TECHNICAL MEMORANDUM

20

Programming systems, documentation, Olympus, Doctor

J.K. Gibson

Operations Department

August 1980

European Centre for Medium-Range Weather Forecasts
Europäisches Zentrum für mittelfristige Wettervorhersage
Centre européen pour les prévisions météorologiques à moyen

CONTENTS

Programming Systems

1) Introduction
2) The Olympus System
3) The Doctor System
4) Documentation Extraction Programmes
5) Conclusion

Appendix 1 - The Olympus Programming System

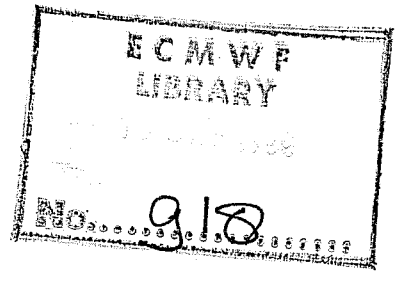
Appendix 2 - The Doctor System

Appendix 3 - Doctor System Utilities

Appendix 4 - DOC - a Documentation Extraction Programme

Appendix 5 - DOCK - CDC Documentation Extraction Programme ;

-.-.-.-.-.-



PROGRAMMING SYSTEMS

1. Introduction

From the beginning it was considered desirable to adopt a formal programming system as a programming standard for the Research Department of ECMWF. The Olympus system (K.V. Roberts - 1974) was chosen as a suitable standard and used as a basis for the Centre's first grid point forecast model, spectral forecast model, and data assimilation scheme programs. Development of the grid point forecast model resulted in some departure from the basic Olympus scheme, although many features are still retained. The purpose of this paper is to describe three programming systems - Olympus, Modified Olympus, and Doctor. Modified Olympus represents the programming system used in the current operational grid point forecast model. Doctor is a proposed system, based on Modified Olympus, which incorporates additional documentary orientated features extracted from the Control Data Corporation programming standard.

2. The Olympus Programming System

A description of the Olympus Programming System (J. Charlewood and D.M. Burridge, 1975) is reproduced as Appendix 1 of this paper. Readers not familiar with this paper should read Appendix 1 before continuing.

2.1 Problems Encountered with the Olympus System

In use, various minor problems were experienced associated with the basic Olympus system. Some of these problems were associated with the way in which the Olympus Library was set up at ECMWF, others were not so much problems, rather features of the system which were found not to be necessary due to the sophistication of the software available on the ECMWF computers.

2.1.1 Routine and Deck Names

Originally, the Olympus subprogram reference number (see table 3 of Appendix 1) was modified to the form CmSn to give routine deck names for the UPDATE library containing the Olympus source code, where m = subprogram class
n = subprogram number.

As the number of routines grew, cross-referencing the deck names with the routine names became a confusing and difficult process. Eventually, deck names were changed to agree with routine names.

2.1.2 Common Block Names

Again, the Olympus numerical structure for Common Blocks was discontinued in favour of using the character Common Block names, for similar reasons to those given above.

2.1.3 The Omit Facility

As a consequence of the abandonment of the numerical classification of subprograms, and also due to the complexity and interplay between routines, it was found desirable to discontinue the ability to omit routines by setting NLOMT switches. CDC UPDATE, FTN, and CFT facilities allow dummy routines to be selected with comparative ease, so the loss of this facility did not detract significantly from the usefulness of the system.

2.1.4 Dumping Facilities

Annotated dumps are available, in the event of failure, from both CYBER and CRAY-1 computers. In consequence, it was not considered worth while to maintain and update the Olympus routines to update blank common and arrays in named common blocks.

2.1.5 Documentation

Preparing and maintaining a documentation manual for the

grid point model proved to be a major task. Attempts to produce such documentation so far have failed to make use of the Olympus listable comment (CL) facility. Programs have subsequently been written to scan source code, print out routine names, common blocks called, listable (CL) and ordinary (C) comments. The resulting output is helpful, but not of a sufficient standard to be regarded as documentation. This is probably because the source code was written before the documentation processor. It is conceivable that programmers familiar with the documentation processor would write Olympus source code that would be almost self documenting.

2.2 Modified Olympus

The differences between the programming system used for the ECMWF grid point forecast model and the basic Olympus system are described in 2.1 above. This modified system has been given the name Modified Olympus.

3. THE DOCTOR SYSTEM

Although the concept of a completely self-documenting programming system is probably an impossible ideal for programs involving the solution to complex mathematical expressions, current experience with the grid point forecast model suggests that external documentation is difficult to maintain, and should be kept to a minimum. The CDC programming standard, together with the document extraction program DOCK, provides a ready-made means of obtaining structural documentation from source code provided certain rules are obeyed. DOCTOR (Documentary Orientated System) has been formulated to combine the features of Modified Olympus, which have proved useful in the past, with sufficient features from the CDC programming standard to provide a programming system which is almost self documenting. Lists of mathematical equations, etc., which are not easy to include in the source code can be produced externally as

Appendices to the "DOCK" produced documentation. As routines are structured in accordance with the Olympus principle of sections and sub-sections, such external documentation can be cross-referenced with the computer produced documentation and the source code. The three levels of documentation are available, but not all need be used if such structuring is considered to be too complex - "overview" and "external" categories could be combined into a single level.

Appendix 2 contains a full description of the Doctor system. Appendix 3 contains the source code of utility routines of Olympus type prepared for the Doctor system, and Appendix 4 contains a sample program to produce documentation from source code written using the Doctor system.

4. DOCUMENTATION EXTRACTION PROGRAMS

4.1 Introduction

Documentation extraction programs are programs which read a file containing source code, and extract such items as routine names, common blocks, comments, etc., to produce documentation which is then written to a print file. The quality of the documentation so produced is a function of

a) the quality of the documentation extraction program, and

b) the source code used as input to the extraction program.

At ECMWF, documentation extraction programs are available for use with Olympus/Modified Olympus, and Doctor. In addition, programs using the Doctor system may be processed using the CDC documentation extraction program DOCK.

4.2 Documentation Extraction Programs for Olympus

Two basic versions are under development and are available in a "pre-release" stage. Version 1 lists only routine

names and listable comments (i.e. comment lines beginning with CL in columns 1 and 2). Version 2 lists routine names, common blocks called, and all comments (i.e. all comment lines beginning with C). Additionally, common block cross reference information can be extracted using a programme which reads UPDATE source files as input.

4.3 Documentation Extraction Programs for Doctor

A FORTRAN written documentation extraction program is available for use with the Doctor system. It was written to enable the system to be used in conjunction with computers other than CDC, and is coded in easily adaptable FORTRAN. For users of ECMWF's Cyber 175 it is recommended that the CDC documentation extraction program DOCK be used, full details of which are listed in Appendix 5.

5. CONCLUSION

A formal programming system encourages the production of clearly laid out source code. The Olympus system in use has led to the production of source code which is well annotated, and, given sufficient external documentation, easy to comprehend and maintain. Difficulties have been experienced in maintaining the external documentation, whereas experiments with documentation extraction programs has shown that internal documentation is usually modified in line with changes to source statements.

The documentation so far produced using documentation extraction programs in conjunction with Olympus-type source code has not been sufficiently detailed to act as documentation in its own right. The omissions in such documentation are mainly due to a lack of sufficient detail in the listable comments - a situation which may well not have been so had the documentation extraction programs been available when the source code was written. The Doctor system has been developed to maximise the documentation

facilities available from source code. I personally would recommend its use for

a) new large projects;

b) new standard library routines (e.g. ECLIB, etc.);

c) new general purpose routines and packages.

For areas where Olympus and Modified Olympus have been used in the past, I would recommend that

a) new routines be made as near self documenting as possible using the documentation extraction programmes;

b) existing routines be made as near self documenting as possible when major modifications are necessary.

Appendix 1

The Olympus System

The Olympus Programming System

J. Charlewood and D.M. Burridge

1. INTRODUCTION

The Olympus programming system (1) which has been designed and implemented by the Computational Physics Group at the UKAEA, Culham, was originally developed for initial-value problems in physics. Their approach has been to design a common programming strategy for dealing with these problems. However, they claim that they have found this approach equally useful with some minor modifications for a much larger class of computing work, in particular equilibrium and stability calculations used in controlled thermonuclear research. (These problems are similar to the conventional diagnostic initialisation schemes used by meteorologists). At present most of the Olympus programs are written in (ANSI) Standard Fortran, though the basic ideas could readily be adapted for PL/1, Algol 60, Algol 68 or Assembler code. Some coding is inevitably machine dependent (for example accessing the timer error recovery code) but these codes can always be accessed through well defined Fortran Subprogram interfaces. The original Fortran version of Olympus has so far been successfully operated on at least 30 computer installations of 7 different types in several countries. This short paper gives only a brief description of the Olympus approach and the potential user is referred to the description given in (2) for more details.

2. PROGRAM STRUCTURE

The program is organised into a (universal) main program together with a set of subprograms, while the data is organised into labelled COMMON blocks. The subprograms are divided into classes m with decimal labels $\langle m,n \rangle$, and in the Olympus system the broad classification indicated in Table 1 has been adopted.

Class	Subprogram
0	Control
1	Prologue
2	Calculation
3	Ouput
4	Epilogue
5	Diagnostics
u	Utilities

Table 1. Classes of subprograms

The COMMON blocks are divided into groups r with labels (Cr.s), and on most computer systems it is usually possible to retain only one copy of each labelled block which can be accessed when required by means of a simple control statement. The COMMON blocks are divided into the five groups shown in Table 2. Each COMMON block has a Fortran name beginning with the letters COM.

Group	COMMON block
1	General Olympus data
2	Physical problem
3	Numerical scheme
4	Housekeeping
5	I/O and diagnostics

Table 2. Groups of labelled COMMON blocks

The Olympus system has a standard program CRONUS, which contains a set of basic control subprograms, which sets the basic structure for all programs in the Olympus family. The control programs call a set of standard (class 0) and dummy subprograms which in turn initialise data, make any modifications for the current run, control the calculation and output, and terminate the run. The user, where necessary, provides replacements for the dummies in order to solve his own particular problem. CRONUS contains the subprograms and COMMON blocks listed in Table 3, and apart from the programs in class 0, these routines are essentially dummies. Thus the programmer has to supply a subroutine STEPON ⟨2.1⟩ which organises the calculation step and calls in other subprograms ⟨2.2⟩ , ⟨2.3⟩ , and so on, to do the actual work.

Name	No.	Title	Status
Subprograms			
Class 0 Main Control			
(MAIN)	0.0	Fortran main program	P
BASIC	0.1	Initialise basic control data	C
MODIFY	0.2	Modify basic data if required	D
COTROL	0.3	Control the run	P
EXPERT	0.4	Modify standard operation of program	M

Table 3 (continued overleaf)

Name	No.	Title	Status
Class 1 Prologue			
LABRUN	1.1	Label the run	D
CLEAR	1.2	Clear variables and arrays	D
PRESET	1.3	Set default values	D
DATA	1.4	Define data specific to run	D
AUXVAL	1.5	Set auxiliary values	D
INITAL	1.6	Define physical initial conditions	D
RESUME	1.7	Resume run from previous record	D,M
START	1.8	Start the calculation	D
Class 2 Calculation			
STEPON	2.1	Step on the calculation	D
Class 3 Output			
OUTPUT	3.1	Control the output	D
Class 4 Epilogue			
TESEND	4.1	Test for completion of run	M
ENDRUN	4.2	Terminate the run	M
Class 5 Diagnostics			
REPORT	5.1	Control the diagnostics	M
CLIST	5.2	Print COMMON variables	D
ARRAYS	5.3	Print COMMON arrays	D
COMMON blocks			
Group 1		General OLYMPUS data	
COMBAS C1.1		Basic system parameters	
COMDDP C1.9		Development and diagnostic parameters	
Status codes			
C - Minor changes may be needed			
D - Dummies			
M - May be left in or modified as required			
P - Permanent control routine, should not require alteration			

Table 3. Cronus Subprograms and COMMON Blocks

The group 1. COMMON blocks (C1.1) and (C1.9) are standard library versions which are part of the Olympus package and are available to all programs. The variables and arrays in these blocks are listed in Tables 4 and 5 respectively. The structure of CRONUS is illustrated in fig.1.

Name	Type	Dimension	Purpose	Preset value
Variable names				
ALTIME	R		CPU time allocated for job (secs)	Supervisor Call
CPTIME	R		CPU time used so far (secs)	0.0
NLEDGE	I		Channel for restart records	30
NLEND	L		.TRUE.if run to be terminated	.FALSE.
NLRES	L		.TRUE. if run being resumed	.FALSE.
NONLIN	I		Channel for on-line input/output	1
NOUT	I		Current output channel	NPRINT
NPRINT	I		Channel for printed output	96
NREAD	I		Channel for card input (or equivalent)	95
NREC	I		Current record number	1
NRESUM	I		Resume from record on this channel	NLEDGE
NSTEP	I		Current step number	0
STIME	R		Start time (secs)	0.0
LABEL1	IA	12	Labels used to describe the run, set by program in LABRUN	BLANK
LABEL2	IA	12		
LABEL3	IA	12		
LABEL4	IA	12		
LABEL5	IA	12	Labels available to programmer	
LABEL6	IA	12		
LABEL7	IA	12	Labels reserved for system use	
LABEL8	IA	12		
NDIARY	I		Channel for diary	NPUNCH
NIN	I		Current input channel	NREAD
NPUNCH	I		Channel for punched card output (or equivalent)	97
NRUN	I		Maximum number of steps	1

Table 4. C1.1 COMBAS basic system parameters

Name	Type	Dimension	Purpose	Preset value
MAXDUM	I		Maximum dimension of dump arrays	20
MXDUMP	I		Actual dimension of dump arrays	10
NADUMP	IA	20	Codes for array dumps	0
NCLASS	I		Most recent class reported	0
NPDUMP	IA	20	Codes for dumping points	0
NPOINT	I		Most recent point recorded	0
NSUB	I		Most recent subprogram reported	1
NVDUMP	IA	20	Codes for dumping variables	0
NLCHED	L		.TRUE. if report heads needed for control class 0	.FALSE.

Table 5 (continued overleaf)

Name	Type	Dimension	Purpose	Preset value
NLHEAD	LA	9	.TRUE. if report heads needed for class 1-9	.FALSE.
NLOMT1	LA	50	.TRUE. if subprogram in class 1 to be omitted	.FALSE.
NLOMT2	LA	100	.TRUE. if subprogram in class 2 to be omitted	.FALSE.
NLOMT3	LA	50	.TRUE. if subprogram in class 3 to be omitted	.FALSE.
NLREPT	L		.TRUE. if any reports required	.FALSE.

Table 5. C.1.9. COMDDP development and diagnostic parameters

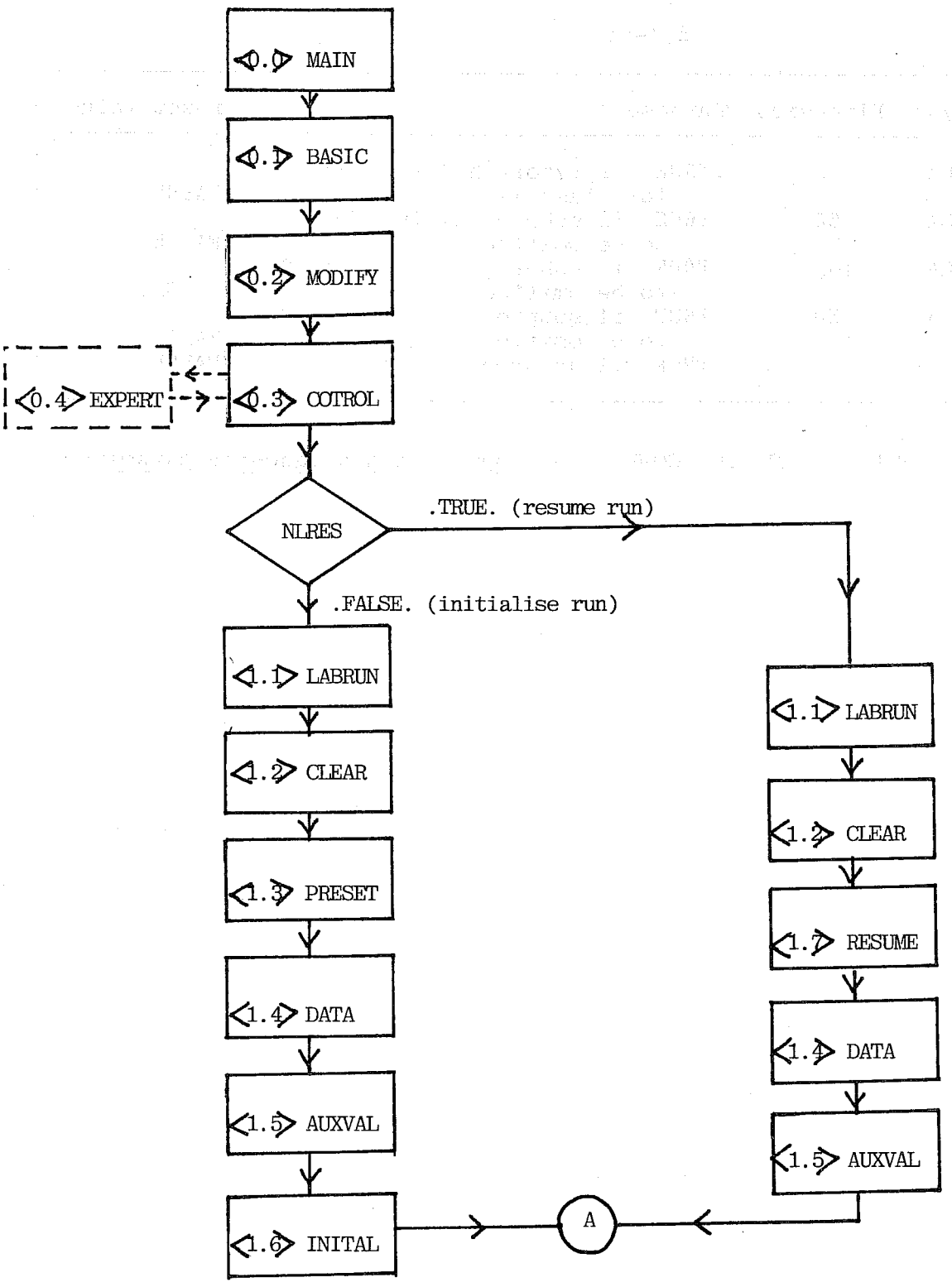


Figure 1. Flow diagram for CRONUS (part 1)

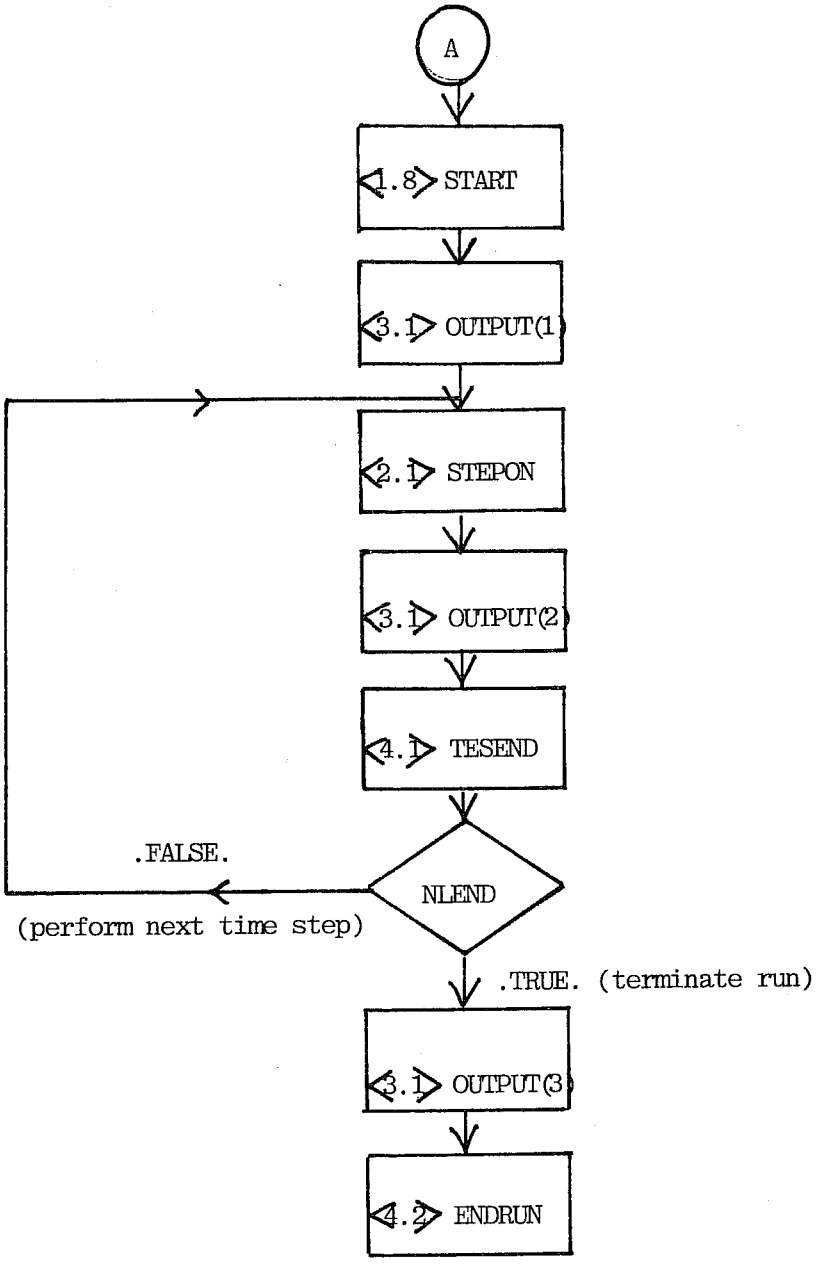


Figure 1. Flow diagram for CRONUS (part 2)

3. NOTATION, LAYOUT AND DOCUMENTATION

In order to make the program listing neater and easier to read, many programming elements have been standardised. These are summarised in Table 6. In addition to the numbering of subprogrmas and COMMON blocks, the individual subprograms are divided into numbered sections and sub-sections which are correlated with Fortran statement numbers. A useful technique is to cross reference the listing with a program commentary or 'write up' so that the two can be read together. An example of this is given in Appendix A. For mathematics sections the listing should refer to equation numbers in the 'write up' while the 'write up' refers to subprograms in the code. The initial letter conventions defined in Table 6a allow the reader to see at a glance which variables and arrays are in COMMON and which are local. In addition, symbolic rather than arithmetic notation should be used wherever possible. This makes programs more flexible, as well as easier to read.

4. DIAGNOSTICS AND UTILITIES

An elaborate set of diagnostic tools have been developed in order to enable programs to be checked as quickly as possible. The programs are designed so that the diagnostic output can be switched on and off either by including coded data in a NAMELIST input deck or by inserting a few extra statements. Facilities are available to output messages, to trace the flow of the program, to print the names and values of individual real, integer, logical or Hollerith variables or arrays, to print out COMMON blocks in alphanumeric order, and to switch individual subprograms off if they are found to contain catastrophic errors.

In order to 'back up' the diagnostics and to provide the user with an output package, a set of standard utilities are available (class U subprograms) see Table 7. There is no real need for the user to code a FORMAT statement within the problem area of his code. The utilities also include some routines for arithmetic manipulation.

(a) Decimal numbering scheme

Subprograms, e.g. <2.1> STEPON
Common blocks, e.g. <C4.1> COMIOC
Division of subprograms into sections and subsections
Statement numbers correlated with sections

(b) Notation for variables and arrays

Initial letters are used to distinguish between:
Common/Internal,
integer/logical/real
variable/formal
parameter/index
(see Table 6a. below)

(c) Layout

Standard columns
Spacers and ruled lines
(see Table 6b. below)

(d) Symbolic notation

Channel numbers, Table size
Character codes, constants
Dimensions

(e) Standardisation

Control variables, File names
Subprograms, Fundamental constants
Common blocks.

Table 6. Notation and Layout

	Real and complex	Integer and hollerith	Logical
Subprogram dummy arguments	P	K	KL
Common variable and array names	A-H, O, Q-Y	L,M,N	LL,ML,NL
Local variable and array names	Z	I	IL
Loop indices		J	

Table 6a. Initial letters and array names

```
Column 1        7              15             25
       C
       C
       C
       C
       CL                      2.Ø            CALCULATE NEXT ROW
       C
          2ØØ   CONTINUE
                NROW=NROW+1
       C
       CL                      2.1            READ NEW ROW
          21Ø   CONTINUE
                CALL READ1 (NROW, NUNIT, IERR)
```

Table 6b.

Name	No.	Dummy arguments	Title
MESAGE	U.1	KMESS	Print 48-character message on output channel
PAGE	U.2		Fetch new page on output channel
BLINES	U.3	K	Insert blank lines on output channel
RVAR	U.4	KNAME,PVALUE	Print name and value of real variable
IVAR	U.5	KNAME,KVALUE	Print name and value of integer variable
HVAR	U.6	KNAME,KVALUE	Print name and value of Hollerith variable
LVAR	U.7	KNAME,KLVAL	Print name and value of logical variable
RARRAY	U.8	KNAME,PA,KDIM	Print name and values of real array
IARRAY	U.9	KNAME,KA,KDIM	Print name and values of integer array
HARRAY	U.10	KNAME,KA,KDIM	Print name and values of Hollerith array
REPTHD	U.11	KCLASS,KSUB,KPOINT	Print heading for diagnostic report
RUNTIM	U.12		Update CPU time (secs) and print it
DAYTIM	U.13		Print date and time
RESETR	U.14	PA,KDIM,PVALUE	Reset real array to specified value
RESETI	U.15	KA,KDIM,KVALUE	Reset integer array to specified value
RESETH	U.16	KA,KDIM,KVALUE	Reset Hollerith array to specified value
JOBTIM	U.17	PTIME	Fetch allocated jobtime (secs)
RESETL	U.18	KLA,KDIM,KLVAL	Reset logical array to specified value
LARRAY	U.19	KNAME,KLA,KDIM	Print name and values of logical array
RARAY2	U.20	KNAME,PA,KDIMX,KX,KY	Print doubly-subscripted real array
SCALER	U.21	PA,KDIM,PC	Scale a real array by a real value
SCALE1	U.22	KA,KDIM,KC	Scale an integer array by an integer value
COPYR	U.23	PA1,K1,PA2,K2,KDIM	Copy one real array into another
COPY1	U.24	KA1,K1,KA2,K2,KDIM	Copy one integer array into another
SIGNR	U.25	PA,KDIM	Change the sign of a real array
SIGN1	U.26	KA,KDIM	Change the sign of an integer array
DUMCOM	U.27	KCLASS,KSUB,KPOINT	Dump selected common blocks

Table 7. Utility programs

5. PROGRAM CONTROL

All Olympus programs share the same main program <0.0> and the same master control program <0.3> . Initialisation of COMMON variables can be achieved in the subprograms PRESET <1.3> (default values), DATA <1.4> (alterations to small numbers of variables) and INITAL <1.6> (definition of physical initial conditions). NAMELIST (Non-Standard ANSI) is frequently used to input small amounts of data.

Finally ad hoc program modifications are made using subprogram EXPERT <0.4> which is called from many points throughout the code with the 3 parameters KCLASS, KSUB, KPOINT which identify the CALLING point. The user can then insert his program modifications in his own version of EXPERT leaving the original program unchanged. EXPERT also controls the diagnostics and may be useful for other purposes. Appendix B contains a listing of COTROL <0.3> , EXPERT is frequently called enabling modifcations to the program to be easily implemented.

.../12

CONCLUSION

A version of Olympus has been produced by members of the research department. The CDC 6500 version produced by the Culham group has been purchased by the operations group at a cost of about £10 (+ £11 tape handling charge). By standardising operational programs, by the use of Olympus and the NAG subroutine library, we are laying the foundations of a system whereby porgrams will be extremely portable, easy to understand and simple to modify. By reading the cross-referenced documentation while studying the program, a person unfamiliar with the program should very soon find himself in the situation whereby he can confidently modify or even debug the program.

Versions of Olympus for different machine ranges may be obtained from:

The CPC Program Library
Department of Applied Mathematics,
Queen's University,
Belfast BT7 1NN,
Northern Ireland

The Culham group has published many papers on Olympus and on programs which use the system, many of which are reproduced in the journal, Computer Physics Communications (CPC). The following references refer to some of these.

(1) K.V. Roberts CPC, 7, 237 (1974)
(2) CPC, 7, No. 5 (1974)
(3) M.H. Hughes and K.V. Roberts CPC, 8, 123, (1974)
(4) M.H. Hughes and K.V. Roberts CPC, 10, 167 (1975)

2. FLOW DIAGRAM

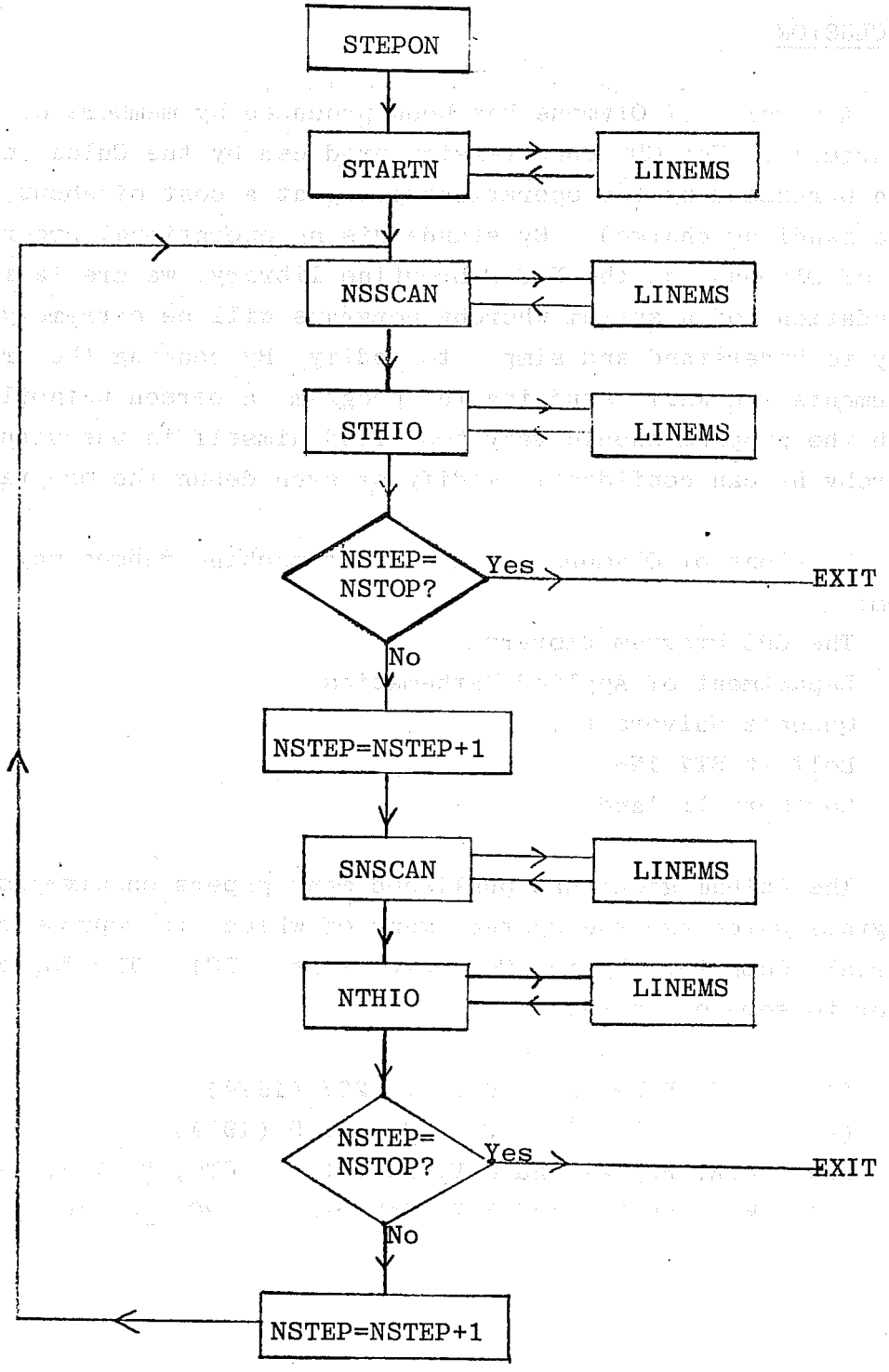


Fig. 1

The subroutine sequence is illustrated above. Each is described in greater detail below.

APPENDIX B: Examples of I/O Subroutines with Cross Reference Documentation

Gemini I/O Scheme

1. General Description

The I/O scheme uses 2 random access work files, containing data at adjacent time steps. Data for time NSTEP is read from the first file, the forecast equations are called to calculate the values at time NSTEP+1, and the new values are written, row by row, to the 2nd file. At the next time step, the data for time NSTEP is overwritten by the newly calculated values for time NSTEP+2.

The I/O scheme uses 4 input buffers and 2 output buffers. The forecast equations require data from the rows immediately to the north and south of the row which is currently being updated. While these three rows are being used in the calculation, the next row is being read into the 4th buffer using the subroutine TRANSR, which can read data in parallel with CPU processing. Similarly, while the new values for the current row are being calculated into one output buffer, the updated values for the previous row are being written from the second output buffer using the subroutine TRANSW, which can write data in parallel with CPU processing.

At the end of the calculation for one row, the pointers to the buffers are swapped cyclically so that the oldest input and output buffers are overwritten on the next row by new data.

The I/O scheme starts at the northern boundary and works south row by row to the southern boundary. At the next time step, the scheme works back from south to north, and this cycle is repeated until NSTEP=NSTOP, the final timestep.

3. STEPON

Subroutine STEPON controls the I/O scheme.

<1.1> - STARTN is called to start the I/O at the northern boundary.
<1.2> - NSSCAN is called to scan from north to south.
<1.3> - STHIO is called to do the I/O at the southern boundary.. Within STHIO, there is a test to check if the time NSTEP is equal to the time at the final I/O step, NSTOP (held in common /COMIOC/). If so, the logical variable NLEN (held in common /COMBAS/) is set to .TRUE. STEPON tests NLEND after each call to NTHIO and STHIO, and returns control to the subroutine COTROL if NLEND=.TRUE.
<1.4> - SNSCAN is called to scan from south to north.
<1.5> - NTHIO is called to do the I/O of the northern boundary. As in STHIO, NLEND is set to .TRUE. if NSTEP=NSTOP. If NLEND=.TRUE control returns to subroutine COTROL. If NLEND=.FALSE., the subroutine returns to <1.2>,NSSCAN for the next time step.

5. NSSCAN

Subroutine NSSCAN scans from north to south, with a read/calculate/write cycle for each row.

<1.1>- The subroutine tests to see if the row being updated, NROW, is within a row of the southern boundary. If NROW=MAXROW-1, control returns to STEPON, which in turn calls STHIO.
<1.2>- The buffer pointers are cyclically swapped, as in Fig. 3 below.

Calculate row NROW

Input Buffers		Output Buffers	
	TIME NSTEP		TIME NSTEP+1
NLINE1(1) ————	NROW-1	NLINE2(1) ---write---	NROW-1*
NLINE1(2) ————	NROW	NLINE2(2) ————	NROW*
NLINE1(3) ————	NROW+1		
NLINE1(4) ---read---	NROW+2		

Calculate row NROW+1

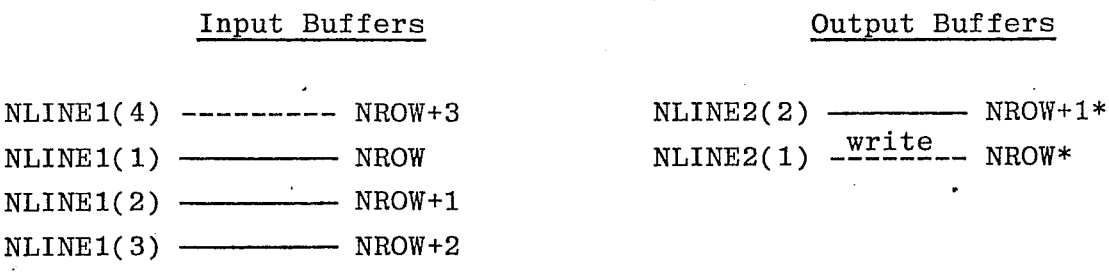


Fig. 3

⟨1.3⟩ - The read is initiated for row NROW+2 into the buffer to which NLINE1(4) points.

⟨1.4⟩ - The write is initiated for row NROW-1 from the buffer to which NLINE2(1) points.

⟨1.5⟩ - Subroutine LINEMS is called to calculate the values of row NROW at time NSTEP+1, and store the updated values in the buffer to which NLINE2(NORS) points.
NROW is incremented by 1, and the program jumps back to ⟨1.1⟩

Appendix 2

The Doctor Programming System

PROGRAM DOCTOR

THE DOCTOR SYSTEM - A DOCUMENTARY ORIENTED PROGRAMMING SYSTEM

J. K. GIBSON 5.2.80

CHAPTER ONE - SYSTEM OUTLINE

1.1 INTRODUCTION

DOCTOR IS A PROGRAMMING STANDARD BORN OUT OF THE SHORT-COMMINGS AND ADVANTAGES FOUND BY THE AUTHOR DURING HIS EXPERIENCE WITH OTHER PROGRAMMING STANDARDS. ALTHOUGH ITS APPLICATION WILL BE DESCRIBED WITH PARTICULAR REFERENCE TO FORTRAN, THERE IS NO REASON WHY, SUITABLY MODIFIED, SOME OR ALL OF ITS PRINCIPLES CANNOT BE USED IN CONJUNCTION WITH ANY PROGRAMMING LANGUAGE. MANY OF THE FEATURES ARE BORROWED UNASHAMEDLY FROM ROBERTS'S OLYMPUS SYSTEM. INDEED, DOCTOR COULD BE DESCRIBED AS A MODIFICATION OF OLYMPUS WITH MAJOR RELAXATIONS AND MINOR ADDITIONS. OTHER FEATURES HAVE BEEN BORROWED FROM THE CONTROL DATA CORPORATION'S PROGRAMMING STANDARD, ESPECIALLY THOSE FEATURES WHICH ALLOW R.H.FRANK'S DOCUMENTATION PROCESSOR TO BE USED ON THE RESULTING CODE. (DOCK, AN INTERNAL/EXTERNAL DOCUMENTATION PROCESSOR, IS A C.D.C. PROPRIETARY PRODUCT, COPYRIGHT CONTROL DATA CORPORATION, 1971)

1.2 THE NEED FOR DOCUMENTATION

ANY PROGRAMME OR ROUTINE WHICH IS LIKELY TO HAVE ANY LASTING VALUE MUST BE CAPABLE OF BEING UNDERSTOOD. DOCUMENTATION IS A MEANS OF RETAINING THE ABILITY OF CODE TO BE UNDERSTOOD. THE PRODUCTION OF SUCH DOCUMENTATION IS A SKILL AT LEAST AS IMPORTANT AS THE SKILL REQUIRED TO DESIGN AND GENERATE THE CODE ITSELF. GOOD DOCUMENTATION INCREASES THE VALUE OF CODE - IT ASSISTS MAINTENANCE, AIDS UNDERSTANDING, AND CAN BE INVALUABLE IF LANGUAGE TO LANGUAGE RECODING SHOULD EVER BE NECESSARY.

1.3 WHERE SHOULD DOCUMENTATION EXIST?

ONE OF THE BIGGEST PROBLEMS WITH DOCUMENTATION IS KEEPING IT UP TO DATE. WHEN DOCUMENTATION EXISTS AS TYPED OR PRINTED MATTERIAL THERE IS ALWAYS A DELAY IN PRODUCING AMMENDMENTS. THERE IS ALSO A VERY REAL DANGER OF CHANGING THE CODE, BUT NOT CHANGING THE DOCUMENTATION. THIS DANGER STILL EXISTS, BUT IS

CONSIDERABLY REDUCED, IF MUCH OF THE DOCUMENTATION IS INCORPORATED IN THE TEXT OF THE SOURCE CODE. IF FACILITIES EXIST TO PRODUCE THE WRITTEN DOCUMENTATION BY PROCESSING THE SOURCE CODE, DELAYS IN PRODUCING AMMENDMENTS ARE OVERCOME, AND LABOUR COSTS ARE REDUCED.

THE CONVENTION ADOPTED IN *DOCTOR* IS TO INCLUDE SUFFICIENT DOCUMENTATION WITHIN THE SOURCE CODE OF EACH PROGRAMME OR ROUTINE TO ENABLE IT TO BE UNDERSTOOD AND USED BY ANOTHER USER. ADDITIONAL DOCUMENTATION SHOULD BE CONFINED TO CROSS-REFERENCED LISTS OF EQUATIONS AND SYMBOLS ETC., WHICH CANNOT EASILY BE INCLUDED IN THE SOURCE CODE. THE AIM IS THEN ANOTHER USER. ADDITIONAL DOCUMENTATION SHOULD BE CONFINED TO TO PRODUCE AS MUCH OF THE DOCUMENTATION AS POSSIBLE BY PROCESSING THE SOURCE CODE.

1.4 BASIC AIMS OF THE SYSTEM

DOCTOR ATTEMPTS TO

(A) PROVIDE WELL PRESENTED CODE.
(B) PRODUCE SOURCE CODE FOLLOWING A STANDARD STRUCTURE.
(C) SET UP REFERENCE POINTS TO EXTERNAL DOCUMENTATION.
(D) ENABLE THE INCLUSION WITHIN THE SOURCE CODE OF DOCUMENTATION WHICH CAN BE EXTRACTED BY DOCUMENTATION EXTRACTION PROGRAMMES.
(E) ALLOW MAXIMUM COMMUNICATION BETWEEN SUBROUTINES BY STORING UNIVERSAL VARIABLES IN STRUCTURED POOLS OR COMMON BLOCKS.
(F) ENABLE VARIABLE TYPES TO BE RECOGNISED, AND THE DIFFERENTIATION BETWEEN VARIABLES OF LOCAL, COMMON AND DUMMY ARGUMENT TYPES.
(G) MAKE USE OF A UNIVERSAL SET OF UTILITY ROUTINES FOR WRITING MESSAGES, COPYING VECTORS, RESETTING ARRAYS, PRINTING DATA, ETC., MOST OF WHICH ARE BASED ON STANDARD *OLYMPUS* UTILITY ROUTINES.

CHAPTER TWO - SYSTEM DEFINITION

2.1 NAMES OF VARIABLES

2.1.1 VARIABLE TYPES

LOGICAL VARIABLES ARE PREFIXED BY KL, NL, OR IL. ALL OTHER VARIABLES FOLLOW THE STANDARD *FORTRAN* CONVENTION, I.E. VARIABLES PREFIXED I, J, K, L, M, N ARE *INTEGER*, WHILE VARIABLES PREFIXED BY OTHER LETTERS ARE *REAL*.

2.1.2 LOCAL VARIABLES

VARIABLES THAT ARE LOCAL TO A ROUTINE ARE PREFIXED IL IF THEY ARE *LOGICAL*, I IF THEY ARE *INTEGER*, OR Z IF THEY ARE *REAL*.

2.1.3 DUMMY ARGUMENTS

DUMMY ARGUMENTS TO ROUTINES ARE PREFIXED KL IF THEY ARE *LOGICAL*, K IF THEY ARE *INTEGER*, AND P IF THEY ARE *REAL*.

2.1.4 LOOP VARIABLES

DO LOOP CONTROL VARIABLES SHOULD HAVE NAMES BEGINNING WITH J.

2.1.5 COMMON VARIABLES

PREFIXES NOT RESERVED FOR LOCAL VARIABLES, DUMMY ARGUMENTS, OR LOOP CONTROL VARIABLES ARE AVAILABLE FOR USE AS GLOBAL VARIABLES ASSOCIATED WITH COMMON BLOCKS, SUBJECT TO THE TYPE CONVENTION CONTAINED IN 2.1.1 ABOVE.

2.2 COMMON BLOCKS

THE USE OF PARAMETERS FOR PASSING INFORMATION TO ROUTINES IS DISCOURAGED, EXCEPT FOR INPUT/OUTPUT AND UTILITY ROUTINES. IN GENERAL, INFORMATION REQUIRED BY MORE THAN ONE ROUTINE WILL BE AVAILABLE THROUGH A DATA POOL OR COMMON BLOCK.

2.3 MODULARITY AND ROUTINE ORGANISATION

CODE SHOULD BE BROKEN DOWN INTO A MODULAR STRUCTURE, EACH MODULE OR ROUTINE FULFILLING A GIVEN FUNCTION. *OVERVIEW* LEVEL COMMENTS AT THE HEAD OF EACH MODULE SHOULD STATE CLEARLY THE FUNCTION OF THE FOLLOWING ROUTINE. *EXTERNAL* LEVEL COMMENTS SHOULD CONTAIN DETAILS GIVING SUFFICIENT INFORMATION FOR THE MODULE TO BE USED AND UNDERSTOOD, WHILE *INTERNAL* LEVEL COMMENTS SHOULD INCLUDE SUFFICIENT INFORMATION FOR THE MODULE TO BE MODIFIED, AND MAINTAINED.

MODULES SHOULD BE DIVIDED INTO SECTIONS AND SUBSECTIONS. SECTIONS AND SUBSECTIONS SHOULD BE NUMBERED USING A TWO LEVEL NUMERICAL SCHEME (EG 2.1, 6.3, ETC.). STATEMENT LABELS SHOULD REFLECT THE NUMBERING SCHEME (EG IN *FORTRAN* SECTION 2.1 SHOULD BEGIN 210 CONTINUE). SECTIONS SHOULD BE RULED OFF FROM EACH OTHER BY A COMMENT LINE CONTAINING 65 MINUS SIGNS FROM COLUMN 7 TO COLUMN 71 INCLUSIVE.

2.4 COMMENT AND DOCUMENTATION LEVELS

2.4.1 INTRODUCTION

IT IS ENVISAGED THE A SOURCE CODE PROCESSOR WILL BE USED TO EXTRACT DOCUMENTATION FROM ROUTINES CODED TO THE *DOCTOR* STANDARD. ON CDC MACHINES, THE CDC PRODUCT *DOCK* MAY BE USED. FOR OTHER MACHINES IT SHOULD BE POSSIBLE TO CODE A SUITABLE PROCESSOR. AN EXAMPLE OF A SIMPLE *FORTRAN* PROCESSOR IS GIVEN IN APPENDIX 2.

TO ENABLE DIFFERENT LEVELS OF DOCUMENTATION TO BE PRODUCED FOUR TYPES OF DOCUMENTATION ARE DEFINED:-

MANDATORY - WILL ALWAYS BE LISTED.

OVERVIEW - OVERVIEW INFORMATION, TO BE LISTED WHEN AN OVERVIEW OF THE SOURCE CODE IS REQUIRED.

EXTERNAL - EXTERNAL DOCUMENTATION, INCLUDING OVERVIEW INFORMATION, TO BE LISTED WHEN A FULL EXTERNAL LISTING OF THE DOCUMENTATION IS REQUIRED.

INTERNAL - INTERNAL DOCUMENTATION, INCLUDING OVERVIEW AND EXTERNAL INFORMATION, TO BE LISTED WHEN A DETAILED DESCRIPTION DOWN TO THE FLOW OF EACH MODULE IS REQUIRED.

EACH LEVEL OF DOCUMENTATION IS TRIGGERED BY A COMMENT CARD OF APPROPRIATE FORM. THE FIRST CHARACTER OF THIS CARD DEPENDS ON THE LANGUAGE BEING USED. IN THE FOLLOWING DESCRIPTION, *FORTRAN* IS THE ASSUMED LANGUAGE, AND THE FIRST CHARACTER IS THE LETTER C.

2.4.1 MANDATORY LEVEL

TRIGGER:- C*** IN COLUMNS 1 TO 4.
TERMINATOR:- C*** IN COLUMNS 1 TO 4.

THE MANDATORY LEVEL ENSURES THAT ALL STATEMENTS BETWEEN THE TRIGGER AND THE TERMINATOR ARE LISTED (COMMENTS OR SOURCE CODE). THE LISTING WILL OCCUR REGARDLESS OF THE LEVEL OF DOCUMENTATION REQUESTED. THIS LEVEL IS ESPECIALLY USEFUL FOR LISTING RUN DEPENDENT STATEMENTS (EG DATA STATEMENTS) OR PARTICULARLY IMPORTANT FEATURES (EG FUNCTION DEFINING STATEMENTS, OPERATION DEFINITIONS, ETC.).

2.4.2 OVERVIEW LEVEL

TRIGGER:- C**** IN COLUMNS 1 TO 5

TERMINATOR:- NO MORE CONSECUTIVE COMMENTS CARDS.

THE OVERVIEW TRIGGER CARD SHOULD CONTAIN A TITLE. AS A MINIMUM, EACH ROUTINE SHOULD CONTAIN OVERVIEW CARDS GIVING:-

(A) A TITLE.

(B) THE AUTHOR'S NAME AND THE DATE WRITTEN.

(C) MODIFICATION DATES.

(D) A SHORT DESCRIPTION OF THE FUNCTION OF THE ROUTINE.

2.4.3 EXTERNAL LEVEL

TRIGGER:- C** IN COLUMS 1 TO 3

TERMINATOR:- NO MORE CONSECUTIVE COMMENT CARDS.

THE EXTERNAL TRIGGER CARD SHOULD CONTAIN A TITLE. AS A MINIMUM, EACH ROUTINE SHOULD CONTAIN EXTERNAL CARDS GIVING:-

(A) A TITLE

(B) LINKAGE DETAILS (CALLING SEQUENCE, PARAMETERS, ETC.).

(C) DEFAULT VALUES, IF ANY.

(E) DETAILS OF EXTERNAL REQUIREMENTS (EG INPUT FILES, ROUTINES REQUIRED,ETC.).

(F) ANY OTHER INFORMATION REQUIRED TO ENABLE AN EXTERNAL USER TO USE THE ROUTINE.

ALL APPROPRIATE INFORMATION CONCERNING THE METHOD USED, UNUSUAL DATA OR CODE STRUCTURES USED, ETC., SHOULD BE INCLUDED UNDER THE EXTERNAL LEVEL. IN PARTICULAR, THIS LEVEL SHOULD BE EXPANDED TO CONFORM TO THE DOCUMENTATION STANDARD IN FORCE OR DESIRED AT ANY INSTALLATION.

2.4.4 INTERNAL LEVEL

TRIGGER:- C* IN COLUMNS 1 AND 2.

TERMINATOR:- NO MORE CONSECUTIVE COMMENT CARDS.

THE INTERNAL TRIGGER CARD SHOULD CONTAIN A TITLE OR COMMENT. OFTEN THIS WILL SIMPLY BE THE TITLE OF A SECTION OR

SUB-SECTION WITHIN THE CODE. AS A MINIMUM, EACH SET OF INTERNAL LEVEL COMMENTS SHOULD CONTAIN THIS TITLE OR COMMENT. AN INTERNAL LEVEL COMMENT SHOULD APPEAR AFTER EACH BRANCH IN THE SOURCE CODE, AND AFTER EACH LABELLED STATEMENT TO WHICH A BRANCH MAY BE MADE. IF THIS IS CORRECTLY DONE, THE INTERNAL DOCUMENTATION WILL LIST THE FLOW PATH FOR EACH ROUTINE.

IT SHOULD NOT BE NECESSARY TO INCLUDE MANY ADDITIONAL COMMENTS IN THE INTERNAL DOCUMENTATION. WHERE IT IS DESIRABLE TO HAVE SOME EXTERNAL FORM OF DOCUMENTATION (LISTS OF EQUATIONS, ETC.) THESE SHOULD FOLLOW THE NUMBERING SYSTEM OF THE SOURCE CODE (OR VICE VERSA). THIS GIVES CODE WHICH CAN BE CROSS-REFERENCED WITH SUCH DOCUMENTATION.

2.5 STATEMENT NUMBERS

STATEMENT NUMBERS ARE RELATED TO THE SECTION AND SUB-SECTION OF THE ROUTINE BY THE CONVENTION DEFINED IN 2.3 ABOVE. STATEMENT NUMBERS ARE RESTRICTED TO *CONTINUE* AND *FORMAT* STATEMENTS, AND SHOULD NOT BE ASSOCIATED WITH ANY EXECUTABLE STATEMENT. EACH *DO* STATEMENT SHOULD BE ASSOCIATED WITH A SEPARATE, LABELLED *CONTINUE* STATEMENT AT THE END OF ITS RANGE. *DO* LOOPS WITH AN EXTENDED RANGE ARE NOT PERMITTED. WHERE POSSIBLE, THE *OLYMPUS* INPUT/OUTPUT ROUTINES AND UTILITIES SHOULD BE USED, TO AVOID THE USE OF MACHINE DEPENDENT FORMATS. *FORMAT* STATEMENTS SHOULD NORMALLY BE GROUPED TOGETHER AT THE END OF A ROUTINE, AND ARE ALLOCATED LABELS OF THE FORM 99XX. THEY MAY BE PLACED NEAR TO THE APPROPRIATE INPUT/OUTPUT CALL IF THIS ADDS TO THE CLARITY OF THE CODE (EG IN A LONG ROUTINE, TO ENABLE THE *FORMAT* TO BE VIEWED ON THE SAME PAGE AS THE I/O STATEMENT).

2.6 UTILITY ROUTINES

A SET OF UTILITY ROUTINES SHOULD BE AVAILABLE, BASED ON THE *OLYMPUS* SYSTEM, TO ASSIST IN THE PERFORMANCE OF SPECIALISED FUNCTIONS. APPENDIX 1 CONTAINS A LISTING OF EXAMPLES OF SUCH ROUTINES. THEY PROVIDE A MEANS OF OBTAINING ADDITIONAL INFORMATION WITH A MINIMUM OF EFFORT. DATA, ARRAYS, ETC., CAN BE LISTED WITHOUT THE NEED TO CODE FORMAT STATEMENTS - A FEATURE WHICH IS ESPECIALLY USEFUL DURING DEBUGGING OR WHEN INVESTIGATING SPECIAL PROBLEMS.

CHAPTER 3 - DOCUMENTATION OF THE UTILITY ROUTINES

THE ROUTINES THAT FOLLOW ENABLE SOME SIMPLE FUNCTIONS TO BE PERFORMED WITH A MINIMUM OF PROGRAMMING EFFORT. ROUTINES ARE PROVIDED TO PRINT REAL, INTEGER, LOGICAL, AND HOLLERITH VARIABLES AND ARRAYS WITHOUT THE PROGRAMMER

HAVING TO CODE FORMAT STATEMENTS. REAL AND INTEGER ARRAYS MAY BE RESET, COPIED, OR SCALED. SIMPLE MESSAGES MAY BE WRITTEN. THE NAMES OF THE ROUTINES SUGGEST THEIR FUNCTION. ALL ROUTINES PERFORM THE SAME FUNCTION AS, AND ARE MODIFICATIONS OF THE CORRESPONDING ROUTINES IN ROBERTS'S OLYMPUS SYSTEM.

Appendix 3

Doctor System Utilities

Table Appendix 3 (2) contains details of a set of utility routines available as part of the DOCTOR system. These routines enable basic functions to be performed, and format-free messages and annotated output of variables to be printed.

Table App. 3(1) illustrates the common block, COMDOC, used to contain system parameters used by the utility routines.

```
      COMDOC - SYSTEM PARAMETERS FOR DOCTOR SYSTEM

      J. K. GIBSON      5/2/80

      *OLYMPUS* COMBAS MODIFIED FOR DOCTOR SYSTEM

 COMMON /COMDOC/
+ NOUT,      NPRINT,    NREAD,     NIN,       NPUNCH,
+ LABEL1,    LABEL2,    LABEL3,    LABEL4,    LABEL5,    LABEL6,
+ LABEL7,    LABEL8
  DIMENSION
H LABEL1(5),            LABEL2(5),            LABEL3(5),
H LABEL4(5),            LABEL5(5),            LABEL6(5),
H LABEL7(5),            LABEL8(5)

      KEY TO VARIABLES:-

         NOUT   - OUTPUT STREAM FOR RESULTS
         NPRINT - OUTPUT STREAM FOR PRINT
         NREAD  - INPUT STREAM FOR CARD INPUT
         NIN    - INPUT STREAM FOR INPUT DATA
         NPUNCH - OUTPUT STREAM FOR CARD OUTPUT
         LABEL1 - LABEL1 TO LABEL8 CONTAIN SPACE FOR STORING
                  LABELS IN CHARACTER FORM
```

Table App. 3 (1) - Common block COMDOC

SUBROUTINE	CALL	DESCRIPTION
i)	ii)	iii)
MESAGE	CALL MESAGE (KMESS)	KMESS IS A 48 CHARACTER MESSAGE TO BE PRINTED ON FILE NOUT
PAGE	CALL PAGE	ADVANCE TO A NEW PAGE ON FILE NOUT
BLINES	CALL BLINES(K)	K IS THE NUMBER OF BLANK LINE TO BE WRITEN TO FILE NOUT
RVAR	CALL RVAR(KNAME, PVALUE)	PVALUE - VALUE OF VARIABLE TO BE PRINTED IN FILE NOUT
IVAR	CALL IVAR (KNAME,KVALUE)	KNAME - CHARACTER STRING CONTAINING VARIABLE NAME KVALUE-VALUE OF VARIABLE TO BE PRINTED ON FILE NOUT

SUBROUTINE	CALL	DESCRIPTION
i)	ii)	iii)
HVAR	CALL HVAR (KNAME, KVALUE)	KNAME - CHARACTER STRING CONTAINING VARIABLE NAME KVALUE - VALUE OF VARIABLE TO BE PRINTED ON FILE NOUT
LVAR	CALL LVAR (KNAME, KLVAL)	KNAME - CHARACTER STRING KLVAL - VALUE OF VARIABLE TO BE PRINTED
RARRAY	CALL RARRAY (KNAME,PA, KDIM)	KNAME - CHARACTER STRING PA - ARRAY TO BE PRINTED ON FILE NOUT KDIM - LENGTH OF ARRAY TO BE PRINTED
IRRAY	CALL IARRAY (KNAME,KA, KDIM)	KNAME - CHARACTER STRING CONTAINING ARRAY NAME KA - ARRAY OF VALUES TO BE PRINTED ON FILE NOUT KDIM - NUMBER OF VALUES IN ARRAY
HARRAY	CALL HARRAY (KNAME,KA, KDIM)	KNAME - CHARACTER STRING CONTAINING ARRAY NAME KA - ARRAY OF VALUES TO BE PRINTED ON FILE NOUT KDIM - NUMBER OF VALUES IN ARRAY

SUBROUTINE	CALL	DESCRIPTION
i)	ii)	iii)
RESETR	CALL RESETR (PA,KDIM, PVALUE)	PA - ARRAY OF VALUES TO BE RESET KDIM - NUMBER OF VALUES TO BE RESET PVALUE- VALUE TO WHICH PA IS TO BE RESET
RESETI	CALL RESETI (KA,KDIM, KVALUE)	KA - ARRAY TO BE RESET KDIM - NUMBER OF VALUES TO BE RESET KVALUE- VALUE TO WHICH KA IS TO BE RESET
RESETH	CALL RESETH (KA,KDIM, KVALUE)	KA - ARRAY TO BE RESET KDIM - NUMBER OF VALUES TO BE RESET KVALUE- VALUE TO WHICH KA IS TO BE RESET
RESETL	CALL RESETL (KLA,KDIM, KLVAL)	KLA - ARRAY TO BE RESET KDIM - NUMBER OF VALUES TO BE RESET KLVAL - VALUE TO WHICH KLA IS TO BE RESET
LARRAY	CALL LARRAY (KNAME,KLA, KDIM)	KNAME - CHARACTER STRING CONTAINING ARRAY NAME KLA - ARRAY CONTAINING VALUES TO BE PRINTED KDIM - NUMBER OF VALUES TO BE PRINTED

SUBROUTINE i)	CALL ii)	DESCRIPTION iii)
RARAY2	CALL RARAY2(KNAME, PA, KDIMX,KX,KY)	KNAME - CHARACTER STRONG CONTAINING NAME OF ARRAY PA - ARRAY CONTAINING VALUES TO BE PRINTED ON FILE NOUT KDIMX - DIMENSION OF FIRST SUBSCRIPT OF ARRAY PA KX - NUMBER OF FIRST SUBSCRIPT VALUES OF PA TO BE PRINTED KY - NUMBER OF SECOND SUBSCRIPT VALUES FOR WHICH KX FIRST SUBSCRIPT VALUES ARE TO BE PRINTED
SCALER	CALL SCALER(PA, KDIM,PC)	PA - ARRAY TO BE SCALED KDIM - NUMBER OF VALUES OF PA TO BE SCALED PC - SCALE FACTOR
SCALEI	CALL SCALEI(KA, KDIM, KC)	KA - ARRAY TO BE SCALED KDIM - NUMBER OF VALUES OF KA TO BE SCALED KC - SCALE FACTOR
COPYR	CALL COPYR(PA1, K1, PA2, K2,KDIM)	PA1 - ARRAY TO BE COPIED K1 - PA1(K1) IS THE FIRST VALUE OF PA1 TO BE COPIED PA2 - RESULT ARRAY K2 - PA2(K2) RECEIVES THE FIRST COPIED VALUE KDIM - NUMBER OF VALUES TO BE COPIED

SUBROUTINE i)	CALL ii)	DESCRIPTION iii)
COPYI	CALL COPYI(KA1, K1, KA2, K2, KDIM)	KA1 - ARRAY TO BE COPIED K1 - KA1 (K1) IS THE FIRST VALUE OF PA1 TO BE COPIED KA2 - RESULT ARRAY K2 - KA2(K2) RECEIVES THE FIRST COPIED VALUE KDIM - NUMBER OF VALUES TO BE COPIED
SIGNR	CALL SIGNR(PA, KDIM)	PA - ARRAY CONTAINING VALUES TO BE NEGATED KDIM - NUMBER OF VALUES OF PA TO BE NEGATED
SIGNI	CALL SIGNI(KA, KDIM)	KA - ARRAY OF VALUES TO BE NEGATED KDIM - NUMBER OF VALUES TO BE NEGATED

Appendix 4

DOC - A Documentation Extraction Programme

```
PROGRAM DOC(TAPE8,INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
```

```
PROGRAM DOC(TAPE8,INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT                80/07/0

      DOC - PROGRAMME TO LIST SOURCE DOCUMENTATION
            WRITTEN TO *DOCTOR* STANDARD.

      J. K. GIBSON  4/2/80

      *DOC*

      LANGUAGE - NON-STANDARD FORTRAN, SUITABLE FOR CDC.

      PURPOSE - TO READ SOURCE CODE WRITTEN IN FORTRAN TO
                *DOCTOR* SPECIFICATION AND EXTRACT VARIOUS
                LEVELS OF DOCUMENTATION.
```

```
      DOC - EXTERNAL DOCUMENTATION

      DOC IS A SINGLE MAIN PROGRAMME, REQUIRING AN INPUT AND AN
      OUTPUT FILE; A SEPARATE FILE MAY BE USED FOR CONTROL DATA.

      INPUT FILE - CONTAINS SOURCE STATEMENTS OF SOURCE CODE.

      OUTPUT FILE - CONTAINS LISTING PRODUCED BY DOC.

      CONTROL DATA - THE LEVEL OF DOCUMENTATION IS SUPPLIED ON
                     A SINGLE STATEMENT. THIS CONTROL STATEMENT
                     MAY BE THE FIRST CARD OF THE INPUT FILE, OR
                     CAN BE PRESENTED AS A SEPARATE FILE.

      CONTROL DATA FORMAT - A SINGLE CARD CONTAINING:-

      12345678901234567890123456789012345678901234567890
                1         2         3         4
      DOCTOR    CCC

      WHERE CCC IS:-  *OVR* - OVERVIEW DOCUMENTATION ONLY
                      *EXT* - EXTERNAL + OVERVIEW DOCUMENTATION
                      *INT* - EXTERNAL + INTERNAL + OVERVIEW

      DATA STATEMENTS - DATA STATEMENTS WHICH CONTROL FILE
                        ALLOCATION AND COMMENT CONTROL ARE
                        LISTED BELOW.

      CRITICAL VARIABLES - THE FOLLOWING VARIABLES ARE IMPORTANT
                           IF IT IS DESIRED TO CHANGE THE LISTED
                           DATA STATEMENTS:-

            IC      - CONTAINS COMMENT CONTROL CHARACTER
            IN      - NUMBER OF INPUT STREAM CONTAINING SOURCE
            ICONT   - NUMBER OF INPUT STREAM CONTAINING CONTROL CARD
            IOUT    - NUMBER OF OUTPUT STREAM FOR OUTPUT LISTING
            INLINE  - NUMBER OF LINES PER PAGE
```

```
PROGRAM DOC(TAPE8,INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT                80/07/
```

DATA STATEMENTS

```
DATA IC/1HC/,IN/8/,ICONT/5/,IOUT/6/,INLINE/55/


1.          FIND CONTROL CARD AND SET VARIABLES

1.1         READ FIRST CARD ON IN
BRANCH TO 9.1 IF NO DATA ON FILE IN
BRANCH TO 1.3 IF FIRST CARD LOOKS LIKE CONTROL CARD

1.2         READ CONTROL CARD FROM FILE ICONT
BRANCH TO 9.2 IF NO DATA ON FILE ICONT

BRANCH TO 1.2 IF CONTROL CARD INVALID

1.4         FIND DOCUMENTARY LEVEL REQUIRED
BRANCH TO 9.3 IF INVALID LEVEL REQUESTED

2.          MAIN LOOP

2.1         READ CARD AND TEST TYPE
BRANCH TO 300 AT END OF DATA ON FILE IN

ENTER MAIN LOOP HERE IF FIRST CARD ALREADY READ

BRANCH TO 2.15 IF LIST SWITCH IS ON

BRANCH BACK TO 2.1 IF LIST OFF AND NOT COMMENT

BRANCH BACK TO 2.1 IF LIST OFF AND NOT NEW LEVEL

2.15        LIST SWITCH ON
BRANCH TO 2.65 IF NOT COMMENT AND MANDATORY LISTING
BRANCH TO 2.8 IF NOT COMMENT AND NOT MANDATORY LIST

2.2         COMMENT CARDS - TEST FOR LEVEL
BRANCH TO 2.6 IF NO NEW LEVEL FOUND

2.3         NEW LEVEL - OBTAIN VALUE
2.4         TEST FOR MANDATORY LEVEL
BRANCH TO 2.7 IF NOT MANDATORY LEVEL

2.5         MANDATORY LEVEL COMMENT - SET SWITCHES
2.6         MODIFY AND PRINT CARD
2.65        PRINT CARD
BRANCH BACK TO BEGINNING OF MAIN LOOP (2.1)

2.7         NEW LEVEL NOT MANDATORY - IS IT REQUIRED?
IF NOT REQUIRED BRANCH TO 2.8
```

IF REQUIRED, SWITCH ON PRINT AND BRANCH TO 2.6

2.8 END LIST SEQUENCE - SWITCH OFF PRINT
BRANCH BACK TO 2.1

---

3. END OF CARD INPUT

END OF PROGRAMME

---

9. ERROR HANDLING

9.1 NO DATA ON FILE IN

9.2 NO CONTROL CARD ON FILE ICONT

9.3 INVALID CONTROL CARD

DOC - END OF LISTING

```
      PROGRAM DOC(TAPE8,INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C****     DOC - PROGRAMME TO LIST SOURCE DOCUMENTATION
C              WRITTEN TO *DOCTOR* STANDARD.
C
C         J. K. GIBSON  4/2/80
C
C         *DOC*
C
C         LANGUAGE - NON-STANDARD FORTRAN, SUITABLE FOR CDC.
C
C         PURPOSE - TO READ SOURCE CODE WRITTEN IN FORTRAN TO
C                   *DOCTOR* SPECIFICATION AND EXTRACT VARIOUS
C                   LEVELS OF DOCUMENTATION.
C
C     ------------------------------------------------------------------
C
C**       DOC - EXTERNAL DOCUMENTATION
C
C         DOC IS A SINGLE MAIN PROGRAMME, REQUIRING AN INPUT AND AN
C         OUTPUT FILE; A SEPARATE FILE MAY BE USED FOR CONTROL DATA.
C
C         INPUT FILE - CONTAINS SOURCE STATEMENTS OF SOURCE CODE.
C
C         OUTPUT FILE - CONTAINS LISTING PRODUCED BY DOC.
C
C         CONTROL DATA - THE LEVEL OF DOCUMENTATION IS SUPPLIED ON
C                        A SINGLE STATEMENT. THIS CONTROL STATEMENT
C                        MAY BE THE FIRST CARD OF THE INPUT FILE, OR
C                        CAN BE PRESENTED AS A SEPARATE FILE.
C
C         CONTROL DATA FORMAT - A SINGLE CARD CONTAINING:-
C
C         12345678901234567890123456789012345678901234567890
C                   1         2         3         4
C         DOCTOR    CCC
C
C         WHERE CCC IS:-  *OVR* - OVERVIEW DOCUMENTATION ONLY
C                         *EXT* - EXTERNAL + OVERVIEW DOCUMENTATION
C                         *INT* - EXTERNAL + INTERNAL + OVERVIEW
C
C         DATA STATEMENTS - DATA STATEMENTS WHICH CONTROL FILE
C                           ALLOCATION AND COMMENT CONTROL ARE
C                           LISTED BELOW.
C
C         CRITICAL VARIABLES - THE FOLLOWING VARIABLES ARE IMPORTANT
C                              IF IT IS DESIRED TO CHANGE THE LISTED
C                              DATA STATEMENTS:-
C
C               IC      - CONTAINS COMMENT CONTROL CHARACTER
C               IN      - NUMBER OF INPUT STREAM CONTAINING SOURCE
C               ICONT   - NUMBER OF INPUT STREAM CONTAINING CONTROL CA
C               IOUT    - NUMBER OF OUTPUT STREAM FOR OUTPUT LISTING
C               INLINE  - NUMBER OF LINES PER PAGE
C
C     ------------------------------------------------------------------
C
```

```
PROGRAM DOC         74/175  OPT=1 ROUND=+-*/ MANTRAP          FTN 4.7+470


      LOGICAL ILIST,ILMAND,ILCARD
      DIMENSION ICARD(13),ICONCD(7),IDOCT(5),ITP(3)
C
C***              DATA STATEMENTS
C
      DATA IC/1HC/,IN/8/,ICONT/5/,IOUT/6/,INLINE/55/
C
C***
      DATA ITP/10HINT       ,10HEXT       ,10HOVR       /
      DATA IBLANK/1H /,IDOCT/1HD,1HO,1HC,1HT,1HO/
      DATA ISTAR/1H*/,IPAGE/0/,IILINE/0/
      DATA ILIST/.FALSE./,ILMAND/.FALSE./,ILCARD/.FALSE./
C
C     ------------------------------------------------------------------------
C
C*                1.        FIND CONTROL CARD AND SET VARIABLES
C
  100 CONTINUE
      WRITE(IOUT,9905)
C
C*                1.1       READ FIRST CARD ON IN
  110 CONTINUE
      READ(IN,9901)ICARD
      IF (EOF(IN).NE.0.0) GO TO 910
C*                BRANCH TO 9.1 IF NO DATA ON FILE IN
      DO 112 J=1,7
      ICONCD(J)=ICARD(J)
  112 CONTINUE
C
      IF (ICARD(1).EQ.IDOCT(1)) GO TO 130
C
C*                BRANCH TO 1.3 IF FIRST CARD LOOKS LIKE CONTROL CARD
C
C                 1.2       READ CONTROL CARD FROM FILE ICONT
  120 CONTINUE
      READ(ICONT,9901)ICONCD
      IF (EOF(ICONT).NE.0.0) GO TO 920
C
C*                BRANCH TO 9.2 IF NO DATA ON FILE ICONT
C
      ILCARD=.TRUE.
C
C                 1.3       CHECK CONTROL CARD
  130 CONTINUE
      DO 132 J=1,5
      IF (ICONCD(J).NE.IDOCT(J)) GO TO 120
  132 CONTINUE
C
C*                BRANCH TO 1.2 IF CONTROL CARD INVALID
C
C                 1.4       FIND DOCUMENTARY LEVEL REQUIRED
  140 CONTINUE
      DO 142 J=1,3
      IF (ICONCD(7).EQ.ITP(J)) GO TO 144
  142 CONTINUE
      GO TO 930
C
```

```
PROGRAM DOC         74/175  OPT=1 ROUND=+-*/ MANTRAP        FTN 4.7+470

      C*                BRANCH TO 9.3 IF INVALID LEVEL REQUESTED
      C
        144 CONTINUE
            IIREQD=J
            IF (ILCARD) GO TO 212
      C
      C     ------------------------------------------------------------------
      C
      C*                2.          MAIN LOOP
      C                             ---- ----
        200 CONTINUE
      C
      C*                2.1         READ CARD AND TEST TYPE
        210 CONTINUE
            READ(IN,9901)ICARD
            IF (EOF(IN).NE.0.0) GO TO 300
      C
      C*                BRANCH TO 300 AT END OF DATA ON FILE IN
      C
        212 CONTINUE
      C
      C*                ENTER MAIN LOOP HERE IF FIRST CARD ALREADY READ
      C
            IF (ILIST) GO TO 215
      C
      C*                BRANCH TO 2.15 IF LIST SWITCH IS ON
      C
            IF (ICARD(1).NE.IC) GO TO 210
      C
      C*                BRANCH BACK TO 2.1 IF LIST OFF AND NOT COMMENT
      C
            IF (ICARD(2).NE.ISTAR) GO TO 210
      C
      C*                BRANCH BACK TO 2.1 IF LIST OFF AND NOT NEW LEVEL
      C
      C                 2.15        LIST SWITCH ON
        215 CONTINUE
            IF (ILMAND.AND.ICARD(1).NE.IC) GO TO 265
      C*                BRANCH TO 2.65 IF NOT COMMENT AND MANDATORY LISTING
            IF (ICARD(1).NE.IC) GO TO 280
      C
      C*                BRANCH TO 2.8 IF NOT COMMENT AND NOT MANDATORY LIST
      C
      C                 2.2         COMMENT CARDS - TEST FOR LEVEL
        220 CONTINUE
            IF (ICARD(2).NE.ISTAR) GO TO 260
      C*                BRANCH TO 2.6 IF NO NEW LEVEL FOUND
      C
      C                 2.3         NEW LEVEL - OBTAIN VALUE
        230 CONTINUE
            DO 232 J=1,3
            IF (ICARD(J+2).NE.ISTAR) GO TO 240
        232 CONTINUE
            J=4
      C
      C*                2.4         TEST FOR MANDATORY LEVEL
        240 CONTINUE
```

```
      IF (J.NE.3) GO TO 270
C*                BRANCH TO 2.7 IF NOT MANDATORY LEVEL
C
C                 2.5         MANDATORY LEVEL COMMENT - SET SWITCHES
  250 CONTINUE
      ILMAND=.NOT.ILMAND
      ILIST=ILIST.OR.ILMAND
C
C*                2.6         MODIFY AND PRINT CARD
  260 CONTINUE
      DO 262 J=1,5
      ICARD(J)=IBLANK
  262 CONTINUE
C
C*                2.65        PRINT CARD
  265 CONTINUE
      IILINE=IILINE+1
      IF (MOD(IILINE,INLINE).NE.0) GO TO 266
      IPAGE=IPAGE+1
      IILINE=0
      WRITE(IOUT,9902)IPAGE
  266 CONTINUE
      WRITE(IOUT,9903)ICARD
      GO TO 210
C
C*                BRANCH BACK TO BEGINNING OF MAIN LOOP (2.1)
C
C*                2.7         NEW LEVEL NOT MANDATORY - IS IT REQUIRED?
  270 CONTINUE
      J=MIN0(J,3)
      IF (J.LT.IIREQD) GO TO 280
C
C*                IF NOT REQUIRED BRANCH TO 2.8
C
      ILIST=.TRUE.
      GO TO 260
C
C*                IF REQUIRED, SWITCH ON PRINT AND BRANCH TO 2.6
C
C                 2.8         END LIST SEQUENCE - SWITCH OFF PRINT
  280 CONTINUE
      ILIST=.FALSE.
      GO TO 210
C*                BRANCH BACK TO 2.1
C
C     ------------------------------------------------------------------
C
C                 3.          END OF CARD INPUT
C                             --- -- ---- -----
C
  300 CONTINUE
      DO 302 J=IILINE,INLINE
      WRITE(IOUT,9903)
  302 CONTINUE
      IPAGE=IPAGE+1
      WRITE(IOUT,9902)IPAGE
      WRITE(IOUT,9904)
```

```
PROGRAM DOC          74/175  OPT=1 ROUND=+-*/ MANTRAP          FTN 4.7+470          8C

            STOP
C*                  END OF PROGRAMME
C
C     ------------------------------------------------------------------------------
C
C                   9.          ERROR HANDLING
C                               ------- ----------
C
C                   9.1         NO DATA ON FILE IN
C
  910 CONTINUE
      WRITE(IOUT,9910)IN
      GO TO 990
C
C*                  9.2         NO CONTROL CARD ON FILE ICONT
C
  920 CONTINUE
      WRITE(IOUT,9920)ICONT
      GO TO 990
C
C*                  9.3         INVALID CONTROL CARD
C
  930 CONTINUE
      WRITE(IOUT,9930)
  990 CONTINUE
      STOP
 9901 FORMAT(5A1,A5,7A10)
 9902 FORMAT(1H0,35X,4HPAGE,I6/1H1)
 9903 FORMAT(1X,5A1,A5,7A10)
 9904 FORMAT(23H1END OF DATA ON FILE IN)
 9905 FORMAT(1H1)
 9910 FORMAT(16H1NO DATA ON FILE,I4)
 9920 FORMAT(24H1NO CONTROL CARD ON FILE,I4)
 9930 FORMAT(48H1PROGRAMME TERMINATED - CONTROL CARD INVALID     )
C
C*                  DOC - END OF LISTING
C
C     ------------------------------------------------------------------------------
C
      END
```

```
SYMBOLIC REFERENCE MAP (R=1)

OINTS
DOC

ES      SN  TYPE            RELOCATION
IBLANK      INTEGER                          6402  IC        INTEGER
ICARD       INTEGER     ARRAY                6554  ICONCD    INTEGER     ARRAY
ICONT       INTEGER                          6563  IDOCT     INTEGER     ARRAY
IILINE      INTEGER                          6536  IIREQD    INTEGER
ILCARD      LOGICAL                          6413  ILIST     LOGICAL
ILMAND      LOGICAL                          6403  IN        INTEGER
```

Appendix 5

DOCK - CDC Documentation Extraction Programme

DOCK - A D O C U M E N T A T I O N PROCESSOR

DOCK - A D O C U M E N T A T I O N PROCESSOR 80/07/0

DOCK - INTERNAL/EXTERNAL DOCUMENTATION PROCESSOR.

R.H.FRANK 71/11/08.

OPERATING SYSTEM - 6000 SERIES - SCOPE 3.3 / SCOPE 3.4
6000 SERIES - KRONOS
7000 SERIES - SCOPE 1.0
(*KRONOS* - ASSUMED TO BE THE SAME AS *MACE*.)

DOCK

THE PURPOSE OF THIS PROGRAM IS TO PROVIDE FOR THE USER/CODER ON CDC MACHINES A PRACTICAL DOCUMENTATION PROCESSOR, GENERAL ENOUGH TO BE USED FOR MACHINE ORIENTED LANGUAGES.

WITH THE GREATER DEGREE OF JOB SPECIALIZATION, THE INCREASE IN COMPLEXITY OF HARDWARE, THE PROLIFERATION OF PROGRAMMING LANGUAGES, AND THE COMMUNICATIONS NEED FOR A STANDARD, DOCK AND KODING ARE PRESENTED TO YOU.

PROGRAMMERS, UNLIKE PEOPLE, ALL TOO OFTEN RELATE TOO CLOSELY WITH THE MACHINE, FORGETTING THAT THE CODE WITHIN MUST ALSO BE OF USABLE FORM TO HUMANS.

DOCUMENTING A PROGRAM IS AN INTEGRAL SKILL OF CODING....

```
        OS - OPERATING SYSTEM
OS      EQU     0               DEFINE SCOPE  ASSEMBLY
OS      EQU     1               DEFINE KRONOS ASSEMBLY
```

CONTROL CARD CALL.

DOCK(P1,P2,P3,...,PN)

DEFINITION
DEFAULT, IF PARAMETER NOT SPECIFIED.
ASSUMED, IF PARAMETER SPECIFIED, BUT NOT EQUIVALENCED.

I= NAME OF PROGRAM *SOURCE* FILE.
(ASSUMED TO BE A *UPDATE* COMPILE OR SOURCE FILE
OF UP TO 90 COLUMN BCD CHARACTERS.)
DEFAULT = *SOURCE*.
ASSUMED = *COMPILE*.

L= NAME OF FILE ON WHICH DOCUMENTATION IS LISTED.
(CANNOT BE THE SAME NAME AS *I*.)
DEFAULT = ASSUMED = OUTPUT.

F= CAN BE UP TO 25 CHARACTERS AND WILL BE PRINTED AT THE
BOTTOM LEFT CORNER OF EACH PAGE OF DOCUMENTATION.
DEFAULT *INT*, FOLIO = $INTERNAL DOCUMENTATION.$
EXT, FOLIO = $EXTERNAL DOCUMENTATION.$
OVR, FOLIO = $OVERVIEW DOCUMENTATION.$
ASSUMED = $I M S.$

INT INTERNAL - ALL INTERNAL AND EXTERNAL
DOCUMENTATION WILL BE LISTED ON FILE *L*.

EXT EXTERNAL - ONLY EXTERNAL DOCUMENTATION WILL BE
LISTED ON FILE *L*.
DEFAULT = *INT*.

OVR OVERVIEW - ONLY OVERVIEW DOCUMENTATION WILL BE
LISTED ON FILE *L*.

INDEX BUILDS AN INDEX THAT IS PRINTED AT THE END OF EACH
ROUTINE OF ALL SYMBOLS FOUND IN LOCATION FIELD OF --
EJECT, SPACE, TITLE, AND TTL CARDS
ROUTINE PROCESSED.
DEFAULT = INDEX OFF.

NR NO REWIND OF INPUT FILE. (*I*)
DEFAULT = REWIND OF INPUT.

NT NO TABLE GENERATION.
DEFAULT = TABLE GENERATION.

NP NO PROPROGATION OF PAGE NUMBERS ACROSS ROUTINE.
DEFAULT = PAGE PROPROGATION.

TE DOCUMENTATION FILE, *L*, FORMATTED FOR INPUT INTO
PROGRAM *TEXTJAB*.

```
          DEFAULT = NO *TEXTJAB* OUTPUT.

     DEFAULT PARAMETER SETTINGS.

     DOCK(I=SOURCE,L=OUTPUT,F=$INTERNAL DOCUMENTATION.$,INT)

     ASSUMED PARAMETER SETTINGS.

     DOCK(I=COMPILE,L=OUTPUT,F=$I M S.$,INT)
```

DAYFILE MESSAGES ISSUED BY *DOCK*.

FL TOO SHORT FOR DOCK. (REQUIRES 12K).

NOT ENOUGH FIELD LENGTH WAS ALLOWED. CURRENT MINIMUM FIELD LENGTH IS 12K (OCTAL).

FILE NAME CONFLICT.

INPUT, *I*, AND LIST,*L*, FILE NAME ARE THE SAME.

MEMORY OVERFLOW IN BUILDING INDEX TABLE.

NOT ENOUGH FIELD LENGTH FOR INDEX TABLE. INCREASE FIELD LENGTH BY 4K (OCTAL).

EMPTY INPUT FILE. NO DOCUMENTATION PROCUCED.

INPUT FILE WAS EMPTY.

* INPUT FILE NAME IS ILLEGAL.*
* OUTPUT FILE NAME IS ILLEGAL.*

ILLEGAL CHARACTER SPECIFIED IN FILE NAME.

* FILE EQUIVALENCE MAY NOT BE 0.*

A FILE PARAMETER CANNOT BE SET TO ZERO.

GENERAL DISCRIPTION OF DOCUMENTATION PRODUCED BY *DOCK*.

A. GENERAL CONVENTIONS.

0. *OVERVIEW DOCUMENTATION* -

FIVE ASTERISKS STARTING IN COLUMN 1 START DOCUMENTATION WHICH CONTINUES UNTIL ALL CONSECUTIVE CARDS WITH COLUMN 1 ASTERISKS HAVE BEEN EXHAUSTED.
IE.

CORRECT FORM -

```
*****     NAME - DESCRIPTION
*
*         A.B. ORIGINAL.          77/11/28.
*
*         A.B. MODIFIER.          78/11/28.
```

INCORRECT FORM -

```
*****
*
*         NAME - DESCRIPTION
```

1. *EXTERNAL DOCUMENTATION* -
THREE ASTERISKS STARTING IN COLUMN 1 START DOCUMENTATION WHICH CONTINUES UNTIL ALL CONSECUTIVE CARDS WITH COLUMN 1 ASTERISKS HAVE BEEN EXHAUSTED.
FIVE ASTERISKS STARTING IN COLUMN 1 ACT THE SAME AS THREE ASTERISKS.
IE.

CORRECT FORM -

```
***       CONTROL CARD CALL.
*         XXX(P1,...,PN)
```

INCORRECT FORM -

```
***
*
*         CONTROL CARD CALL.
*         XXX(P1,,,,,,,PN)
```

2. *INTERNAL DOCUMENTATION* -
TWO OR THREE ASTERISKS STARTING IN COLUMN 1 START DOCUMENTATION WHICH CONTINUES UNTIL ALL CONSECUTIVE CARDS CONTAINING COLUMN 1 ASTERISKS HAVE BEEN EXHAUSTED.

IN ADDITION, ANY CARD WITH FOUR (4) ASTERISKS STARTING IN COLUMN 1 ACTS AS A TOGGLE FOR DOCUMENTATION. IN THIS MANNER WHEN A CARD IS FOUND TO CONTAIN 4 ASTERISKS STARTING IN COLUMN 1, THAT CARD AND ALL SUCCEDING CARDS THROUGH ANOTHER LIKE CARD (REGARDLESS OF THE COLUMN 1

CHARACTER) ARE CONSIDERED TO BE PART OF THE DOCUMENTATION.

FIVE ASTERISKS STARTING IN COLUMN 1 ACT THE SAME AS TWO AND THREE ASTERISKS.

IE.

CORRECT FORMS -

```
**          RNR - READ NEXT RECORD.
*           ENTRY

****        ASSEMBLY CONSTANTS.
 CH         EQU     10B
****
```

INCORRECT FORMS -

```
**
*
*           RNR - READ NEXT RECORD
*           ENTRY

****
**          ASSEMBLY CONSTANTS.
 CH         EQU     10B
****
```

B. SPECIAL FORMS

1. **DOCK OPTION SELECTION.
A CARD STARTING WITH 2 ASTERISKS IN COLUMNS 1 AND 2 WITH DOCK STARTING IN COLUMN 3 AND SELECTION IN COLUMN 18 IS PROCESSED AS A SPECIAL CONTROL TO DOCK.
CURRENTLY *OPTION* MAY BE EITHER

A. *LIST* SELECTION.
SELECTION MAY BE -

ON
OFF
INTERNAL
EXTERNAL
*

DEFAULT = *TYPE* SELECTION FROM CONTROL CARD.

ON = SAME AS *, RETURNS TO LAST SELECTION.
OFF = TURN ALL DOCUMENTATION OFF UNTIL NEXT DOCK LIST OPTION FOUND.
INTERNAL = PROCESS INTERNAL DOCUMENTATION.
EXTERNAL = PROCESS EXTERNAL DOCUMENTATION.
* = RESET LIST OPTION TO LAST SELECTION.

B. *TITLE* SELECTION.
SELECTION MAY BE -

```
                                    SOFT
                                    HARD
                                    SPACE,P1,P2
                                    *

                DEFAULT = SOFT

                SOFT = PROCESSING OF TITLE CARD RESETS SUB-TITLE
                       WITH EJECT. (NO BANNER PAGE)
                HARD = SAME AS ABOVE, BUT AFTER EJECT A BANNER PAGE
                       CONTAINING THE CONTENTS OF THE TITLE CARD
                       WILL BE PRINTED IN THE CENTER OF THE NEXT
                       PAGE FOLLOWED BY ANOTHER EJECT.
                SPACE= RESET SUB-TITLE AND THEN TREAT AS
                       *SPACE P1,P2* CARD.
                *    = RETURN TO LAST TITLE SELECTION.

            C.  *EJECT*   P1,P2.

                TREAT *EJECT* AS *SPACE P1,P2*.


      2.  **T  OPTION SELECTION
               THE CARD -
**T  EXAMPLE  24/PP PROGRAM NAME+RECALL,18/PARAMETER 1,18/PARAMETER 2
               WOULD GENERATE THE FOLLOWING TABLE PICTURE -
                5         4         3         2         1
 9876543210987654321098765432109876543210987654321098765432109876543210
-------------------------------------------------------------------------
EXAMPLE /PP PROGRAM NAME+RECALL /PARAMETER 1       /PARAMETER 2       /
-------------------------------------------------------------------------

                THE IDENTIFICATION *EXAMPLE* MAY BE OMITTED.
                RESTRICTIONS ON THE IDENTIFICATION LABEL (IF ONE
                IS SPECIFIED) ARE THAT IT MAY NOT BEGIN WITH A
                NUMERAL, MAY NOT BE GREATER THAN EIGHT CHARACTERS
                IN LENGTH, AND MAY NOT CONTAIN EMBEDDED BLANKS
                OR COMMAS.

                EACH TIME A NEW BLOCK OF **T CARDS IS ENCOUNTERED,
                A BIT POSITION HEADER IS LISTED. THIS HEADER IS NOT
                LISTED FOR EACH CONSECUTIVE TABLE CARD OR FOR ANY
                CARD CONTAINING A NON-BLANK CHARACTER IN COLUMN 4 OF
                THE FIRST **T CARD IN A BLOCK.
                CARD FORMAT IS THE SAME AS FOR THE COMPASS *VFD*
                PSEUDO INSTRUCTION, HOWEVER, NO *VFD* MAY BE PRESENT.
                A SLASH */* MUST IMMEDIATELY FOLLOW A BIT COUNT
                FIELD, BUT LEADING SPACES ARE IGNORED.
                ALL BIT COUNTS FOR FIELD WIDTHS MAY BE SPECIFIED IN
                EITHER OCTAL OR DECIMAL. DECIMAL COUNTS ARE ASSUMED
                IN THE ABSENCE OF A POST-RADIX (B) OR (D).
                MAXIMUM PICTURE WIDTH IS 60 BITS.
                A SLASH SEPARATES FIELDS IN THE PICTURE AND THE
```

BIT POSITION IT OCCUPIES IS INCLUDED IN THE FIELD TO ITS LEFT. SINGLE BIT FIELDS ARE LISTED WITHOUT A SLASH FIELD SEPARATOR. ALL TABLE ENTRY DESCRIPTION CARDS WITHIN A **T BLOCK ARE CONSIDERED TO HAVE THE SAME TOTAL NUMBER OF BITS.
FIELD LABELS ARE LEFT JUSTIFIED WITHIN THE FIELD AND WILL BE TRUCATED IF THE LABEL CONTAINS MORE CHARACTERS THAN THE BIT COUNT MINUS 1.
IF THE FOURTH CHARACTER ON THE 1ST **T CARD IS NON-BLANK, NO BIT COUNT HEADER WILL BE PLACED ABOVE THE TABLE ENTRY.

```
**T, EXAMPLE  24/PP PROGRAM NAME+RECALL,18/PARAMETER 1,18/PARAMETER 2
```

WOULD GENERATE THE FOLLOWING TABLE PICTURE -

```
EXAMPLE /PP PROGRAM NAME+RECALL /PARAMETER 1       /PARAMETER 2       /
```

IF THE FOURTH CHARACTER ON SUCEEDING TABLE CARDS IS NON-BLANK NO BIT HEADER OR TOP LINE OF TABLE WILL BE PLACED ON CURRENT TABLE ENTRY.

```
**T   LINE1     24/PP PROGRAM NAME+RECALL,18/PARAMETER,18/PARAMETER
**T,  LINE2     24/TO CALL TO CONTROL PT.,18/1,18/2
```

WOULD GENERATE THE FOLLOWING TABLE PICTURE -

```
                5         4         3         2         1
 98765432109876543210987654321098765432109876543210987654321098765432109876543210
LINE1   /PP PROGRAM NAME+RECALL /PARAMETER          /PARAMETER          /
LINE2   /TO CALL TO CONTROL PT. /1                  /2                  /
```

SINGLE BIT FIELDS WILL BE LISTED WITH A *+* BELOW THE FIELD POSITION. THE ONLY EXCEPTION TO THIS IS THE CASE WHERE ONLY ONE TABLE ENTRY IS LISTED. IN THIS INSTANCE THE *+* WILL BE LISTED BOTH ABOVE AND BELOW THE FIELD POSITION.

```
**T   PPCALL   18/OLLDV,1/O,1/R,4/CP,18/0,18/PARAM
```

WOULD GENERATE THE FOLLOWING TABLE PICTURE -

```
                5         4         3         2         1
 98765432109876543210987654321098765432109876543210987654321098765432109876543210
                        ++
PPCALL  /OLLDV                /ORCP /O                    /PARAM              /
                        ++
```

TO ALLOW FOR THE CENTERING OF A DESCRIPTION WITHIN A FIELD, WITH THE SUBSEQUENT EXTENSION OF THE LINE DESCRIPTION TO MORE THAN 68 POSITIONS, THE DESCRIPTION OF A TABLE LINE MAY BE CONTINUED ON A SECOND TABLE CARD IN THE FOLLOWING MANNER -

```
**T           24/PP PROGRAM NAME+RECALL,18/   PARAMETER 1,
*,T           18/   PARAMETER 2
```

WHICH WILL GENERATE THE FOLLOWING TABLE PICTURE (SIMILAR TO THE FIRST **T EXAMPLE) -

```
          5         4         3         2         1
 9876543210987654321098765432109876543210987654321098765432109876543210
 ------------------------------------------------------------------------
 /PP PROGRAM NAME+RECALL /   PARAMETER 1   /   PARAMETER 2   /
 ------------------------------------------------------------------------
```

C. *COMPASS* PSEUDO-OPS PROCESSED BY *DOCK*.

1. *TITLE*
   THE 1ST TITLE ENCOUNTERED IN THE SOURCE SETS
   MAIN TITLE ON THE LIST OUTPUT UNTIL AN *END* CARD
   IS ENCOUNTERED ON THE SOURCE FILE. ALL SUCCEEDING
   TITLE CARDS RESET THE 2ND HEADER LINE, AS IN
   COMPASS.

2. *TTL*
   RESETS MAIN TITLE LINE WITH NO EJECT.

3. *EJECT*
   DOCK KNOWS 2 FORMS OF THE *EJECT* CARD --
   HARD EJECT,
   AN *EJECT* CARD WITH NO NUMBER SPECIFICATION IN
   COLUMNS 18-30 IS PROCESSED BY *DOCK* THE SAME AS
   COMPASS, A PAGE EJECT.
   SOFT EJECT,
   IF COLUMNS 18-30 CONTAIN NUMERIC DATA, *DOCK* TREATS
   IT AS A SPACE CARD.

4. *SPACE*
   DOCK LIKE *COMPASS* CHECKS COLUMNS 18-30 ON
   CARD TO VERIFY THAT CURRENT SPACE COUNT IS .LT. NUMBER
   OF LINES LEFT ON PAGE.

5. *LIST*
   SAME AS *COMPASS*. ONLY *X* AND *-X* HAVE ANY MEANING
   TO *DOCK*.

6. *CTEXT*/*ENDX* (SAME AS *COMPASS*)
   LIST *X* ON.
   CTEXT CARD TREATED AS A *TITLE* CARD, DOCUMENTATION
   PROCESSED.
   LIST *X* OFF.
   EVERYTHING UNTIL AN ENDING *ENDX* IGNORED.

7. *IDENT*/*END*
   EACH MATCHING *IDENT/END* PAIR IS TREATED AS A UNIT.

D. *FORTRAN* CODE PROCESSING.

1. *PROGRAM*, *BLOCKDATA*, *ENTRY*, *FUNCTION*,
   SUBROUTINE -

PROCESSING LIKE *COMPASS* *IDENT* PROCESSING.
(*FUNCTION* ALSO INCLUDES LOGICAL, INTEGER, REAL,
DOUBLE, AND COMPLEX.)

2. THE ONLY DIFFERENCE BETWEEN *COMPASS* AND *FORTRAN*
DOCUMENTATION IS IN THE FIRST COLUMN
FOR *FORTRAN* TYPE DOCUMENTATION ALL INTERNAL/EXTERNAL
BRACKETED DOCUMENTATION MUST START WITH C OR * IN
COLUMN ONE WITH ALL OTHER COLUMNS THE SAME AS STATED
ABOVE FOR *COMPASS*.

```
IE.
INTERNAL
C*      INTERNAL DOCUMENTATION.
C

EXTERNAL
C**     EXTERNAL DOCUMENTATION.
C

BRACKETED
C***    BRACKETED DOCUMENTATION.
  ANYTHING
  AND AGAIN
C***    END OF BRACKETED DOCUMENTATION.

OVERVIEW
C****   OVERVIEW DOCUMENTATION.
C
```

3. ALL **DOCK CONTROL OPTIONS ARE AVAILABLE AS ABOVE BUT
FIRST COLUMN MUST CONTAIN C OR * .

```
IE.
C*DOCK  *LIST* OFF
```