and may be completely defined with the following 6 operators:

FIX()
VAR()
CNT()
ON()
TO()
END()

The two operators that process VDFs or CDFs, VAR() and CNT(), take the same two parameters defining the size and data type of the variable.The two case statement index operators, ON() and TO(), take the index and its data type as their parameters. Of the remaining two operators, FIX() accepts the ASCII string and the END() operator has no parameters.

The improved Phrase Program Source Language is demonstrated with the phrase example used above.

```
i) Phrase format:
      cleared to start engines
        expect runway - RIGHT wind xxx/yy gusting zz
                      - LEFT       xxx/yy          zz

ii) Phrase example:
      cleared to start engines
        expect runway  22 right wind 215/20 gusting 30

iii) Dictionary entry:
      FIX("CLEARED TO START ENGINES EXPECT RUNWAY ")
      VAR(6U)
      CNT(1U)
              ON(0U)FIX(" LEFT ")
              ON(1U)FIX(" RIGHT ")
        FIX(" WIND ")
      VAR(8U)
      FIX("/")
      VAR(6U)
        CNT(1U)
              ON(1U)FIX(" GUSTING ")
              ON(1U)VAR(6U)
              END()
```

Another likely ATC phrase is shown below :

```
  i) Phrase format:
      xx Nmiles to BCN change SPD to    yyy KNOTS
      xx            BCN        HDG to    yyy DEGREES
      xx            BCN        HT to FL yyy

 ii) Phrase example:
      20 Nmiles to LAM change HDG to 220 degrees

iii) Dictionary entry:
      VAR(7U)
      FIX(" Nmiles to ")
      VAR(18A)
      FIX(" change ")
      CNT(3U)
              ON(0U)FIX(" SPD to")
              ON(1U)FIX(" HDG to")
              ON(2U)FIX(" HT to FL ")
```

```
VAR(9U)

        ON(0U)FIX(" knots ")
        ON(1U)FIX(" degrees ")
        ON(2U)TO(7U)FIX(" error")
END()
```

In this phrase there are two instances of the case statement using the same CDF. By using the rule that ON() operators must compare the last value assigned to the CDF, duplication of the CNT() operator may be avoided.

The source language case statement structure may be described with reference to the example above. The case index, ie. the 3 bit CDF value returned by the CNT() operator, is compared with the first ON() operator argument, ie. value 0. If this condition is satisfied then the operator which follows, ie. FIX("SPD TO"), is interpreted. Otherwise the next ON() operator argument is tested, ie. value 1. This process continues until either a condition is met or there are no more ON() operators. When a TO() operator is located after the ON() operator the case index is compared to the inclusive range between the arguments, ie. value 2 to 7. If this condition is satisfied, the operator following is interpreted, ie. FIX("error").

A limitation of the present language definition is the fact that the case statement only allows a single operator to follow the ON() operator. This structure has been sufficient for all ATC phrases considered thus far. To overcome this limitation a case statement delimiting operator could defined.

The language definition ignores scaling operations and fractional digits, although these could be included at a later date.

3.2.3 Formal Definition - The formal definition of the language is now given. The syntax of the "meta - language" is shown below:

```
::=  means "is replaced by"
[ ]  enclose an optional field
< >  enclose programmer-specified fields
( )  are to be included in statements as shown
```

1. FIX :

```
FIX ("<ASCII_string>")
```

2. VAR :

```
VAR (<number_bits><data_type>)
```

```
<data_type> ::= U/S/A/@
                [unsigned_integer,signed_integer,alhpa,ASCII]


3. CNT :
      CNT (<number_bits><data_type>)
<data_type> ::= U/S/A/@
                [unsigned_integer,signed_integer,alhpa,ASCII]


4. ON :
      ON (<[lower]_limit><data_type>)

where:
<limit> ::= [ unsigned_integer/
                signed_integer/
                alpha_character/alpha_characters
                ASCII_character/ASCII_characters ]
<data_type> ::= U/S/A/@
                [unsigned_integer,signed_integer,alhpa,ASCII]


5. TO :
      TO (<[upper]_limit><data_type>)
where:
<limit> ::= [ unsigned_integer/
                signed_integer/
                alpha_character/alpha_characters
                ASCII_character/ASCII_characters ]
<data_type> ::= U/S/A/@
                [unsigned_integer,signed_integer,alhpa,ASCII]



      6. END:
         END()


Ranges:
   unsigned_integer ::=    0 --> 1023
   signed_integer   ::= -511 --> 512
   alpha_character  ::=    A --> Z
   ASCII_character  ::=  ETX --> ?
```

4 EXPERIMENTAL IMPLEMENTATION

4.1 The Language Processor Design

In the full implementation it is envisaged that a dictionary structure similar to that described in Ref.2. will be incorporated. However, this phase of the project was completed using the simple structure of a 1-dimensional array to simulate the dictionary. Also, in this first stage, only a single dictionary was used and so the ADS field was not included neither was the mechanism for linking phrases.

4.1.1 The Decoder. - The main part of the language processor is the DECODER section. This receives the uplink message, processes the variable message fields, in association with the relevant dictionary entry to produce the original phrase and display it on the console.

The dictionary pointer is set by the first field of the message as it is read in from the internal buffer.

The DECODER routine is coded as a continuous loop which stops when either the END() operator is located or an error occurs. The next function code is fetched from the dictionary by calling the NEXT FUNCTION CODE routine. This function code selects the operator section of code to pass control to. The operator parameters are read from the dictionary and interpreted to give the following;

String address	from	FIX()
Number of bits, data type	from	VAR(), CNT()
Index, data type	from	ON(), TO()

The chosen operator routine (ie. FIX, VAR, CNT, ON) is called and processes the information in the following ways;

* The FIX routine accepts the string address as its parameter and displays the passed string on the console.
* The VAR routine extracts the message VDF and converts it from its coded form (Unsigned, Signed, Alpha, ASCII) into its equivalent string and displays it by calling the FIX routine.
* The CNT routine extracts the value from the message CDF and sets the case index to this value.
* The ON routine parameter is used as a condition test against the case index. If the condition is satisfied then the operator following, either the ON() or TO() operator, is interpreted otherwise it is ignored. This ON() routine

compares the case index with the range of values between a lower and an upper limit. When a single ON() operator is interpreted both the upper and lower limits are set to the ON() argument. If a TO() operator is following the ON() operator, then the lower limit is set by the ON() argument and the upper limit is set by the TO() argument.

4.1.2 The Method Of Testing. - It was felt that to test the language processor thoroughly it would be necessary to construct many message test cases. If this were to be accomplished by presetting individual bits in the form of hex values, (cf Ref.2) this would become a laborious and error prone task. Also, it would be difficult to check as well as being awkward to demonstrate.

It was decided therefore to implement an ENCODER routine. This was based on the same design lines as the DECODER routine except the ENCODER solicits from the user the values to be set in the message fields.

For research purposes it is necessary to be able to modify the list of phrases in a convenient manner and so to accomplish this a phrase file was used to store the present set of phrases used for testing. Each phrase is laid out in free format with its associated phrase number at the beginning of the line, as shown in Table 2.

A CREATE DICTIONARY routine reads in the phrases, removes the layout characters and stores the remaining characters in a 1-D array. The phrase number maps to its associated dictionary pointer, pointing to the start of the phrase.

A further consideration was to allow the tester to view the phrase format while inputting field values. To enable this a call is made to the DISPLAY PHRASE routine with the phrase number as its argument. Finally the completed message is displayed in binary and octal format for reference purposes. Refer to the hard copy example of a testing session, Fig 1, which shows the phrase example as it was prepared, Fig 1a,and as it was displayed after decoding, Fig 1b.

This coding project was initiated as part of the datalink project. It seemed therefore appropriate to use the implementation of the datalink software to send the encoded message to be decoded. The system works in the following way;

Refer to the ACP diagram Fig 2.

i) Phrase file name selected.

ii) Dictionary created by CREATE DICTIONARY routine.

iii) Phrase selected by number.

iv) Phrase displayed to user by DISPLAY PHRASE routine.

v) Values for VDF and CDF solicited and coded by ENCODER routine.

vi) Completed message displayed.

vii) Message written to global buffer by PRODUCER routine.

viii) SYSTEM 1 transceiver transmits message via local PAD.

ix) SYSTEM 2 transceiver receives message via local PAD.

x) Message read from global buffer by CONSUMER routine.

xi) VDF and CDF message fields processed and decoded by DECODER routine.

xii) Phrase displayed on console.

The PAD is an X25 Packet Assembler/Disassembler connection to a communications line.

The ENCODER routine becomes a constituent part of the PREpare MESSage sub-process that uses the PRODUCER routine to fill the transmit buffer for SYSTEM 1. Meanwhile, the DECODER routine becomes a constituent part of the DISplay MESSage sub-process that uses the CONSUMER routine to empty the receive buffer for SYSTEM 2.

An identical dictionary had to be produced for both the ENCODER and DECODER routines to access. This was done by using the same phrase file as input for the CREATE DICTIONARY routine in both the PREMESS and DISMESS sub-processes.

In the real world SYSTEM 1 represents the ground-based ATC system, while SYSTEM 2 represents the pilot data link interface. The PADs would connect to the communications system of the air-ground data link.

4.2 TESTING.

In the referenced paper there is a particular emphasis on the compact size of the language processor and its fast execution

times. Obviously in this implementation any comparisons would be meaningless.

All phrases in the example phrase file, Table 2, have been tested with various appropriate values including the limiting cases. In all cases the system has reproduced the original phrase, typed in at the 'ATC console', and displayed it successfuly on the 'pilot console'. One present minor limitation of the encoding routine is that it does not carry out a comparison on the size of the value entered with the number of message field bits available. It sets only the number of bits defined and ignores those outside.

5 CONCLUSION

Some study and the practical experience of implementing a phrase program has suggested some improvements. These should enable the proposed method to be applied in as efficient a manner as possible.

The work reported here has been part of a larger project to investigate the system problems of operating an ATC data link. This will cover both operational and engineering aspects and so it was considered that it was right to make the implementation of the message coding as close as possible to the likely ICAO specification.

REFERENCES

1.Draft US National Aviation Standard for Data Link Applications of the Mode Select Beacon System. January 1985.

2. Heath W.S. Tactical Data Link Message Coding Validation. MIT Lincoln Laboratory ATC Project Memorandum No.42PM-Data Link-0004. 4 December 1984.

1. Turn xxx degrees
2. Expedite turn xxx degrees
3. Turn right xxx degrees
4. Expedite turn right xxx degrees
5. Turn left xxx degrees
6. Expedite turn left xxx degrees
7. Reduce speed to xxx knots
8. Increase speed to xxx knots
9. Maintain speed at xxx knots
10. No speed restriction
11. Report your flight level xx miles before BCN
12. Next report at BCN
13. Descend to FLxxx
14. Expedite descent to FLxxx
15. Climb to FLxxx
16. Expedite climb to FLxxx
17. Resume own navigation at BCN
18. Hold on BCN
19. Release hold on BCN
20. xxx miles to BCN change speed to vvy knots
    heading to yyy degrees
    height to FLyyy
21. xxx miles after BCN change speed to yyy knots
    heading to yyy degrees
    height to FLyyy
22. Release from hold at time xx Hrs yy Mins zz Secs
23. Release from hold immediately
24. Change to RT frequency xxx.yy MHz
25. xxx miles from BCN cleared down to FLvvy
26. xxx miles from BCN cleared down to FLvyy at
    time pp Mins qq Secs
27. Standard GMT is xx Hours yy Mins zz Secs

Table 1. Experimental Phrase List

```
1     FIX("turn ")VAR(9U)FIX(" degrees")
      END()
2     FIX("expidite turn ")VAR(9U)FIX(" degrees")
      END()
3     FIX("turn right ")VAR(9U)FIX(" degrees")
      END()
4     FIX("expedite turn right ")VAR(9U)FIX(" degrees")
      END()
5     FIX("turn left ")VAR(9U)FIX(" degrees")
      END()
6     FIX("expedite turn left ")VAR(9U)FIX(" degrees")
      END()
7     FIX("reduce speed to ")VAR(10U)FIX(" knots")
      END()
8     FIX("increase speed to ")VAR(10U)FIX(" knots")
      END()
9     FIX("maintain speed at ")VAR(10U)FIX(" knots")
      END()
10    FIX("no ATC speed restriction")
      END()
11    FIX("report your level ")VAR(9U)FIX(" miles before ")VAR(18A)
      END()
12    FIX("next report at ")VAR(18A)
      END()
13    FIX("descend to FL")VAR(9U)
      END()
14    FIX("expedite descend to FL")VAR(9U)
      END()
15    FIX("climb to FL")VAR(9U)
      END()
16    FIX("expedite climb to FL")VAR(9U)
      END()
17    FIX("resume own navigation at ")VAR(18A)
      END()
18    FIX("hold on ")VAR(18A)
      END()
19    FIX("release hold on ")VAR(18A)
      END()
20    VAR(7U)FIX(" miles to ")VAR(18A)FIX(" change ")CNT(3U)
                                          ON (0U)FIX("SPD to ")
                                          ON (1U)FIX("HDG to ")
                                          ON (2U)FIX("HT to FL")
                                                VAR(9U)
                                          ON (0U)FIX(" knots")
                                          ON (1U)FIX(" degrees")
                                          ON (3U)TO(7U)FIX(" ERROR")
      END()
21    VAR(7U)FIX(" miles after ")VAR(18A)FIX(" change")CNT(3U)
                                          ON (0U)FIX(" SPD ")
                                          ON (1U)FIX(" HDG ")
                                          ON (2U)FIX(" HT  FL")
                                                VAR(9U)
                                          ON (0U)FIX(" knots")
                                          ON (1U)FIX(" degrees")
                                          ON (3U)TO(7U)FIX(" ERROR")
```

Table 2 Experimental Dictionary File

```
        END()
22      FIX("Release from hold at time: ")
        VAR(4U)FIX(" Hrs ")VAR(6U)FIX(" Mins ")VAR(6U)FIX(" secs ")
        END()
23      FIX("Release from hold immediatly")
        END()
24      FIX("change RT frequency ~ ")VAR(8U)FIX(".")VAR(11U)FIX(" MHz")
        END()
25      VAR(7U)FIX(" miles from ")VAR(18A)FIX(" cleared down to  FL")VAR(9U)
        END()
26      VAR(7U)FIX(" miles from ")VAR(18A)FIX(" cleared down to  FL")VAR(9U)
        FIX(" at time: ")VAR(6U)FIX(" Mins ")VAR(6U)FIX(" secs ")
        END()
27      FIX("standard GMT is : ")VAR(4U)FIX(" Hrs ")VAR(6U)FIX(" Mins ")VAR(6U)FIX(" secs ")
        END()
```

Table 2 (cont)

PREPARE PHRASE process

```
enter name of phrase file
RTFPHRASE
Enter phrase number
20
        VAR(7U)FIX(" miles to ")VAR(18A)FIX(" change ")CNT(3U)
                                                    ON (0U)FIX("SPD to ")
                                                    ON (1U)FIX("HDG to ")
                                                    ON (2U)FIX("HT to FL")
                                                          VAR(9U)
                                                    ON (0U)FIX(" knots")
                                                    ON (1U)FIX(" degrees")
                                                    ON (3U)TO(7U)FIX(" ERROR")
        END()

23
 miles to LAM
 change 0
SPD to 250
 knots

                                    word 1          word 2          word 3          word 4
                                 (ADS & dict ptr)
Comm A message in binary format: 0000001100110111 0010011000010111 1010000001101000 0000000000001111
Comm A message in octal format:       001467          023027          120150          000017

Phrase :                         23 miles to LAM change SPD to 250 knots
```

(Entries by operator are underlined)

Fig 1a

DISPLAY PHRASE process

enter name of phrase file
RTFPHRASE

23 miles to LAM change SPD to 250 knots

Fig 1b

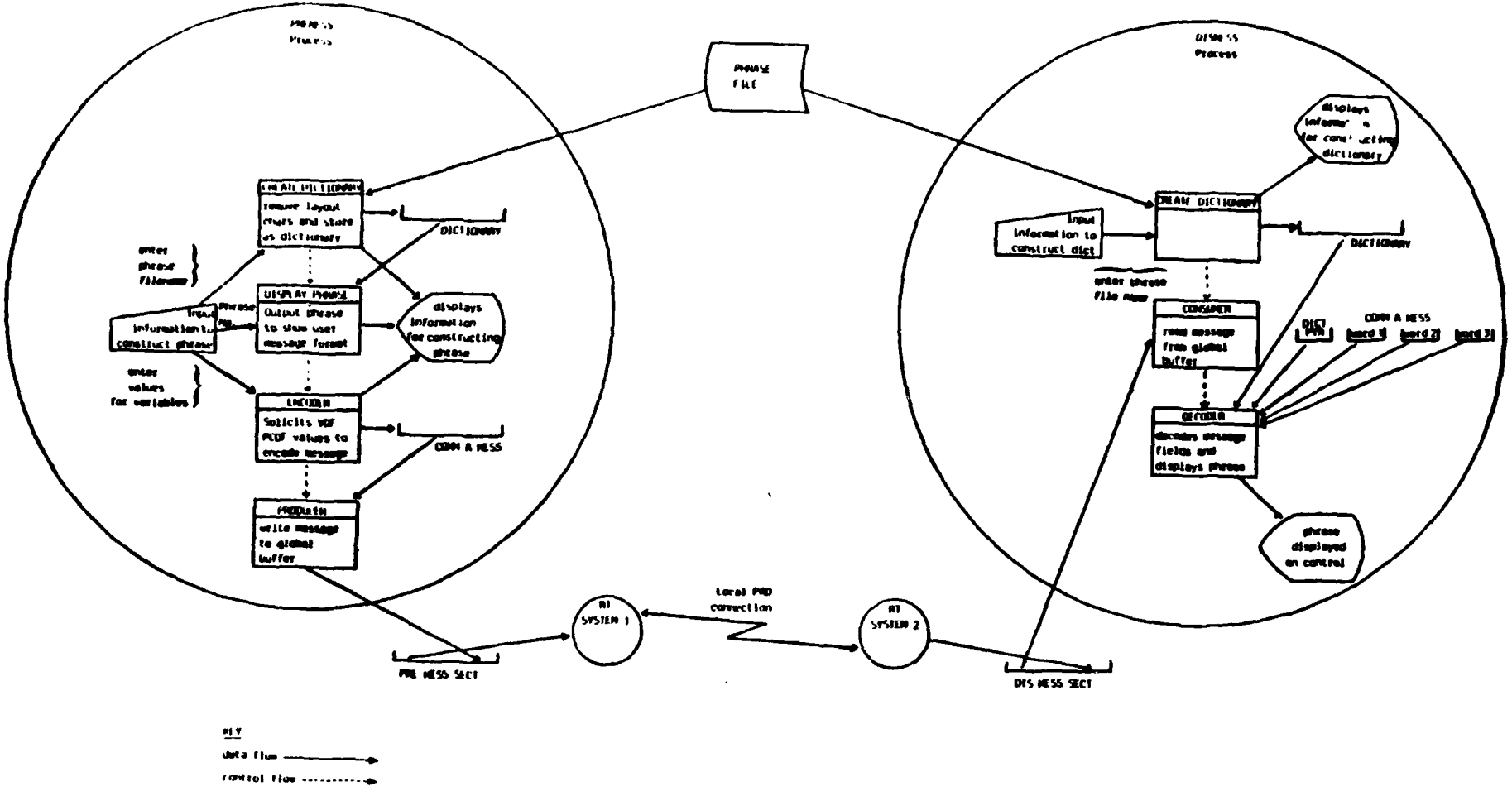


Fig 2 ACP Diagram

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	MEMO 3859		UNCLAS

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
TASK NO 165	ROYAL SIGNALS AND RADAR ESTABLISHMENT, MALVERN

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location
2.2.2.1 (2.5.1.2)	Chief Scientist/CAA-RD1

7. Title
Mode S Data Link Phrase Coding

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
HUMPHREY P T	LARKIN R J		July 85	

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement
UNLIMITED

Descriptors (or keywords)
Mode S
SSR
ATC Data Link
Phrase Coding

continue on separate piece of paper

Abstract

This Memo suggests improvements to a proposed method of phrase coding for the Mode S data link and describes the testing of the phrase language.

SD/48

END

FILMED

3-86

DTIC