ROGALGOL ALGOL-60 COMPILER

CONTENTS

1. INTRODUCTION

ROGALGOL IS AN EFFICIENT ALGOL-60 SYSTEM FOR THE PDP8. ITS SPEED IS USUALLY LIMITED BY THE SPEED OF THE FLOATING POINT PACKAGE, WHICH MAKES IT THREE TIMES FASTER THAN OS/8 FORTRAN, OR SIX TIMES FASTER ON A MACHINE WITH EAE. THE COMPILED CODE IS ONE THIRD OF THE LENGTH OF COMPILED FORTRAN, AND THERE IS CONSIDERABLY MORE SPACE AVAILABLE FOR IT, SO THAT PROGRAMS AT LEAST FOUR TIMES AS LONG CAN BE RUN THAN IS POSSIBLE WITH FORTRAN (UNLESS SOME PARTS OF THE PROGRAM ARE OVERLAID ON A MASS STORAGE DEVICE). ROGALGOL IS ALSO FASTER AND MORE COMPACT THAN OS/8 FORTRAN IV, BUT NOT BY SUCH A WIDE MARGIN. DATA STORAGE IS SEPARATE FROM PROGRAM STORAGE, AND WITH AN 8K MACHINE ABOUT 1160 DECIMAL LOCATIONS ARE AVAILABLE. THIS IS ASSIGNED DYNAMICALLY AT RUN TIME SO THAT THERE IS NO WASTAGE - EQUIVALENCING AS IN FORTRAN IS NOT NECESSARY, THE SYSTEM DOES IT FOR YOU. A MECHANISM IS PROVIDED FOR STORING EXTRA DATA IN ANY FREE PROGRAM SPACE. ARRAYS CAN BE PUT INTO A THIRD MEMORY FIELD IF THE MACHINE HAS MORE THAN 8K OF CORE STORE. THE COMPILED CODE CAN ALSO BE PLACED IN A THIRD OR FOURTH MEMORY FIELD, ALLOWING THE SYSTEM TO USE UP TO 16K.

THE MINIMUM HARDWARE IS AN 8K PDP8 WITH TELETYPE, BUT THE SYSTEM CAN USE ANY PERIPHERALS, AND WILL RUN UNDER DISK MONITOR OR OS/8. THE COMPILER TRANSLATES THE ALGOL SOURCE IN A SINGLE PASS INTO SIX-BIT OPERATION CODES AND CONSTANTS. A RUN-TIME PROGRAM READS THE COMPILER OUTPUT DIRECTLY INTO MEMORY, AND CARRIES OUT THE OPERATIONS SPECIFIED BY THE CODES. THE COMPACTNESS OF THE COMPILED CODE IS DUE TO THE FACT THAT TWO INSTRUCTIONS ARE PACKED INTO A SINGLE PDP8 WORD AND THEY ARE MORE POWERFUL INSTRUCTIONS THAN THE MACHINE CODE. THE COMPILER OUTPUT IS IN MACRO LANGUAGE, AND CAN THEREFORE BE ASSEMBLED INTO BINARY. THERE IS NORMALLY NO NEED TO DO THIS, AS THE RUN-TIME SYSTEM CONTAINS ITS OWN LOADER.

THE COMPILER IS ITSELF WRITTEN IN ALGOL. THIS HAS ALLOWED ALMOST ALL THE FEATURES OF ALGOL-60 TO BE IMPLEMENTED EVEN IN SUCH A SMALL MACHINE AND WITH A ONE PASS COMPILER. THE FLEXIBILITY WHICH RESULTS FROM THE FACT THAT THE ALGOL LANGUAGE IS DEFIND RECURSIVLY IS RETAINED IN FULL. THERE IS NO RESTRICTION ON THE COMPLEXITY OF BLOCK STRUCTURE, PROCEDURE DECLARATION OR CALLING, CONDITIONAL STATEMENTS AND EXPRESSIONS OR THE NUMBER OF ARRAY SUBSCRIPTS. PROCEDURE PARAMETERS OF ALL TYPES ARE ALLOWED, BUT VARIABLES CANNOT BE CALLED BY NAME NOR ARRAYS BY VALUE. FOR STATEMENT LISTS ARE NOT RESTRICTED.

THE INPUT/OUTPUT ROUTINES HAVE BEEN ARRANGED IN SUCH A WAY THAT USERS CAN EASILY ADD THEIR OWN AND CHANGE THE STANDARD ONES. THE SYSTEMS DEVICE (IF PRESENT) IS HANDLED BY AN OVERLAY WITH ONLY THREE ENTRY POINTS, MAKING THE ALGOL COMPILER AND RUN-TIME PROGRAMS SYSTEM INDEPENDENT. OVERLAYS ARE AVAILABLE FOR OS/8 AND DISK MONITOR.

2. THE LANGUAGE

ALGOL-60 IS AN INTERNATIONALLY RECOGNISED LANGUAGE. IT HAS TWO ADVANTAGES OVER FORTRAN FOR THE MINI-COMPUTER USER. THE FIRST IS THAT DATA STORAGE IS ASSIGNED DYNAMICALLY AS IT IS NEEDED, AND IS RETURNED WHEN IT IS NO LONGER REQUIRED. THIS MEANS THAT THERE IS NO NEED TO EQUIVALENCE ARRAYS - IT IS DONE AUTOMATICALLY. THE SECOND ADVANTAGE IS THAT THE GREATER POWER OF THE LANGUAGE ALLOWS THE SAME TASK TO BE ACCOMPLISHED WITH FEWER STATEMENTS AND MUCH GREATER ELEGANCE.

THIS DOCUMENT ASSUMES A KNOWLEDGE OF ALGOL. FOR THOSE REARED ON FORTRAN AND THOSE WHO HAVE ONLY USED THE ALGOL LANGUAGE IN AN ELEMEMTARY WAY, THE FOLLOWING EXAMPLES ILLUSTRATE THE FLEXIBILITY AND POWER OF ITS STATEMENTS.

1. THE RECURSIVE ABILITY OF ALGOL IS COMMONLY ILLUSTRATED BY THE FACTORIAL FUNCTION:

```
FACTORIAL:='IF' N=Ø 'THEN' Ø 'ELSE' N*FACTORIAL(N-1);
```

THIS IS NOT A VERY GOOD EXAMPLE, BECAUSE THE OBVIOUS WAY OF COMPUTING FACTORIAL DOES NOT INVOLVE RECURSION. THE EXAMPLE BELOW IS A PROGRAM TO CONVERT NON-PARITY TAPES TO PARITY TAPES. A NON-RECURSIVE PARITY PROCEDURE WOULD BE MUCH LONGER. THE PROGRAM STOPS WHEN CTRL/Z IS READ. THE EXPRESSION (X-X%2*2) HAS THE VALUE 1 IF X IS ODD, OTHERWISE Ø.

```
'BEGIN''INTEGER' I;
'INTEGER''PROCEDURE' PARITY(X);'VALUE' X;'INTEGER' X;
PARITY:='IF' X=Ø 'THEN' Ø
          'ELSE' ABS(PARITY(X%2)-(X-X%2*2));

'FOR' I:=CHIN(2) 'WHILE' I#154 'DO'
          CHOUT(2,I-128*PARITY(I));
'END'$
```

2. ANYWHERE THAT AN ARITHMETIC VALUE IS NEEDED, ANY ARITHMETIC EXPRESSION CAN BE USED. TYPE CONVERSION IS DONE AUTOMATICALLY.

```
ARRAY[COS(X+SIN(Y)-2),1,I]:='IF' X#Ø 'THEN' 25 'ELSE'
          COS(ARRAY[I,I+2.3*SIGN(Y+.5),1Ø])-Z+I;
```

3. THE LOOP STATEMENT (FORTRAN DO) IS A LIST OF ELEMENTS WHICH MAY BE A SINGLE VALUE, A STEP ELEMENT (INCREMENTING OR DECREMENTING) OR AN ASSIGNMENT TO BE MADE REPEATEDLY UNTIL A BOOLEAN EXPRESSION IS FALSE.

```
'FOR' A:=Ø.1, 11 'STEP' -Ø.15 'UNTIL' 1, 25.3,
A*1.3 'WHILE' A*Z<1ØØØ,'IF' X<2Ø 'THEN' 1ØØØ 'ELSE' 5ØØ 'DO'
```

IN FORTRAN, IT WOULD BE NECESSARY TO DELCARE A SUBROUTINE TO EXECUTE THE EQUIVALENT OF THE ALGOL 'FOR' CONTROLLED STATEMENT. THE FORTRAN EQUIVALENT IS:

```
            A=0.1
            CALL SUB
            A=11.
1           IF (A-1.)2,9,9
9           CALL SUB
            A:=A-.15
            GOTO 1
2           A=25.3
            CALL SUB
3           A=A*1.3
            IF (A*Z-1000.)4,5,5
4           CALL SUB
            GOTO 3
5           IF (X-20)6,7,7
6           A=1000.
            GOTO 8
7           A=500.
8           CALL SUB
```

4. IT IS EFFICIENT TO DECLARE A FUNTION TO DO A SMALL CALCULATION THAT OCCURS SEVERAL TIMES.

```
'BEGIN'
'REAL''PROCEDURE' SUM(J); 'VALUE' J; 'REAL' J;
SUM:=J+2*J↑2+3*J↑3;

A:=0;
'FOR' I:=1 'STEP' 1 'UNTIL' 10 'DO'
          A:=A+SUM(ARRAY[I])+SUM(I+ARRAY[I+2]);
'END' LOCAL BLOCK;
```

5. COMPLEX LOGICAL CONSTRUCTIONS ARE HANDLED MUCH MORE EASILY THAN IN FORTRAN, IN WHICH THE FOLLOWING WOULD BE DONE BE A NUMBER OF LABELLED STATEMENTS AND CONDITIONAL JUMPS.

```
'IF' X≠0 'THEN'
'BEGIN''IF' X=5 'THEN' Y:='IF' Z=3 'THEN' 2 'ELSE' 3;
          'IF' X=6 'THEN'
          'BEGIN' WRITE(1,Y); SKIP(1)
          'END''ELSE''IF' X=7 'THEN' Y:=Y↑2
          'ELSE' Y:=Y+X
'END' THE CASE THAT X≠0
'ELSE'
'BEGIN''REAL' TEMP;
TEMP:=Y*Z; 'IF' TEMP>20 'THEN' X:=TEMP
          'ELSE' Y:=TEMP-5;
'END' THE CASE THAT X=0;
```

THE COMPILER TRANSLATES A SOURCE PROGRAM WRITTEN IN ALGOL-60, SUBJECT TO THE RESTRICTIONS LISTED IN SECTION 2.5. IT MAY BE ASSUMED THAT ANY FACILITY NOT MENTIONED THERE IS AVAILABLE. THE MAIN RESTRICTIONS ARE (1) THERE ARE NO SWITCHES AND (2) PROCEDURE PARAMETERS OF TYPE REAL, INTEGER AND BOOLEAN CANNOT BE CALLED BY NAME, WHILE ARRAYS CANNOT BE CALLED BY VALUE.

VARIABLE ADDRESSES ARE 6 BITS. THIS LIMITS THE NUMBER OF

VARIABLES IN SCOPE AT ONCE IN ANY PROCEDURE OR THE MAIN PROGRAM TO 63. A FEW ARE USED INTERNALLY, LEAVING 62 IN THE MAIN PROGRAM AND 60 IN PROCEDURES FOR THE ALGOL. ARRAYS COUNT AS ONE VARIABLE. 'FOR' STATEMENTS REQUIRE ONE VARIABLE SLOT FOR EACH DEPTH OF 'FOR' STATEMENT USE - 'FOR' STATEMENTS NOT EMBEDDED IN OTHERS WILL USE THE SAME LOCATION. THE TOTAL NUMBER OF VARIABLES IN THE PROGRAM IS NOT LIMITED, EXCEPT THAT THERE MAY NOT BE MORE THAN 140 IDENTIFIERS IN SCOPE AT ONCE AND THERE MAY NOT BE MORE THAN 63 PROCEDURES IN TOTAL. THE TOTAL OF LABELS AND PROCEDURES IN SCOPE PLUS UNRESOLVED FORWARD REFERENCES IS LIMITED TO 100 AT ANY INSTANT DURING COMPILATION, BUT NOT IN TOTAL.

2.1 PROGRAM FORMAT

THE SOURCE PROGRAM IS TYPED IN FREE FORMAT. EXCEPT WITHIN STRINGS, ALL EDITING CHARACTERS AND BLANK TAPE AND RUBOUT ARE IGNORED. WITHIN STRINGS, SPACE IS COPIED AND TAB IS CONVERTED TO SPACE.

LONG BASIC SYMBOLS ARE WRITTEN ENCLOSED IN SINGLE QUOTES, FOR EXAMPLE 'IF','BEGIN'. INTEGER DIVISION IS DENOTED BY %, NOT EQUAL TO BY #, MULTIPLY BY *, LESS THAN OR EQUAL TO BY <= AND GREATER THAN OR EQUAL TO BY >=. ALL OTHER SYMBOLS ARE REPRESENTED BY THEIR TELETYPE EQUIVALENT. ALL LETTERS ARE CAPITALS. THERE MUST BE A $ FOLLOWING THE LAST 'END', WITH AN END COMMENT BETWEEN THEN IF DESIRED.

2.2. INPUT/OUTPUT ROUTINES.

THE VARIOUS DEVICES ARE ADDRESSED BY A DEVICE NUMBER, CALLED DEV IN THE FOLLOWING EXAMPLES. IN THE STANDARD CONFIGURATION, THESE ARE TELETYPE (DEVICE 1), HIGH SPEED READER/PUNCH (DEVICE 2) AND SYSTEM FILES (DEVICE 3). DEVICE 0 WILL FAIL IN AN INPUT ROUTINE, BUT IN AN OUTPUT ROUTINE THE EFFECT IS THAT THERE IS NO OUTPUT AT ALL. OTHER DEVICES MAY BE CATERED FOR BY WRITING MACHINE CODE INSTRUCTIONS IN THE ALGOL PROGRAM, OR BY SETTING IN THE DEVICE LIST THE ADDRESS OF A MACHINE CODE SUBROUTINE TO HANDLE IT. THE METHOD IS DESCRIBED IN SECTION 5. THE MONITOR AND OS/8 OVERLAYS USE THE DEVICE THREE LINKING ROUTINES, AS THEY ARE IN FIELD 0. THIS WILL BE DESCRIBED IN SECTION 6.

THE FOLLOWING PROCEDURES ARE BUILT INTO THE SYSTEM.

'PROCEDURE' SKIP(DEV); OUTPUT A NEWLINE TO DEV.

'INTEGER''PROCEDURE' CHIN(DEV); READ NEXT CHARACTER FROM DEV, THE RESULT OF THE PROCEDURE IS THE VALUE OF THE CHARACTER.

'REAL''PROCEDURE' READ(DEV); READ A FLOATING POINT OR INTEGER NUMBER FROM DEV. ALL CHARCTERS OCCURING BEFORE THE START OF A NUMBER ARE IGNORED. FREE FORMAT, E IS USED FOR DECIMAL

EXPONENTIATION.

'PROCEDURE' IOC(PARAM); INITIALLY, INPUT FROM THE TELETYPE KEYBOARD IS NOT ECHOED ON THE PRINTER. IF PARAM#Ø, THIS SWITCHES ON THE ECHO. IOC(Ø); WILL SWITCH IT OFF AGAIN. IOC(-1) WILL SEND AN RFC TO THE HIGH SPEED READER, TO INITIALISE THE HARDWARE. THE PROGRAM CAN BE MADE TO READ TAPE IN SECTIONS BY MAKING IT FIRST WAIT FOT A CHARACTER FROM THE TELETYPE (AS A CUE TO START), THEN DOING IOC(-1) BEFORE READING FROM DEVICE 2.

'PROCEDURE' TEXT(DEV,"STRING"); OUTPUT A STRING TO DEV. BETWEEN STRING QUOTES, PRINTED CHARACTERS ARE REPRODUCED AS THEY ARE TYPED, THAT IS WORDS ENCLOSED BY SINGLE QUOTES ARE NOT CONVERTED TO LONG BASIC SYMBOLS. SPACES ARE COPIED, AND THE TAB CHARACTER IS CONVERTED TO A SINGLE SPACE. THE STRING MAY ALSO BE A STRING PARAMETER OF THE PROCEDURE IN WHICH TEXT IS CALLED, IN WHICH CASE IT IS NOT ENCLOSED IN STRING QUOTES.

'PROCEDURE' RWRITE(DEV, VALUE); PRINT VALUE IN FLOATING POINT FORMAT ON DEV. THE FLOATING POINT PACKAGE OUTPUT ROUTINE IS USED. FORMATTED OUTPUT CAN BE OBTAINED BY INCORPORATING THE ALGOL PRINT PROCEDURE PROVIDED SEPARATELY OR A SIMILAR ONE IN THE ALGOL PROGRAM.

'PROCEDURE' WRITE(DEV,IVAL); PRINT IVAL AS AN INTEGER ON DEV. NO NON-SIGNIFICANT CHARACTERS ARE PRINTED.

'PROCEDURE' CHOUT(DEV,CHAR); OUTPUTS A SINGLE CHARACTER TO DEV. CHAR IS IN DECIMAL.

'PROCEDURE' DISK(PARAM); INITIALISE SYSTEM DEVICE FILES. PARAM IS Ø TO 4. THIS ROUTINE WILL BE DISCUSSED IN GREATER DETAIL IN SECTION 6.

DISK(Ø); START AGAIN AT THE BEGINNING OF THE INPUT FILE, THAT IS REWIND IT.
DISK(1); OPEN AN INPUT FILE. ONLY ONE MAY BE OPEN AT ONCE, ANY PREVIOUSLY OPEN FILE IS AUTOMATICALLY CLOSED.
DISK(2); OPEN AN OUTPUT FILE.
DISK(3); CLOSE THE OUTPUT FILE AND OPEN IT FOR READING. IN THE MONITOR ROUTINES, THERE MUST ALSO HAVE BEEN A FILE OPEN FOR READING, WHICH IS CLOSED AND OPENED FOR WRITING. DISK(1) AND DISK(2) USE THE COMMAND DECODERS OF MONITOR AND OS/8.
FILES MAY BE INITIALISED AT ANY TIME, NOT JUST WHEN A PROGRAM IS FIRST STARTED.

DEVICES 1 AND 2 DO NOT HAVE TO BE OPENED OR CLOSED. SYSTEMS DEVICE INPUT FILES DO NOT HAVE TO BE CLOSED, BUT OUTPUT FILES MUST BE CLOSED BY SENDING CTRL/Z TO THEM, BY DOING CHOUT(3,154). THIS MUST ALWAYS BE DONE, OTHERWISE THE LAST BUFFER WILL NOT BE TRANSFERRED, AND IN THE CASE OF THE OS/8 SYSTEM THE TENTATIVE FILE WILL BE LOST ALTOGETHER.

2.3 STANDARD FUNCTIONS

THE FULL SET IS AVAILABLE, THAT IS
SQRT, SIN, COS, ARCTAN, EXP, LN, SIGN, ENTIER, ABS.

2.4 MACHINE CODE STATEMENTS

THE SYMBOLS 'TRANS' 'ALGOL' ACT AS STATEMENT BRACKETS IN THE SAME WAY AS 'BEGIN' 'END'. EVERYTHING BETWEEN THEM IS COPIED TO THE OUTPUT DEVICE, EXCEPT EDITING CHARACTERS. SPACE IS TRANSLITERATED, HOWEVER. THIS ALLOWS MACHINE CODE TO BE WRITTEN BETWEEN THE 'TRANS' AND THE 'ALGOL'. A CODE INSTRUCTION TO LEAVE THE INTERPRETER IS GENERATED AUTOMATICALLY AT THE 'TRANS', AND A SUBROUTINE JUMP TO RE-ENTER IT AND CHAGE THE RADIX FOR NUMBERS BACK TO DECIMAL AT THE 'ALGOL'. REMEMBER TO DO "OCTAL;" IF YOU WANT TO WRITE ADDRESSES ETC. IN OCTAL. REMEMBER ALSO THAT CARRIAGE RETURN IS NOT COPIED, SO ALL MACHINE CODE INSTRUCTIONS MUST BE TERMINATED WITH ";". THE COMPILER WILL CHANGE ALL SEMICOLONS TO CR/LF, BECAUSE OS/8 CANNOT DEAL WITH LINES OF INDEFINITE LENGTH.

IF MACHINE CODE IS INCLUDED IN AN ALGOL PROGRAM, THE ALGOL COMPILER OUTPUT MUST BE FUTHER COMPILED INTO BINARY BEFORE IT CAN BE RUN. PAL MAY BE USED IF THE ALGOL SOURCE CONTAINS NO FLOATING POINT LITERALS, OTHERWISE MACRO MUST BE USED.

LOCATION 22 (OCTAL) IN FIELD 1 CONTAINS THE ADDRESS OF THE START OF THE VARIABLES OF THE CURRENT PROCEDURE. THE RESULT OF A TYPE PROCEDURE IS STORED IN VARIABLE 3, AND PARAMETERS ARE AT 4 UPWARDS. SUPPOSE WE WANT THE ADDRESS OF VARIABLE X, AND THE CONTENT OF LOCATION 22 IS Y. THEN THE ADDRESS OF THE MOST SIGNIFICANT OF THE 3 WORDS OF X IS GIVEN BY

Y+3*(X-1).

INTEGERS ARE STORED IN THE LEAST SIGNIFICANT WORD, THAT IS AT AN ADDRESS 2 GREATER THAN THE ONE CALCULATED AS ABOVE. ALL VARIABLES ARE IN FIELD ONE. THE CODE YOU WRITE WILL ALWAYS BE IN FIELD Ø.
LOCATIONS Ø-27 IN FIELD Ø ARE FREE.

THE PROGRAM SIZE PRINTED BY THE COMPILER (WHICH IS DECIMAL) DOES NOT INCLUDE ANY WORDS OF MACHINE CODE INTRODUCED BY THIS FACILITY.

NOTE THAT THE INSTRUCTIONS MAY START ANYWHERE ON A PAGE, SO IF MACRO FAILS PE, YOU MUST WRITE CODE TO START ON A NEW PAGE. EXAMPLE:

```
'TRANS' OCTAL; JMP I .+1; START;
PAGE;
```

```
START, TAD 10; . . . . . .
'ALGOL';
```

SHOULD THE "JMP I .+1;" TURN OUT TO BE IN THE LAST LOCATION OF A PAGE (FAIL ILLEGAL INDIRECT), REPLACE IT WITH "NOP".

THE ADDRESS OF THE NEXT FREE LOCATION IN FIELD Ø IS HELD IN ØØ175, AND IS ALSO EQUAL TO THE SYMBOL V, WHICH IS DEFINED BY THE ALGOL COMPILER OUTPUT.

2.5 RESTRICTIONS AND HOW TO GET AROUND THEM.

MOST OF THESE POINTS WILL APPLY INFREQUENTLY IN NORMAL ALGOL USAGE. THE BRACKETED NUMBERS REFER TO THE CAUSE OF THE RESTRICTION, (1) MEANS IT IS DUE TO THE COMPILER OPERATING IN ONE PASS AND (2) THAT THE PDP8 IS TOO SMALL TO COMPILE PROGRAMS USING THIS FEATURE UNDER THE PRESENT SYSTEM.

1. (1) RELATIONAL BOOLEAN PRIMARIES SHOULD NOT START WITH (. THE REASON IS THAT THIS SYMBOL (AND 'IF') ARE LEGAL AT THE START OF AN ARITHMETIC OR BOOLEAN EXPRESSION, AND THE COMPILER ASSUMES THEY START A BOOLEAN IF THE CONTEXT IS SUCH THAT THE FINAL RESULT OF THE EXPRESSION MUST BE BOOLEAN. START THE RELATION WITH A +, OR REARRANGE IT. EXAMPLES:

```
'IF' (I-5)=25 'THEN';
BOOL:=('IF' B 'THEN' 5 'ELSE' 6)>X;      WILL NOT COMPILE.
```

THE FOLLOWING WILL COMPILE:

```
'IF' I-5=25 'THEN'
'IF' +(I-5)=25 'THEN'
'IF' 25=(I-5) 'THEN'
BOOL:=+('IF' B 'THEN' 5 'ELSE' 6)>X;
```

2. (2) NO SWITCHES.
A CONDITIONAL STATEMENT CAN OFTEN DO THE SAME JOB, BUT IT IS POSSIBLE TO WRITE A PROCEDURE WHICH ACTS AS A SWITCH. FOR EXAMPLE, 'SWITCH' S1:=L1,L2,S2(3);
IS EQUIVALENT TO

```
'PROCEDURE' S1(INDEX); 'VALUE' INDEX; 'INTEGER' INDEX;
'IF' INDEX=1 'THEN' 'GOTO' L1 'ELSE' 'IF' INDEX=2 'THEN'
'GOTO' L2 'ELSE' 'IF' INDEX=3 'THEN' S2(3);
```

IN THIS EXAMPLE, S2 WOULD BE ANOTHER "SWITCH PROCEDURE". INSTEAD OF 'GOTO' S1(I);, S1(I); WOULD BE WRITTEN.

3. (1) NO MULTIPLE ASSIGNMENTS.

4. (1). IN 'FOR' STATEMENTS, THE CONTROLLED VARIABLE MUST BE UNSUBSCRIPTED.

5. THE RESULT OF AN ↑ OPERATION IS ALWAYS REAL. ACCORDING TO THE ALGOL 60 REPORT, THE TYPE OF THE RESULT OF AN INTEGER↑INTEGER OPERATION IS NOT KNOWN UNTIL RUN TIME, AS IT DEPENDS ON THE SIGN OF THE SECOND INTEGER.

6. (2) NO OWN VARIABLES.
DECLARE THE VARIABLES IN THE OUTER BLOCK. THIS HAS EXACTLY THE SAME EFFECT, EXCEPT IN THE CASE OF OWN ARRAYS WITH DYNAMIC BOUNDS, OR IF THE IDENTIFIERS ARE REDECLARED IN AN INNER BLOCK.

7. (1) THE COMPILER MUST BE ABLE TO INFER THE TYPE OF A PROCEDURE ACTUAL PARAMETER, AT THE TIME IT IS COMPILED.

IN GENERAL, WHEN A PROCEDURE IS CALLED, NEITHER THE TYPE OF THE ACTUAL NOR FORMAL PARAMETER WILL BE KNOWN. SUPPOSE THE ACTUAL PARAMETER IS AN UNDECLARED IDENTIFIER, AND THE PROCEDURE BEING CALLED HAS NOT BEEN DECLARED EITHER. THE UNDECLARED IDENTIFIER MAY BE A TYPE PROCEDURE, BUT EVEN IF THIS IS KNOWN THE COMPILER DOES NOT KNOW WHETHER TO SET ITS ADDRESS AS THE PARAMETER (IF THE FORMAL IS A PROCEDURE) OR WHETHER TO ENTER IT AND WORK OUT ITS VALUE (IF THE FORMAL IS A VARIABLE). OBVIOUSLY, IN A ONE PASS COMPILER, THE TYPE OF AN ACTUAL PARAMETER MUST BE KNOWN AT THE TIME IT IS COMPILED.

ROGALGOL USES THE FOLLOWING CONVENTIONS TO GIVE THE COMPILER THIS INFORMATION.

(A) THE COMPILER IS INFORMED IF THE ACTUAL PARAMETER IS A LABEL OR PROCEDURE BY PREFIXING THE NAME OF THE LABEL OR PROCEDURE WITH A TYPE DECLARATION. EXAMPLE:

```
FRED('LABEL' L1, 'REAL''PROCEDURE' RP, 2.345);
```

DESIGNATIONAL EXPRESSIONS ARE ALLOWED AS PROCEDURE PARAMETERS, FOR EXAMPLE:

```
FRED('LABEL''IF' X=5 'THEN' L1 'ELSE' L2);
```

THIS PREFIXING IS DONE ONLY WHEN THE PROCEDURE IS CALLED, THE DECLARATION OF A PROCEDURE IS STANDARD ALGOL-60.

(B) IF THE PARAMETER IS NOT SO PREFIXED, THE TYPE IS TAKEN TO BE REAL, INTEGER, BOOLEAN, STRING OR AN ARRAY, AS DEDUCED FROM THE FIRST SYMBOL OR IDENTIFIER OF THE PARAMETER.

(C) AN IDENTIFIER OCCURING AT THE START OF A PROCEDURE ACTUAL PARAMETER MUST HAVE BEEN DECLARED. THE TYPE OF THE PARAMETER WILL BE THE TYPE OF THE IDENTIFIER, UNLESS IT IS AN ARRAY NAME FOLLOWED BY [. IF THE IDENTIFIER STARTS AN ARITHMETIC EXPRESSION, THE TYPE MAY CHANGE FROM INTEGER TO REAL, BUT THE COMPILER CANNOT CHANGE THE TYPE TO BOOLEAN IF A RELATIONAL EXPRESSION IS USED.

(D) ANY OTHER SYMBOL WILL CAUSE THE COMPILER TO ASSUME AN ARITHMETIC EXPRESSION FOLLOWS, EXCEPT THE SYMBOLS 'NOT', 'TRUE' AND 'FALSE', IN WHICH CASE IT IS ASSUMED TO BE A BOOLEAN EXPRESSION, OR WITH ", IN WHICH CASE IT IS A STRING PARAMETER.

YOU CAN FORCE THE COMPILER TO ACCEPT ANY PARAMETER YOU WANT BY STICKING TO THE LETTER OF THIS RESTRICTION, AND GIVING IT A "CLUE". E.G. TO USE A CONDITIONAL BOOLEAN EXPRESSION YOU COULD USE 'FALSE ''OR' (REQUIRED EXPRESSION), AND TO USE AN UNDECLARED REAL PROCEDURE Ø.Ø+NAME.
THE FIRST OF THESE WILL ADD ONLY ONE WORD TO THE CODE,

THE SECOND FOUR OR 4.5 WORDS.

8. (1,2) PROCEDURE PARAMETERS OF TYPE 'INTEGER', 'BOOLEAN' AND 'REAL' MUST BE CALLED BY VALUE, AND ALL OTHERS BY NAME.

IN MOST CASES THE PARAMETER CALLED BY NAME WOULD BE USED FOR STORING A RESULT, I.E. IT IS AN OUTPUT NAME ASSIGNED TO WITHIN THE PROCEDURE, AND THE VALUE IS PICKED UP AFTER THE PROCEDURE IS FINISHED. THIS USE CAN BE SIMULATED BY USING VARIABLES DECLARED OUTSIDE THE PROCEDURE FOR THE OUTPUT PARAMETERS. MORE COMPLICATED USES CAN OFTEN BE SIMULATED BY USING AN ARRAY WITH ONE ELEMENT INSTEAD OF A SIMPLE VARIABLE CALLED BY NAME.

WHERE A NAME PARAMETER IS USED TO SUPPY INFORMATION TO THE PROCEDURE, IT IS BECAUSE SIDE EFFECTS ARE REQUIRED TO MAKE USE OF (FOR EXAMPLE) JENSEN'S DEVICE. IN THESE CASES A FULL COMPILER WOULD GENERATE AS THE ACTUAL PARAMETER THE ADDRESS OF THE ROUTINE WHICH MUST BE OBEYED TO WORK OUT THE PARAMETER. THIS USE CAN BE SIMULATED BY SUPPYING THE NAME OF A TYPE PROCEDURE, WHICH CAN BE DECLARED LOCALLY. SUPPOSE THE CODE IS REQUIRED TO HAVE THE EFFECT:
P(A[I]), WHERE THE PARAMETER IS CALLED BY NAME. IF THE PROCEDURE NEEDS TO FETCH THE PARAMETER ONLY, THIS CAN BE WRITTEN:

```
'BEGIN''REAL''PROCEDURE' DUMMY; DUMMY:=A[I];
P('REAL''PROCEDURE' DUMMY);
'END';
```

IN THE PROCEDURE BODY OF P, THE NAME OF THE FORMAL PROCEDURE REPLACED IN THE CALL BY DUMMY WOULD BE USED IN JUST THE SAME WAY AS THE NAME OF A REAL PARAMETER CALLED BY NAME WOULD BE USED.

9. (2) STANDARD PROCEDURE NAMES CANNOT BE USED AS FORMAL PROCEDURES. FOR EXAMPLE P('REAL''PROCEDURE' COS) IS NOT ALLOWED. A DUMMY PROCEDURE MUST BE DECLARED TO EVALUATE COS, FOR EXAMPLE:

```
'REAL''PROCEDURE' COSINE(X); 'VALUE'X; 'REAL'X; COSINE:=COS(X);
```

P('REAL''PROCEDURE' COSINE) IS ALLOWED.
OF COURSE, STANDARD FUNCTIONS MAY BE USED WHEN THE PROCEDURE PARAMETER IS A NUMERICAL VALUE, THE RESTRICTION APPLIES ONLY WHEN THE FORMAL PARAMETER IS A PROCEDURE.
FOR EXAMPLE, SIN(COS(1)) IS ALLOWED.

10. (1) VARIABLES MUST BE DECLARED BEFORE THEY ARE USED. IF THEY ARE NOT, THE COMPILER WILL ASSUME THAT THE UNDECLARED IDENTIFIER IS A FUNCTION (TYPE PROCEDURE) OR ORDINARY PROCEDURE, ACCORDING TO CONTEXT. THEREFORE, IF A NON-LOCAL VARIABLE IS USED INSIDE A PROCEDURE, THE DECLARATION OF THE VARIABLE MUST COME BEFORE THE DECLARATION OF THE PROCEDURE. IF, WHEN AN ASSUMED PROCEDURE IS ACTUALLY DECLARED, IT IS NOT OF THE SAME TYPE AS THAT EXPECTED EARLIER, A FAIL 2 WILL OCCUR, EVEN IF A PROCEDURE OF THE TYPE ACTUALLY DECLARED COULD HAVE BEEN LEGALLY USED.

11. ONLY THE FIRST 6 CHARACTERS OF AN IDENTIFIER ARE SIGNIFICANT. THIS IS BECAUSE OF THE LIMITED SPACE AVAILABLE

FOR THE COMPILER'S IDENTIFIER TABLES.

12. (2) THE BOOLEAN OPERATOR 'IMPLIES' IS NOT AVAILABLE. A 'IMPLIES' B IS THE SAME AS 'NOT'(A 'AND' 'NOT'B).

13. (1) IN THE FOLLOWING EXAMPLE, THE 'GOTO' L; STATAMENT WILL REFER TO THE FORMAL LABEL, NOT THE LABEL DECLARED AFTER THE 'GOTO'. THIS IS NOT CORRECT ALGOL. THE REASON IS THAT JUMPS TO FORMALS ARE HANDLED DIFFERENTLY FROM ORDINARY JUMPS, AND THE COMPILER MUST DECIDE AT THE 'GOTO' WHICH TYPE TO COMPILE. FORMAL AND LOCAL LABELS MAY BE MIXED IN A DESIGNATIONAL EXPRESSION (SEE * IN THE EXAMPLE).

```
'PROCEDURE' XXX(L); 'LABEL' L;
'BEGIN' S1; S2;
          'GOTO' L;
          S3;
          'GOTO''IF' B 'THEN' L 'ELSE' L2; 'COMMENT' *;
L:        S4;
L2:       S5;
'END';
```

THE SAME APPLIES TO FORMAL PROCEDURES.

14. (2) NO INTEGER LABELS. MOST COMPILERS HAVE THIS RESTRICTION.

15. (2) PARAMETER DELIMITERS OF THE TYPE ")STRING:(" ARE NOT ALLOWED. COMMAS SHOULD BE USED.

16. FUNCTION DESIGNATORS CANNOT BE USED AS A PROCEDURE STATEMENT, SO ; COS(X); IS ILLEGAL. THIS COULD EASILY BE IMPLEMENTED BUT AN EXTRA VARIABLE WOULD BE NEEDED IN THE MAIN PROGRAM. USERS CAN DECLARE THEIR OWN DUMMY VARIABLE TO ASSIGN TO IF THEY WANT TO THROW AWAY THE RESULT OF A FUNCTION.

3. THE COMPILER

3.1 COMPILER OUTPUT FORMAT

THIS IS NOT REQUIRED INFORMATION FOR USING THE COMPILER.

THE COMPILER OUTPUT IS IN ASCII FORMAT, AND IS COMPATIBLE WITH MACRO. IT CONSISTS OF FOUR SORTS OF ITEM.
(1) 6 BIT INSTRUCTION CODES. THESE ARE PACKED TWO TO A WORD, AND ARE OUTPUT AS SIGNED DECIMAL NUMBERS.
(2) LABEL ADDRESSES. THESE ARE THE LETTER L FOLLOWED BY A DECIMAL NUMBER. THE LOADER REPLACES THESE SYMBOLIC ADDRESSES BY THEIR BINARY EQUIVALENTS.
(3) FLOATING POINT LITERALS. THESE HAVE THE FORM FLTG;-1.23E-4;DECIMAL;. IF NO FLOATING POINT LITERALS (I.E. ONES CONTAINING . OR E) ARE WRITTEN IN THE ALGOL SOURCE, THE COMPILER OUTPUT IS ALSO COMPATIBLE WITH PAL.
(4) LABEL DEFINITIONS. THESE ARE EITHER A LABEL (AS (2))

FOLLOWED BY A COMMA, OR ELSE A DEFINITION OF THE FORM L111=X. X MAY BE A CONSTANT OR ANOTHER PREVIOUSLY DEFINED LABEL.
EACH ITEM IS OUTPUT ON A NEW LINE.

IT IS NOT NORMALLY NECESSARY TO ASSEMBLE THE ALGOL COMPILER OUTPUT INTO BINARY, BECAUSE THE LOADER IN THE RUN-TIME SYSTEM CAN PUT IT STRAIGHT INTO MEMORY. HOWEVER, IT MUST BE COMPILED IF THE ALGOL SOURCE CONTAINS MACHINE CODE INSTRUCTIONS (SECTION 2.4).

3.2 RUNNING THE COMPILER

THE COMPILER IS A SINGLE BINARY TAPE. IT CAN BE LOADED BY ANY BINARY LOADER, AS THE LAST PAGE OF EACH MEMORY FIELD IS UNUSED. START THE COMPILER AT LOCATION 00200. A HEADING WILL BE TYPED, FOLLOWED BY THE QUESTION OUTPUT- . ANSWER WITH AN INTEGER BETWEEN 0 AND 3. (BUT SEE ALSO A LATER PARAGRAPH ABOUT THE SYMBOL TABLE). THIS IS THE DEVICE NUMBER FOR OUTPUT. IF DEVICE 3 IS SPECIFIED, THE COMPILER WILL ASK FOR FILENAMES AS DESCRIBED IN SECTION 6. THE COMPILER WILL THEN TYPE INPUT-, TYPE IN THE INPUT DEVICE NUMBER. IF YOU ARE COMPILING FROM THE HIGH SPEED READER, THE TAPE SHOULD BE LOADED BEFORE THE COMPILER IS STARTED.

AT THE END OF COMPILATION, THE SIZE OF THE COMPILED CODE IN DECIMAL WILL BE TYPED. THE CODE STARTS AT 00202 SO THIS MUST BE ADDED TO OBTAIN THE ADDRESS OF THE NEXT FREE LOCATION IN FIELD 0. ADDITIONALLY, THE NUMBER OF WORDS OF MACHINE CODE (IF ANY) MUST BE ADDED, SINCE THE COMPILER DOES NOT COUNT THEM.
A NEGATIVE SIZE OF PROGRAM MEANS THAT THE SIZE IS GREATER THAN 2047. THE CORRECT SIZE IS 4096+(SIZE PRINTED).

THE COMPILER MAY BE INSTRUCTED TO PRINT ITS SYMBOL TABLE BY GIVING AN OUTPUT DEVICE NUMBER GREATER THEN 9. THE UNITS DIGIT WILL BE TAKEN AS THE ACTUAL DEVICE NUMBER. OUTPUT-12 WILL SEND THE OUTPUT TO THE HIGH SPEED PUNCH AND CAUSE THE TABLE TO BE PRINTED ON THE TELETYPE. THE TABLE IS A LIST OF DECLARED IDENTIFIERS, EACH ONE FOLLOWED BY 3 NUMBERS GIVING INFORMATION ABOUT IT.

THE FIRST IS EITHER THE VARIABLE NUMBER, GIVING THE POSITION OF THE VARIABLE IN THE STACK RELATIVE TO THE POINTER IN 10022 (SEE 2.4), OR IT IS THE LABEL NUMBER ASSIGNED TO A LABEL OR PROCEDURE.

THE SECOND NUMBER IS THE PROCEDURE NUMBER OF THE ENCLOSING PROCEDURE IN WHICH THE IDENTIFIER IS DECLARED. THE MAIN PROGRAM IS 0, AND PROCEDURES ARE NUMBERED SERIALLY AS THEY ARE ENCOUNTERED, REGARDLESS OF DEPTH OF DECLARATION. AS AN EXCEPTION, THE ACTUAL NUMBER OF A PROCEDURE IS PRINTED, INSTEAD OF THE NUMBER OF THE ENCLOSING PROCEDURE.

THE THIRD NUMBER IS THE TYPE OF THE IDENTIFIER.

Ø PROCEDURE FORMAL PARAMETER (TYPE NOT KNOWN YET)
1 REAL
2 INTEGER
3 BOOLEAN
5 REAL ARRAY
6 INTEGER ARRAY
7 BOOLEAN ARRAY
1Ø PROCEDURE
11 REAL PROCEDURE
12 INTEGER PROCEDURE
13 BOOLEAN PROCEDURE

WHEN THE COMPILER HAS FINISHED, ↑ WILL BE TYPED. THE COMPILATION CAN BE INTERRUPTED BY TYPING CTRL/S, WHICH WILL ALSO LEAD TO ↑. CTRL/P WILL START THE COMPILER AFRESH, CTRL/C WILL RETURN TO MONITOR. CTRL/P AND CTRL/C CAN BE TYPED AT ANY TIME.

3.3 COMPILER ERROR MESSAGES

FAIL X ON LINE Y IDENT Z SYMBOL S.

X IS THE FAILURE NUMBER (SEE BELOW), Y THE LINE ON WHICH IT OCCURRED, Z THE LAST IDENTIFIER READ, AND S THE DECIMAL VALUE OF THE LAST SYMBOL.

COMPILATION CONTINUES AFTER AN ERROR IS FOUND, BUT THERE MAY BE SOME FALSE ERROR MESSAGES AFTER AN INITIAL CORRECT ONE. COMPILER OUTPUT IS SWITCHED OFF.

1. IDENTIFIER DECLARED TWICE IN SAME BLOCK.
2. UNDECLARED IDENTIFIER.
3. NO [AFTER ARRAY NAME, EXCEPT AS A PROCEDURE PARAMETER, OR ORDINARY PROCEDURE USED AS A FUNCTION.
4. NO] AT END OF SUBSCRIPT LIST.
5. MORE THAN 63 VARIABLES IN MAIN PROGRAM OR A PROCEDURE.
6. NO $ AT END OF PROGRAM. (TOO MANY 'END'S).
7. NO 'ELSE' PART OF CONDITIONAL ARITHMETIC EXPRESSION.
8. DITTO. BOOLEAN OR DESIGNATIONAL EXPRESSION.
9. RELATIONAL OPERATOR NOT FOUND WHERE EXPECTED. MAY OCCUR IF RESTRICTION 1 IS IGNORED.
1Ø. ARITHMETIC PRIMARY DOES NOT START WITH +,-,.,(, DIGIT OR IDENTIFIER.
11. % (INTEGER DIVIDE) DOES NOT HAVE TWO INTEGER OPERANDS.
12.) MISSING IN ARITHMETIC EXPRESSION.
13. CONTROLLED VARIABLE IN 'FOR' IS UNDECLARED OR SUBSCRIPTED.
14.) MISSING IN BOOLEAN OR DESIGNATIONAL EXPRESSION.
15. MORE THAN 14Ø IDENTIFIERS IN SCOPE AT ONCE.
16. STATEMENT STARTS INCORRECTLY. IF THIS OCCURS AT THE TERMINATING DOLLAR, IT MEANS THERE ARE NOT ENOUGH 'END'S.
17. UNDECLARED OR UNSUITABLE IDENTIFIER ON LEFT OF :=
18. ARRAY DECLARATION FAULTY.
19. TYPE SPECIFICATION OF ACTUAL PARAMETER IS NOT 'LABEL',

'PROCEDURE', 'REAL''PROCEDURE', 'BOOLEAN''PROCEDURE' OR 'INTEGER''PROCEDURE'.
20. WRONG NUMBER OF SUBSCRIPTS. IN THE CASE OF FORMAL ARRAYS, THIS ERROR CANNOT BE DETECTED UNTIL RUN TIME.
21. NO) AFTER ACTUAL PARAMETER LIST.
22. 'FOR' STATEMENT ELEMENT NOT TERMINATED BY , OR 'DO'.
23. PROCEDURE BODY NOT DELIMITED BY ;.
24. :=' NOT FOUND WHERE EXPECTED.
25. NO 'THEN' AFTER 'IF'.
26. 'VALUE' SPECIFICATION IS NOT THE FIRST SPECIFICATION OF PROCEDURE FORMAL PARAMETERS.
27. $ IN MIDDLE OF PROGRAM. (TOO MANY 'BEGIN'S, OR UNMATCHED " OR ').
28. PROCEDURE FORMAL PARAMETER LIST NOT ENDED BY).
29. PARAMETER SPECIFIED TWICE, OR IS NOT IN FORMAL LIST, OR SPECIFICATION NOT TERMINATED BY ;.
30. LABEL/PROCEDURE LIST FULL.
31. 'UNTIL' NOT FOUND WHERE EXPECTED.
32. NO (AFTER NAME OF STANDARD PROCEDURE.
33. 'THEN' FOLLOWED IMMEDIATELY BY 'IF'.
34. PROCEDURE ACTUAL PARAMETER STARTS WITH UNDECLARED IDENTIFIER
35. FUNCTION OR VARIABLE USED AS PROCEDURE.
38. ARITHMETIC EXPRESSION CONTAINS BOOLEAN VARIABLE IN ILLEGAL CONTEXT.
39. PARAMETER SPECIFIED VALUE IS NOT IN FORMAL LIST.
40. PARAMETER SPECIFICATION NOT COMPLETE.
41. SIMPLE VARIABLE NOT CALLED BY VALUE, OR OTHER PARAMETER TYPE NOT CALLED BY NAME.
42. INPUT/OUTPUT PROCEDURE CALL ERROR.
44. INTEGER LITERAL NOT IN RANGE +-2047.

3.4 COMPILER CORE MAP

FIELD 0

0-1577 INTERPRETER
1600-2777 VARIABLES, WORKING STACK
3000-5477 ARRAYS (MAINLY IDENTIFIER TABLES)
5500-7577 FREE, BUT OCCUPIED BY SYSTEMS DEVICE HANDLER AT MANY INSTALLATIONS. SEE SECTION 6.

FIELD 1

1-7577 COMPILER CODE. THIS IS A CODE VERY SIMILAR TO THE ONE LISTED IN SECTION 9.

3.5 ORDER OF COMPILER OVERLAYS

BOTH THE SYSTEMS DEVICE OVERLAY AND THE EAE OVERLAY PATCH THE INTERPRETER. THE 16K PATCH CHANGES THE COMPILER CODE IN FIELD 1. THE CORRECT ORDER OF LOADING IS THEREFORE:

1. INTERPRETER

2. (A) EAE OVERLAY
(B) SYSTEM OR HIGH SPEED PAPER TAPE INTERRUPT OVERLAYS
(C) COMPILER CODES

THE ORDER OF LOADING A, B AND C IS IMMATERIAL.
THE BASIC COMPILER PAPER TAPE SUPPLIED CONTAINS ITEMS 1 AND 2(C).
3. 16K OVERLAY

THERE IS NO MODIFICATION TO THE COMPILER IN THE 12K SYSTEM.

4. THE RUN-TIME SYSTEM

THE RUN-TIME SYSTEM IS A SINGLE BINARY PROGRAM. IT INCLUDES THE LOADER AND THE FLOATING POINT PACKAGE, AND MAY INCLUDE A SYSTEM DEVICE HANDLER.
IT CAN BE LOADED WITH ANY ABSOLUTE BINARY LOADER, AND ITS STARTING ADDRESS IS 00200.

4.1 LOADING AND RUNNING A COMPILED PROGRAM

TO RUN A PROGRAM WHICH HAS NOT BEEN COMPILED INTO BINARY, LOAD THE RUN-TIME SYSTEM AND START AT 200 IN FIELD 0. IT WILL TYPE IN- . ANSWER WITH A DEVICE NUMBER 1 TO 3. IF 3 IS ANSWERED, IT WILL ASK FOR THE INPUT FILE NAME AS IS EXPLAINED IN SECTION 6.
WHEN LOADING IS COMPLETED, THE NEXT FREE LOCATION IN FIELD 0 WILL BE TYPED IN OCTAL, FOLLOWED BY ↑. TYPE CTRL/P TO EXECUTE THE PROGRAM, AND CTRL/C TO RETURN TO MONITOR. THE LOADER STARTS ITS TABLES IN FIELD 1 AT THE ADDRESS IN 10600. SHOULD IT BE NECESSARY TO RESTART THE LOADER, THIS MUST BE DONE AT LOCATION 14000. THE LOCATIONS FROM 00200 WILL ALREADY BE CHANGED IF THE LOADER WAS STARTED. IF THE PROGRAM ITSELF IS STARTED, THE LOADER MAY BE DESTROYED.

TO RUN A PROGRAM WHICH HAS BEEN COMPILED INTO BINARY, THE BINARY TAPE IS SIMPLY LOADED AFTER THE RUN-TIME SYSTEM, AND THE MACHINE STARTED AT 00200.

CTRL/S, CTRL/P AND CTRL/C WORK IN THE SAME WAY AS IN THE COMPILER. ADDITIONALLY, THE PROGRAM MAY BE RESTARTED AFTER CTRL/S BY CTRL/R. THE PROGRAM WILL NOT RUN CORRECTLY IF CTRL/S WAS USED TO INTERRUPT A WAIT ON THE HIGH SPEED READER OR TELETYPE, BECAUSE THE NEXT CODE IN LINE WILL BE OBEYED WITHOUT THE CODE RESPONSIBLE FOR THE READ BEING FINISHED. OUTPUT TRANSFERS AND SYSTEMS DEVICE TRANSFERS CANNOT BE INTERRUPTED BY CTRL/S.

4.2 OPERATING CHARACTERISTICS

THE VARIABLES ARE NOT SET TO ZERO WHEN THE RUN-TIME SYSTEM

IS STARTED. MAIN PROGRAM OUTER BLOCK VARIABLES, INCLUDING ARRAYS, WILL REMAIN UNCHANGED AND ACCESSIBLE IF THE PROGRAM IS RE-STARTED AFTER CTRL/S OR CTRL/P. THIS FEATURE CAN BE USED IN A PROGRAM WHICH USES THIS DATA IN DIFFERENT WAYS. A SUB-PROGRAM CAN BE CHOSEN BY AN OPTION QUESTION AT THE BEGINNING OF THE PROGRAM, WHICH WILL BE ASKED EVERY TIME CTRL/P IS TYPED.

THERE IS NO CHECKING FOR ARRAY SUBSCRIPTS OUT OF BOUNDS, OR THAT THE UPPER DECLARED BOUND OF AN ARRAY SUBSCRIPT IS GREATER THAN THE LOWER BOUND.

THERE ARE ABOUT 1160 (DECIMAL) LOCATIONS AVAILABLE FOR VARIABLES AND ARRAYS. REAL ELEMENTS OCCUPY 3 WORDS, AND INTEGERS AND BOOLEANS ONE. THERE IS AN OVERHEAD OF 3 WORDS FOR EVERY ACTIVE DEPTH OF ARRAY DECLARATION (I.E. WHEN AN INNER BLOCK WHICH CONTAINS AN ARRAY DECLARATION IS ACTUALLY ENTERED - THE SPACE IS GIVEN BACK WHEN THE BLOCK IS LEFT). EACH ARRAY CARRIES AN OVERHEAD OF ONE WORD PLUS TWICE THE NUMBER OF SUBSCRIPTS. FOR EXAMPLE, A BLOCK IN WHICH ONE ARRAY IS DECLARED, HAVING THREE SUBSCRIPTS, HAS AN OVERHEAD OF TEN WORDS.

4.3 RUN-TIME SYSTEM FAILURES.

THE MESSAGE IS ?DDDD, WHERE DDDD IS THE LOCATION IN THE RUN-TIME SYSTEM AT WHICH THE ERROR WAS DISCOVERED. FOR THIS REASON, THE FOLLOWING NUMBERS MAY CHANGE SLIGHTLY. TAKE THE NEAREST ONE.
↑ IS TYPED AFTER A FAILURE. ANSWER CTRL/C OR CTRL/P AS AFTER A SUCCESSFUL RUN.

202. NO INTERPRETIVE CODE OR LOADER PRESENT.
564. VARIABLE/ARRAY SPACE USED UP
465. ACTUAL NUMBER OF PARAMETERS DIFFERENT FROM NUMBER OF FORMALS.
516. ACTUAL AND FORMAL PARAMETERS INCOMPATIBLE. THE ONLY POSSIBLE CONVERSIONS ARE BETWEEN INTEGER AND REAL.
1002. ATTEMPT TO READ FROM DEVICE 3 WHEN SYSTEMS DEVICE ROUTINES ARE ABSENT.
1471. FLOATING POINT ERROR, PROBABLY DIVIDE BY ZERO.
1530. ERROR IN STANDARD FUNCTION OR EXPONENTIATION.
1616. WRONG NUMBER OF SUBSCRITS FOR FORMAL ARRAY. IS FREQUENTLY CAUSED BY USING ARRAY SUBSCRIPTS OUT OF BOUNDS.
2002. ATTEMPT TO WRITE TO DEVICE 3 WHEN SYSTEMS DEVICE ROUTINES ARE ABSENT.
2201. ATTEMPT TO DO DISK() WHEN SYSTEMS ROUTINES ARE ABSENT.
2460, 2463. DEVICE NUMBER NOT IN RANGE 0-7.
2465. DEVICE NOT AVAILABLE.

LOADER FAILURES.

THESE USUALLY INDICATE AN INTERNAL SYSTEMS FAILURE,

SINCE THE COMPILER IS SUPPOSED TO OUTPUT LEGAL LOADER CODE. A FAILURE WILL RESULT IF AN ATTEMPT IS MADE TO LOAD MACHINE CODE, IF THE ALGOL PROGRAM IS SO LONG THAT IT LOADS BEYOND THE LOCATION ADDRESSED BY 14200, OR IF IT CONTAINS TOO MANY LABELS FOR THE LOADER.

4042. NEGATIVE DEVICE CODE.
4045. DEVICE CODE>3.
4107. ITEM TO BE LOADED DOES NOT START WITH DIGIT, -, L, V OR F.
4204. L NOT FOLLOWED BY A DIGIT.
4225. LABEL (E.G. L24) NOT FOLLOWED BY , CR/LF OR =.
4243. FORWARD REFERENCE LIST FULL. COMPILER OUTPUT MUST BE ASSEMBLED BY MACRO.
4335. ATTEMPT TO LOAD BEYOND LOCATION ADDRESSED BY 14200. IF THE SYSTEMS DEVICE ROUTINES ARE NOT NEEDED, ALTHOUGH PRESENT, THE COMPILED CODE CAN BE LOADED ON TOP OF THEM. PUNCH THE CODE ONTO PAPER TAPE, PATCH 14200 TO 7577 AND LOAD FROM PAPER TAPE.
4271. LXX= NOT FOLLOWED BY L OR DIGIT.
4275. LXX=LYY, WHERE LYY HAS NOT BEEN DECLARED.
4546. SYSTEM DEVICE ROUTINES NEEDED BUT NOT PRESENT.
4552. TOO MANY LABELS. PROCEED AS FOR 4245.

4.4 RUN-TIME SYSTEM CORE MAP

FIELD 0

0-27 FREE
30-177 RESERVED FOR SYSTEM
200-7577 COMPILED ALGOL PROGRAM
THE TOP END OF THIS SPACE IS OCCUPIED BY THE SYSTEM DEVICE HANDLER, WHEN PRESENT. SEE SECTION 6.

FIELD 1

0-2477 INTERPRETER
2500-4755 VARIABLE AND ARRAY STACK
THE 12K OVERLAY (IF PRESENT) OCCUPIES LOCATIONS 2500-2577
4756-7577 FLOATING POINT PACKAGE

4.5 ARAY STORAGE - WARNING

TO SAVE TIME AND SPACE, THE RUN-TIME SYSTEM CHECKS NEITHER FOR SUBSCRIPTS OUT OF BOUNDS NOR THAT THE UPPER BOUND IS GREATER THAN THE LOWER BOUND WHEN AN ARRAY IS DECLARED. THE MOST FREQUENT MISTAKE IS TO OVERRUN THE DECLARED BOUNDS, AND THIS MAY CAUSE ?1616. HOWEVER, IN GENERAL THE EFFECT OF MISUSING ARRAYS IS UNPREDICTABLE, AND IF A PROGRAM IS NOT BEHAVING IN THE WAY EXPECTED, THIS IS THE FIRST AREA IN WHICH MISTAKES SHOULD BE SOUGHT.

4.6 ORDER OF RUN-TIME SYSTEM OVERLAYS

THE INTERPRETER OVERLAYS SOME INSTRUCTIONS IN THE FLOATING POINT PACKAGE.
THE SYSTEMS DEVICE OVERLAY PATCHES LOCATION 14200 IN THE LOADER (LAST FREE LOCATION IN FIELD 0).
THE 12K OVERLAY PATCHES LOCATION 10600 (ADDRESS OF START OF STACK) WHICH IS ALSO USED BY THE LOADER AS THE STARTING ADDRESS OF ITS TABLES.
THE 16K OVERLAY CHANGES CDF INSTRUCTIONS IN BOTH THE LOADER AND THE INTERPRETER.
USER OVERLAYS MAY WELL OVERWRITE BOTH 10600 AND 14200.

THE CORRECT ORDER FOR LOADING THE COMPONENTS IS THEREFORE:

1. FLOATING POINT PACKAGE (EAE OR NON-EAE VERSION AS REQUIRED)
2. INTERPRETER
3. LOADER
   THE BASIC PAPER TAPE SUPPLIED CONSISTS OF ITEMS 2 AND 3.
4. EAE PATCH IF NEEDED
5. SYSTEMS DEVICE OVERLAY
6. 12K OVERLAY AND 16K OVERLAY IN EITHER ORER
7. USER OVERLAYS

5. INPUT/OUTPUT DEVICE HANDLERS.

5.1 THE DEVICE NUMBER MECHANISM

THE INPUT AND OUTPUT ROUTINES LOOK UP THE ADDRESS OF THE ROUTINE TO TRANSFER TO/FROM ANY DEVICE NUMBER IN A TABLE. IN THE TAPES AS SUPPLIED, DEVICES 4 TO 7 ARE PRESET WITH THE ADDRESS OF THE ERROR ROUTINE, AS IS INPUT DEVICE 0. IN THE RUNTIME SYSTEM, DEVICE 3 IS A LINKING ROUTINE TO GET INTO FIELD 0, WHERE THE SYSTEMS DEVICE HANDLERS ARE - THIS WILL GIVE ITS OWN FAILURE INDICATION IF THERE ARE NO ROUTINES PRESENT.

5.2 ADDING EXTRA DEVICES

ONE APPROACH IS TO WRITE MACHINE CODE STATEMENTS IN THE ALGOL PROGRAM, AS DESCRIBED IN SECTION 2.4. HOWEVER, IT WILL USUALLY BE MORE CONVENIENT TO BUILD THE ROUTINES INTO THE RUN-TIME SYSTEM BY WRITING AN OVERLAY IN MACHINE CODE, SO THAT THE ALGOL PROGRAM DOES NOT HAVE TO BE COMPILED BY MACRO. THE CUSTOMISED OPERATING SYSTEM OR COMPILER MAY THEN BE SAVED ON THE SYSTEMS DEVICE, AND CALLED AS STANDARD.

TO WRITE YOUR OWN ROUTINES, SIMPLY PUT THE ADDRESS OF THE ROUTINE IN THE APPROPRIATE PLACE IN THE TABLE. THE SYSTEM WILL DO A "JMS I" THROUGH IT WHENEVER THAT DEVICE NUMBER OCCURS IN AN ALGOL INPUT/OUTPUT ROUTINE.
FOR OUTPUT DEVICES, THE CHARACTER WILL BE IN THE ACCUMULATOR. THERE IS NO NEED TO CLEAR THE ACCUMULATOR

BEFORE RETURNING. FOR INPUT DEVICES, RETURN WITH THE CHARACTER IN THE ACCUMULATOR.

THE WRITING OF SYSTEMS DEVICE OVERLAYS WILL BE DESCRIBED IN SECTION 6.4. ALTHOUGH DEVICE 3 LEADS TO A LINKING ROUTINE TO FIELD Ø IN THE STANDARD SYSTEM, THERE IS NOTHING TO STOP YOU MAKING DEVICE 3 INTO AN ORDINARY I/O DEVICE.

THE POSITION OF THE LISTS AND THE FREE SPACE AVAILABLE FOR THE ROUTINES IS DIFFERENT IN THE COMPILER AND OPERATING SYSTEMS.

5.3 ADDING DEVICES TO THE COMPILER

EVERYTHING IS IN FIELD Ø. THE INPUT DEVICE ROUTINE LIST GOES FROM 12ØØ (DEVICE Ø) TO 12Ø7 (DEVICE 7), AND THE OUTPUT LIST FROM 1210 (DEVICE Ø) TO 1217 (DEVICE 7); THE SPACE FROM 55ØØ TO 5777 (5677) IS FREE IF THE MONITOR (OS/8) ROUTINES ARE USED, OTHERWISE THE SPACE UP TO 7577 IS FREE. THERE IS NOTHING TO STOP YOU WRITING NEW ROUTINES FOR DEVICES 1 TO 3.
FOR EXAMPLE, A NEW TELETYPE OUTPUT ROUTINE TO DO TLS FIRST AND THEN WAIT FOR THE FLAG..

```
*1211;    TTO        /ADDRESS OF NEW ROUTINE TO TABLE
*55ØØ
TTO,      Ø; TLS; TSF: JMP .-1; JMP I TTO
```

THERE IS NO NEED TO CLEAR THE ACCUMULATOR.

5.4 ADDING DEVICES TO THE RUN-TIME SYSTEM

THE ROUTINES ARE IN FIELD 1. THE INPUT DEVICE LIST STARTS AT 24ØØ, AND THE OUTPUT DEVICE LIST AT 241Ø. THERE IS NO SAFE SPACE IN FIELD ONE IN THE STANDARD SYSTEM, BUT IT CAN BE GENERATED. THE ROUTINES FINISH AT ABOUT 247Ø (2577 IN 12K ROGALGOL), AND NORMALLY THE STACK IS BUILT FROM THIS POINT UPWARDS. LOCATION 6ØØ CONTAINS THE ADDRESS OF THE FIRST LOCATION AVAILABLE FOR THE STACK, AND IF THIS IS OVERWRITTEN, A HOLE BETWEEN THE ROUTINES AND THE STACK CAN BE GENERATED. THERE IS MUCH FREE SPACE IN FIELD Ø, FROM THE END OF THE COMPILED CODE TO 5777(5677) OR 7577, SO IT WOULD BE ADVANTAGEOUS TO PUT LARGE ROUTINES IN FIELD Ø, HAVING ONLY A LINKING ROUTINE IN FIELD 1.
LOCATION 14200, WITHIN THE LOADER ROUTINE, CONTAINS THE ADDRESS OF THE LAST LOCATION WHICH IS AVAILABLE FOR COMPILED CODE. THIS IS 7577 IN THE LOADER ITSELF, AND IS OVERWRITTEN BY 5777 (5677) WHEN THE MONITOR (OR OS/8) HANDLERS ARE PRESENT. USERS CAN FURTHER OVERWRITE THIS LOCATION TO RESERVE SPACE IN FIELD Ø.
THE ADDRESS OF THE NEXT FREE LOCATION IN FIELD Ø IS HELD IN ØØ175 AT RUN TIME.
THE EXAMPLE ILLUSTRATES ALL THESE POINTS.

```
FIELD Ø
*5600
FØTTO,    Ø; TLS; TSF; JMP .-1; CDF CIF 10; JMP I FØTTO

FIELD 1
*4200;    5577        /RESERVE SPACE IN FIELD Ø
*600;     2600        /RESERVE SPACE IN FIELD 1
*2411;    TTO         /ADDRESS NEW ROUTINE OUTPUT DEVICE 1
*2500
TTO,      Ø; CDF CIF; JMS I .+2; JMP I TTO; FØTTO
```

THE INITIALISATION ROUTINE, ENTERED WHEN AN ALGOL PROGRAM IS STARTED, HAS THE INSTRUCTIONS RFC; TLS; PLS; KCC; STARTING AT LOCATION 10640. TWO OF THESE INSTRUCTIONS MAY BE OVERWRITTEN WITH JMP I .+1; OWNINIT;, TO INITIALISE OTHER DEVICES, PROVIDED A RETURN IS MADE WITH THE OVERWRITTEN INSTRUCTIONS OBEYED ELSEWHERE.

ALTHOUGH THE EXAMPLES ARE OF I/O ROUTINES, THE SAME MECHANISM CAN BE USED TO OBEY ANY MACHINE CODE SEQUENCE.

6. SYSTEMS DEVICE HANDLERS

THE SYSTEMS DEVICE ROUTINES ENABLE ALGOL PROGRAMS (INCLUDING THE COMPILER) TO READ AND WRITE ASCII FORMAT FILES ON SYSTEM ADDRESSABLE DEVICES AND FILES. ONCE THE FILES HAVE BEEN INITIALISED, THEY ARE READ AND WRITTTEN BY THE INPUT/OUTPUT PROCEDURES (SECTION 2.2) IN A SEQUENTIAL MANNER JUST LIKE PAPER TAPE. DISK IS THE STANDARD PROCEDURE WHICH IS USED TO OPEN, CLOSE AND REWIND FILES. ACTUAL DEVICE TRANSFERS ARE ACTIVATED WHEN NECESSARY BY THE ORDINARY ROUTINES LIKE READ, WRITE AND TEXT CALLED WITH THE DEVICE NUMBER 3. SYSTEM DEVICE OUTPUT FILES ARE CLOSED BY SENDING CTRL/Z TO THEM BY CHOUT(3,154).

6.1 DISK MONITOR ROUTINES.

THESE ROUTINES CAN ONLY HANDLE FILES ON THE DF32 SYSTEMS DEVICE.

1. DISK(1), OPEN INPUT FILE.

4 PAGES OF CORE ARE SAVED IN SCRATCH BLOCKS 372-375, AND COMMAND DECODER IS CALLED INTO THE SAVED AREA. IT ASKS ONLY *IN-. TYPE A LIST OF UP TO 5 FILES. CHAOS MAY RESULT IF A DEVICE OTHER THAN THE SYSTEMS DEVICE IS SPECIFIED. COMMAND DECODER TYPES * ON A NEW LINE, THE FILE IS OPENED FOR READING. IF NO FILE NAME WAS TYPED, COMMAND DECODER IS CALLED AGAIN.

IF AN INPUT FILE WAS SPECIFIED IN RESPONSE TO DISK(2), (SEE BELOW), COMMAND DECODER IS NOT CALLED, AND THE PREVIOUSLY SPECIFIED FILE IS OPENED.

2. DISK(2). OPEN OUTPUT FILE.

COMMAND DECODER IS CALLED IN THE WAY DESCRIBED ABOVE. THIS TIME *OUT- AND *IN- QUESTIONS ARE ASKED. THE OUTPUT FILE MUST BE SPECIFIED, BUT THE INPUT FILE NEED NOT. IF AN INPUT FILE IS TYPED, IT WILL BE OPENED BY A SUBSEQUENT DISK(1).
IF AN OUTPUT FILE IS NOT SPECIFIED, COMMAND DECODER WILL BE CALLED AGAIN.
IF THERE WAS ALREADY AN ASCII FILE OF THE SAME NAME AS THE NEW OUTPUT FILE, IT IS ERASED BEFORE ANY NEW CHARACTERS ARE SENT TO IT.

3. DISK(Ø). REWIND INPUT FILE.

THIS REQUIRES NO TELETYPE RESPONSE. A FAILURE WILL OCCUR IF THERE WAS NEVER A FILE OPEN.

4. DISK(3). SWOP INPUT AND OUTPUT FILES.

THIS WILL FAIL UNLESS THERE HAS BEEN BOTH AN INPUT AND AN OUTPUT FILE OPEN. THE OUTPUT FILE MUST BE CLOSED BEFORE DOING DISK(3). THE PREVIOUS READING FILE WILL BE ERASED BY DISK(3). THEREFORE, IF YOU DO NOT NEED TO READ A FILE AND THEN WRITE ON IT, BUT JUST WRITE A FILE AND READ IT BACK, YOU MUST GIVE A DUMMY NAME FOR THE INITIAL INPUT FILE, OR CLOSE THE OUTPUT FILE AND OPEN IT FOR READING WITH DISK(1).

5. MONITOR ROUTINE ERRORS.

?6226. HARDWARE READ ERROR.
?6260. INPUT FILE EXHAUSTED.
?6502. HARDWARE ERROR CALLING COMMAND DECODER.
?7022. DISK(3) WITH NO INPUT OPEN.
?7025. DISK(3) WITH NO OUTPUT FILE SPECIFIED.
?7060. WRITE COMMAND WITH NO OUTPUT OPEN.
?7066. HARDWARE WRITE FAILURE.
?6065. HARDWARE ERROR ON SAM BLOCK TRANSFER.
?6075. DISK FULL.
?6116. HARDWARE ERROR ON SCRATCH BLOCK.

THE MONITOR ROUTINES CAN HANDLE ONLY ASCII FILES.
THE FIRST FEW SYSTEMS DELIVERED CONTAIN A MONITOR ROUTINE WHICH CANNOT DEAL WITH MULTIPLE INPUT FILES. LATER VERSIONS WILL ACCEPT UP TO 5 INPUT FILE NAMES. THE NEXT FILE IN THE LIST WILL BE OPENED WHEN EITHER THE PREVIOUS FILE IS EXHAUSTED OR DISK(1) IS EXECUTED AGAIN. IF THERE ARE NO MORE FILES IN THE LIST AND DISK(1) IS DONE, COMMAND DECODER WILL BE CALLED, BUT IF THE LAST FILE IN THE LIST IS EXHAUSTED ?6455 WILL OCCUR. DISK(Ø) WILL CAUSE THE FIRST FILE IN THE LIST TO BE REOPENED. DISK(2) WILL DESTROY ANY EXISTING INPUT LIST, BUT WILL NOT AFFECT AN INPUT FILE WHICH IS ALREADY OPEN. DISK(3) WILL CAUSE THE FIRST FILE IN THE LIST TO BECOME THE OUTPUT FILE. IF THE END OF THE OLD OUTPUT FILE IS EXCEEDED ON READING AFTER DISK(3) THE RESULTS ARE UNDEFINED. DISK(3) SHOULD THEREFORE BE USED WITH CAUTION.

FORM FEED (214) IS NOT TRANSMITTED TO THE PROGRAM, BUT IF THE FILE WAS CREATED BY THE ALGOL SYSTEM END OF FILE IS INDICATED BY CTRL/Z WHICH IS NOT SUPRESSED.

IF IT IS DESIRED TO REWIND TO THE BEGINNING OF THE INPUT LIST EVEN WHEN NEW FILES HAVE BEEN OPENED EXPLICITLY BY DISK(1), ALL OPEN INPUT INSTRUCTIONS EXCEPT THE FIRST SHOULD BE DONE BY DISK(-2047) (=4001).

6.2 OS/8 ROUTINES

THESE ROUTINES WERE WRITTEN BY ALISTAIR WINDRAM, OF THE AGRICULTURAL RESEARCH COUNCIL'S GRASSLAND RESEARCH INSTITUTE, HURLEY, MAIDENHEAD, BERKS.

SCRATCH BLOCKS 40 AND 41 ARE USED BY THESE ROUTINES. THEY CONTAIN CODE TO OPEN AND CLOSE FILES, WHICH IS CALLED INTO THE BUFFER AREAS WHEN NEEDED. THE CODE IS WRITTEN ONTO THE SCRATCH BLOCKS AT THE FIRST USE OF THE SYSTEMS ROUTINES. AN ALGOL PROGRAM USING THEM CANNOT BE SAVED AFTER IT HAS BEEN STARTED, AS SOME OF THE CODE WILL NOT BE IN CORE. THIS INCLUDES LOADING THE COMPILED CODE INTO MEMORY USING DEVICE 3, BECAUSE THE LOADER USES THE SAME ROUTINES AS THE ALGOL PROGRAM.

COMMAND DECODER IS CALLED IN SPECIAL MODE, WHICH MEANS THAT NO FILE EXTENSION IS ASSUMED. UNLIKE THE MONITOR SYSTEM, ANY OS/8 DEVICE MAY BE USED, PROVIDED THAT ITS HANDLER OCCUPIES ONE PAGE AND NOT TWO.

1. DISK(1).
THE ROUTINE TYPES "INPUT FILENAME? ", AND THEN CALLS COMMAND DECODER. WAIT UNTIL COMMAND DECODER TYPES *, AND THEN TYPE A SINGLE FILE NAME, WHICH NEED NOT BE ON THE SYSTEMS DEVICE.

2. DISK(2). AS ABOVE BUT FOR OUTPUT FILES.

3. DISK(0). REWIND INPUT FILE, AS FOR MONITOR.

4. DISK(3). CLOSE OUTPUT FILE (IF NOT DONE ALREADY) AND OPEN IT FOR READING.

5. OS/8 ROUTINE FAILURES.

?5724. ATTEMT TO READ WITH FILE NOT OPENED, OR FILE ALREADY EXHAUSTED.
?5735. FAILURE IN INPUT DEVICE HANDLER.
?6020. ATTEMPT TO WRITE WITH NO FILE OPENED OR ALREADY CLOSED.
?6056. SPACE AVAILABLE TO OUTPUT FILE FULL.
?6067. FAILURE IN OUTPUT DEVICE HANDLER.

?6123. DISK(Ø) WITH INPUT DEVICE NEVER OPENED.
?6143. ERROR IN SYSTEMS DEVICE HANDLER DURING DISK(1).
?6162. DITTO. DURING DISK 2 OR 3, OR CLOSING OUTPUT FILE.
?6212. DISK(3) WITH FILE NEVER OPEN.
?6417. OUTPUT DEVICE HANDLER IS TWO PAGE.
?6420. CATASTROPHIC SOFTWARE FAILURE.
?6476. OUTPUT FILE FAILED TO OPEN. (PTR:?).
?6477. OUTPUT FILE FAILED TO CLOSE.
?7006. ERROR IN SYSTEMS DEVICE HANDLER ON BLOCK 40.
?7013. DITTO. BLOCK 41.
?7416. INPUT DEVICE HANDLER IS TWO PAGE.
?7440. INPUT FILE FAILED TO OPEN (PTP:?).

6.3 WRITING YOUR OWN SYSTEM DECICE HANDLERS

THE SYSTEMS DEVICE ROUTINES ARE ALWAYS IN FIELD Ø. THEY HAVE THREE ENTRY POINTS.

1. INITIALISE FILES. THIS IS ENTERED WHEN DISK(J) OCCURS IN THE ALGOL. J IS IN THE ACCUMULATOR.

2. READ A CHARACTER. RETURN.WITH THE CHARACTER IN THE ACCUMULATOR.

3. OUTPUT A CHARACTER. THE CHARACTER IS IN THE ACCUMULATOR ON ENTRY.

THE OVERLAYS NEED TO OVERWRITE DIFFERENT ADDRESSES IN THE COMPILER AND RUN-TIME SYSTEMS. IN THE CASE OF OVERLAYS FOR THE RUN-TIME SYSTEM, THE ROUTINES MUST DO CDF CIF 1Ø BEFORE RETURNING.

COMPILER.
ALL PATCHES ARE IN FIELD Ø

LOCATION	PATCHED TO CONTAIN
77	ADDRESS OF INITIALISING ROUTINE (FOR DISK)
12Ø3	ADDRESS OF CHARACTER INPUT ROUTINE
1213	ADDRESS OF CHARACTER OUTPUT ROUTINE

A FAILURE MESSAGE CAN BE GENERATED BY DOING JMS 1457

RUNTIME SYSTEM.
ALL PATCHES ARE IN FIELD 1. THE ROUTINES ARE IN FIELD Ø

LOCATION	PATCHED TO CONTAIN
1Ø01	CDF CIF
1Ø04	ADDRESS OF INPUT ROUTINE
2Ø01	CDF CIF
2Ø04	ADDRESS OF OUTPUT ROUTINE
22Ø0	CDF CIF
22Ø4	ADDRESS OF FILE INITIALISATION ROUTINE
42Ø0	LAST LOCATION IN FIELD Ø AVAILABLE FOR COMPILED CODE

A FAILURE MESSAGE CAN BE GENERATED BY PLACING THE FAILURE NUMBER IN 11200, AND JUMPING TO 11201.

7. DATA STORAGE IN UNUSED PROGRAM SPACE

7.1 USING SPARE DEVICE NUMBERS

AS HAS BEEN EXPLAINED IN SECTION 5, THE INPUT/OUTPUT ROUTINES CAN BE USED TO ACTIVATE ANY PIECE OF MACHINE CODE, AND PROBABLY THE BEST WAY TO USE ANY FREE PROGRAM SPACE IS TO USE SOME SPARE DEVICE NUMBERS TO STORE AND FETCH INFORMATION FROM FIELD Ø. THIS HAS THE ADVANTAGE THAT THERE IS NO NEED TO COMPILE THE ALGOL COMPILER OUTPUT WITH MACRO - THE ROUTINES CAN BE BUILT INTO THE OPERATING SYSTEM. AS A SUGGESTION FOR INSTALLATIONS WITH NO SYSTEMS DEVICE, DISK(INDEX) MAY BE USED TO SET A POINTER TO A LOCATION IN FIELD Ø, AND IF THIS POINTER IS AN AUTO-INDEX REGISTER, THE FREE SPACE WILL BEHAVE LIKE A FILE WHEN DEVICE 3 IS USED. DISK(Ø) WILL REWIND THE FILE. A SIMPLE CODING, WITH NO CHECKING FOR INDEX WITHIN BOUNDS, FOLLOWS.

```
FIELD Ø
*10; INDEX, Ø
*7540; SETIND, Ø; TAD 175; DCA INDEX; CDF CIF 10; JMP I SETIND
DEPOS, Ø; DCA I INDEX; CDF CIF 10; JMP I DEPOS
FETCH, Ø; TAD I INDEX; CDF CIF 10; JMP I DEPOS
FIELD 1
*4200; SETIND-1      /PROTECT CODE FROM LOADER
*2001; CDF CDF
*2004; DEPOS
*2200; CDF CIF
*2204; SETIND
*1001; CDF CIF
*1004; FETCH
```

7.2 USING ALGOL PROCEDURES

THE ABOVE METHOD WORKS WELL FOR INTEGERS, BUT IS MORE DIFFICULT TO USE WITH REALS, BECAUSE THE ACCUMULATOR CANNOT HOLD A REAL NUMBER. FOUR ALGOL PROCEDURES WITH MACHINE CODE BODIES ARE PROVIDED, WHICH ALLOW THE FREE SPACE TO BE USED FOR INTEGER OR REAL STORAGE. THEY ACT AS ARRAYS WITH ONE DIMENSION AND A LOWER BOUND OF Ø. TWO VERSIONS ARE PROVIDED, ONE FOR INTEGERS AND ONE FOR REAL NUMBERS. CARE MUST BE TAKEN IF THE TWO VERSIONS ARE TO BE USED TOGETHER, AS THEY STORE NUMBERS IN THE SAME AREA OF MEMORY. FOR REAL NUMBERS, THE INDEX (SUBSCRIPT) IS MULTIPLIED BY 3 BY THE ROUTINES, TO ALLOW THREE WORDS OF STORAGE FOR EACH NUMBER. TO TAKE AN EXAMPLE, IF IT WERE REQUIRED TO STORE 100 INTEGERS, AND SOME REALS AS WELL, THE INTEGERS COULD BE STORED WITH INDICES Ø-99, AND REALS WITH INDICES GREATER THAN 33.

THE HEADINGS ARE:-

```
'PROCEDURE' FØRST(INDEX,VALUE);
'VALUE' INDEX, VALUE; 'INTEGER' INDEX; 'REAL' VALUE;
```

```
'COMMENT' STORE REAL NUMBERS IN FEILD Ø;
'TRANS' . . . . . .'ALGOL';

'REAL''PROCEDURE' FØRGET(INDEX);
'VALUE' INXEX; 'INTEGER' INDEX;
'COMMENT' RETRIEVE REAL NUMBERS FROM FIELD Ø;
'TRANS' . . . . . .'ALGOL';

'PROCEDURE' FØIST(INDEX,VALUE);
'VALUE' INDEX,VALUE; 'INTEGER' INDEX,VALUE;
'COMMENT' STORE INTEGER IN FIELD Ø;
'TRANS' . . . . . .'ALGOL';

'INTEGER''PROCEDURE' FØIGET(INDEX);
'VALUE' INDEX; 'INTEGER' INDEX;
'COMMENT' RETRIEVE INTEGERS FROM FIELD Ø;
'TRANS' . . . . . 'ALGOL';
```

THE PROCEDURES SHOULD BE INSERTED INTO THE PROGRAM IMMEDIATELY AFTER THE FIRST BEGIN, OTHERWISE THE MACHINE CODE INSTRUCTIONS MIGHT CROSS A PAGE BOUNDARY.

THE SPACE AVAILABLE FOR STORING NUMBERS CAN BE FOUND BY FINDING THE LAST LOCATION OCCUPIED BY THE PROGRAM, AND SUBTRACTING THIS FROM 5777 (OR 7577 IF THE DISK ROUTINES ARE NOT USED). THE RESULT MUST BE DIVIDED BY THREE TO FIND THE NUMBER OF REALS WHICH CAN BE STORED. LOAD THE BINARY VERSION OF THE COMPILED PROGRAM (IT MUST BE COMPILED INTO BINARY BY MACRO OR PAL BECAUSE OF THE MACHINE CODE PROCEDURE BODIES) AND FIND THE CONTENTS OF LOCATION 175 IN FIELD 1. THIS IS THE NEXT FREE LOCATION IN FIELD Ø.

8. INTERNAL REPRESENTATION OF ALGOL BASIC SYMBOLS.

THESE ARE DECIMAL, AND ARE THE VALUES WHICH ARE PRINTED IN THE FAILURE MESSAGE. LONG BASIC SYMBOLS ARE REPRESENTED ON PAPER TAPE BY THE WORD ENCLOSED IN SINGLE QUOTES, AND IN THE COMPILER BY 40*1ST LETTER+SECOND LETTER.

IF A COMPILER ERROR MESSAGE CONTAINS A SYMBOL WHICH IS NOT ON THE LIST, AN ILLEGAL COMPOUND SYMBOL FOLLOWS A '. THE USUAL CAUSE OF THIS IS AN UNMATCHED '.

LETTERS A-Z		1-26
DIGITS Ø-9		48-57
"	(STRING BRACKETS)	34
#	(NOT EQUAL TO)	35
$	(END OF PROGRAM)	36
%	(INTEGER DIVIDE)	37
(		40
)		41
*	(MULTIPLY)	42
+		43
,		44

-		45
.		46
/	(REAL DIVIDE)	47
:		58
;		59
[		27
]		29
↑	(EXPONENTIATION)	30
<	(LESS THAN)	60
=		61
>	(GREATER THAN)	62
<=	(LESS THAN OR EQUAL)	63
>=	(GREATER OR EQUAL)	38
:=		33
'BEGIN'		85
'END'		214
'FOR'		255
'STEP'		780
'UNTIL'		854
'DO'		175
'IF'		366
'THEN'		808
'ELSE'		212
'GOTO'		295
'PROCEDURE'		653
'REAL'		725
'INTEGER'		374
'BOOLEAN'		95
'ARRAY'		122
'VALUE'		881
'TRANS'		818
'ALGOL'		116
'TRUE'		818
'FALSE'		241
'NOT'		575
'AND'		118
'OR'		618
'COMMENT'		135
'LABEL'		481
'WHILE'		928
'EQUIVALENT'		217

9. OPERATION CODES

THESE ARE THE SIX BIT CODES WHICH ARE OUTPUT BY THE COMPILER. THE LIST GIVES THEIR NUMBER IN DECIMAL.
WHEN A PROGRAM FAILS, IT IS POSSIBLE TO TRACE WHERE IT WENT WRONG BY EXAMINING THE COMPILER OUTPUT IN MEMORY, IN CONJUNCTION WITH THE COMPILER SYMBOL TABLE. ALL COMPILER OUTPUT IS IN FIELD 0. THE POINTERS BELOW ARE ALL IN FIELD 1.

LOCATION 22 POINTS AT THE START OF THE VARIABLE SPACE OF THE CURRENT PROCEDURE. THE LOCATION POINTED AT (IN FIELD 1)

CONTAINS THE CURRENT PROCEDURE NUMBER. THE NEXT LOCATION CONTAINS THE RETURN ADDRESS, AND THE NEXT A POINTER TO THE START OF THE PREVIOUS CALLING PROCEDURE. USING THESE POINTERS THE SEQUENCE OF CALLING CAN BE TRACED BACKWARDS FROM THE POINT OF FAILURE.
LOCATION 26 CONTAINS THE CURRENT INSTRUCTION WORD.
LOCATION 27 CONTAINS THE ADDRESS OF THE CURRENT INSTRUCTION.

ARITHMETIC AND BOOLEAN OPERATIONS ARE DONE ON A STACK. THE TOP ELEMENT WILL BE CALLED S1, THE NEXT S2, AND SO ON. THE STACK IS ALSO USED BY THE RUN TIME ROUTINES FOR THE PASSING OF ARGUMENTS.

CODE	OPERATION
Ø	NO OPERATION.
1	DECLARE ARRAY. S1=DEPTH OF DECLARATION. S2=NUMBER OF DECLARATIONS IN MULTIPLE. S3=VARIABLE NUMBER OF FIRST DECLARATION. S4=NUMBER OF WORDS IN EACH ELEMENT. S5=NUMBER OF SUBSCRIPTS. S6=UPPER BOUND OF LAST SUBSCRIPT. S7=LOWER BOUND OF LAST SUBSCRIPT. S8, S9, ETC., BOUNDS OF OTHER SUBSCRIPTS.
2	DEVICE:=S1.
3	READ TO S1 FROM INPUT DEVICE.
4	STORE LOCAL VARIABLE FROM S1. FOLLOWED BY VARIABLE NUMBER.
5	PRINT STRING. FOLLOWED BY STRIPPED ASCII CODE, TERMINATED BY ZERO.
6	INTEGER PRINT S1
7	READ NEXT CHARACTER TO S1.
8	PRINT S1 AS CHARACTER.
9	JUMP. LOCATION IS IN NEXT WORD.
10	LEAVE PROCEDURE.
11	ENTER PROCEDURE WITH NO PARAMETERS, ADDRESS IS IN NEXT WORD.
12	GET LOCAL VARIABLE TO S1. FOLLOWED BY VARIABLE NUMBER.
13	INTEGER ADD
14	GET ARRAY ELEMENT. S1=ADDRESS OF DOPE VECTOR, S2=NUMBER OF SUBSCRIPTS, S3 ETC., SUBSCRIPTS.
15	STORE ARRAY ELEMENT. S1=VALUE, S2 ETC. ARE SAME AS S1 ETC. IN GET ARRAY ELEMENT.
16	SET 12 BIT CONSTANT IN S1.
17	INTEGER NEGATE.
18	REAL↑INTEGER.
19	INTEGER MULTIPLY.
20	INTEGER DIVIDE.
21	INTEGER SUBTRACT.
22	S1:=S1=Ø.
23	S1:=S1≥Ø.
24	S1:=S1<Ø.
25	GET ANY VARIABLE TO S1. FOLLOWED BY LEVEL NUMBER AND VARIABLE NUMBER.

```
26      STORE TO ANY VARIABLE FROM S1.
27      STANDARD FUNCTION. FOLLOWED BY ANOTHER CODE.
        2 SQRT    3 SIN     4 COS
        5 ARCTAN  6 EXP     7 LN
        8 SIGN    9 ENTIER  10 ABS
28      JUMP IF S1='FALSE'. ADDRESS IN NEXT WORD.
29      SET ZERO.
30      S1:='NOT' S1.
31      S1:=S1'AND'S2.
32      S1:=S1'OR'S2.
33      S1:=S1'EQUIV' S2.
34      FOR STATEMENT CALCULATOR.
        S1=FINAL VALUE
        S2=INCREMENT
        S3=TYPE OF CONTROLLED VARIABLE.
        S4=0 FOR NO INCREMENT AT THE FIRST TEST
        DEPTH OF CONTROLLED VARIABLE IN NEXT 6 BITS OF
        THE PROGRAM.
        CONTROLLED VARIABLE NUMBER IN THE FOLLOWING 6 BITS.
35      IOC
36      ENTER PROCEDURE, ADDRESS IN NEXT WORD.
        S1=NUMBER OF PARAMETERS.
        S2=TYPE OF FIRST PARAMETER.
        S3=VALUE OF FIRST PARAMETER. ETC.
        THE ADDRESS OF THE PROCEDURE IS IN THE NEXT LOCATION.
        THE FIRST LOCATION OF A PROCEDURE IS THE FIXED SPACE
        ON THE VARIABLE STACK REQUIRED BY THE PROCEDURE.
        THE NEXT 6 BITS IS THE PROCEDURE NUMBER, FOLLOWED
        BY THE TYPE SPECIFICATION OF THE PARAMETERS, IN
        REVERSE ORDER.
37      STORE OUTER BLOCK VARIABLE FROM S1.
38      FETCH OUTER BLOCK VARIABLE TO S1.
39      DISK FILE SET-UP.
40      SKIP
41      INTEGER S1:=SIGN(S2-S1)
42      SET 6 BIT CONSTANT
43      FIX S1
44      FLOAT S1
45      SET FLOATING POINT CONSTANT
46      FLOATING NEGATE
47      SET NEXT WHOLE WORD
48      RWRITE S1
49      REAL↑REAL
50      FLOATING MULTIPLY
51      FLOAT S2
52      FLOATING DIVIDE
53      FLOATING ADD
54      FLOATING SUBTRACT
55      LEAVE INTERPRETER (AT 'TRANS')
56      FLOATING S1:=SIGN(S2-S1)
57      JUMP TO ADDRESS IN S1
58      ENTER PROCEDURE WHOSE ADDRESS IS IN S1.
        NO PARAMETERS
59      AS 58 BUT NUMBER OF PARAMETERS IN S1. FOR 58 AND
        59 THE REST OF THE STACK IS SET UP AS FOR 36
60      PRINT STRING WHOSE ADDRESS IS IN S1
```

61	SET STACK DEPTH. FOLLOWED BY 6 BIT CODES FOR PROCEDURE AND ARRAY DEPTHS REQUIRED
62	IF FOLLOWED BY Ø, DELETE TOP LEVEL OF ARRAY STACK OR IF BY 1 A PROCEDURE LEVEL
63	STOP, END OF PROGRAM. PRINTS ↑

**** END OF TEXT ****