AAEC/E256

AAEC/E256

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

AEJCL - A JCL SYNTAX CHECKING FACILITY

by

I.J. HAYES

February 1973

ISBN 0 642 99536 2

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AEJCL - A JCL SYNTAX CHECKING FACILITY

by

I. J. HAYES

ABSTRACT

AEJCL provides a listing and JCL syntax checking facility on a PDP/9L computer system. The syntax checking system is controlled by a directed graph structure which is initially written in a machine independent language and then translated to PDP/9L code automatically. The input and output sections of the system are handled asynchronously by a multitasking master control program.

National Library of Australia card number and ISBN 0 642 99536 2

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA–INIS–12 (INIS: Manual for Indexing) and IAEA–INIS–13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

A CODES; IBM COMPUTERS; PDP COMPUTERS; PROGRAMMING LANGUAGES

CONTENTS

1. INTRODUCTION

AEJCL is a PDP/9L program (Digital Equipment Corporation 1968) that runs in a partition of the PDP/9L computer system and syntax checks IBM360 Job Control Language (JCL) (IBM 1970) as well as listing cards.

1.1 Description of Facilities

AEJCL may be used to perform the following operations:

a list cards on an ANELEX line printer either single spaced (60 lines per page) or double spaced (30 lines per page);

b syntax check IBM360 Job Control Language, flagging any syntax errors that are detected and giving an appropriate error number;

c call programs from the IBM360 computer system into the PDP/9L computer and release control to the called program; and

d send messages and commands to the IBM360 computer system.

1.2 AEJCL Control Cards

The control cards used by AEJCL for listing and communication with the IBM360 system are the same as those used by the FORTRAN syntax analysis program, AESYNTAX (Barry 1972),with the exception that a $JCL control card is used to denote the start of a deck that is to be JCL syntax checked.

The control cards are:

$LIST1 - for a single spaced listing;

$LIST2 - for a double spaced listing;

$JCL - for a syntax checked listing with any appropriate error diagnostics; and

$CALL - for communication with the IBM360 system either for calling programs into the PDP/9L computer or to send messages or commands to the IBM360 system.

1.3 Outline of AEJCL Program Structure

AEJCL is divided into three logical tasks:

1 read cards into buffers,

2 print out lines from buffers and

3 perform syntax analysis of JCL statements.

These tasks are organised in the above priority ordering so that the input/output bound tasks receive more prompt attention than the analysis phase

and thus enable the input/output devices to be driven at full capacity.

The syntax analysis is performed by a program which matches the input JCL statement against a pattern which represents the syntax of JCL. This pattern is represented internally by a directed graph structure.

2. SYNTAX ANALYSIS IN AEJCL

The syntax of Operating System 360 Job Control Language (JCL) used in AEJCL is represented by a directed graph structure in which each element of the syntax is represented by a node in the graph. For the syntax of a statement to be correct, every element of the statement must match the corresponding node in the directed graph. The elements of the syntax are either single characters, alphanumeric names, keywords or groupings of other elements. Each element in the syntax has a corresponding node in the graph which represents that element and in addition each node in the graph may have a successor (or forward) node and an alternate node so that the ordering of the elements within the complete syntax may be represented. Optionally associated with each node in the graph is an action routine (with an optional operand) which is executed on a match of a node and an element in the statement.

2.1 Directed Graph Node

The directed graph node which represents an element of the syntax consists of:

Symbol - i the internal code for a character,
ii the internal code for a name or keyword,
iii an internal code that will match any character or keyword, or
iv the name of a 'sub-graph' which represents the node.
(see Appendix 2 for a table of internal codes).

Alternate - i a pointer to the next node to be examined if this node does not match,
ii an error number if there is no alternate,
iii an indication that a return from sub-graph is necessary if this node does not match. This return may be either successful (a complete sub-graph has matched) or unsuccessful (none of the sub-graph has matched).

Forward - i a pointer to the next node to examine if this node matches, or
ii an indication that a return from a sub-graph is necessary if this node matches.

Action - an optional subroutine to be executed if this node is matched.

Operand - an optional operand for the subroutine.

2.2 Format of the Directed Graph Node for the PDP/9L Computer

A node in the PDP/9L computer consists of from three to five words:

Ø	1	2	3	4
Symbol	Alternate	Forward	Action	Operand.

where:

Symbol - i if negative it is the internal code for a character, name or keyword,
ii if zero then the node will match any element, or
iii if positive then it is the address of a sub-graph which represents the node.

Alternate - i if positive then the address of an alternate node,
ii if negative then either:
a if bit (1) = Ø then an error number, or
b if bit (1) = 1 then the node represents an alternate return which is either successful (bit (2) = 1) or unsuccessful (bit (2) = Ø).

Forward - i if positive then there is an action routine associated with this node,
ii if negative then there is no action routine associated with this node,
iii if bit (1) = Ø then the address of the forward node, or
iv if bit (1) = 1 then this indicates a forward return is to be taken.

Action - the address of the action subroutine.

Operand - any valid PDP/9L constant.

2.3 The Syntax Checking Algorithm (JCL NODE)

The logical flow of the syntax checking algorithm is given here in a pseudo-computer language (see Appendix 7 for the PDP/9L assembly code version)

```
*                                   JCL NODE

jn1Ø:    call initialise
         current-node: = start-of-graph
jn15:    current-element: = next-element-of-input-statement
jn3Ø:    if current-node represents a sub-graph then go to jn7Ø
         if current-element = symbol (current-node) then go to jn5Ø
         if symbol (current-node) = name then go to jn45
```

```
jn4Ø:     if there is no alternate then go to jn6Ø
*                                MISMATCH
          current-node: = alternate (current-node)
          go to jn3Ø
*                    ALLOW FOR NON-RESERVED KEYWORDS.
jn45:     if current-element is not a keyword then go to jn4Ø
*                                MATCH
jn5Ø:     if there is no action routine then go to jn55
          call action-routine (operand)
          if action-routine indicates a mismatch then go to jn4Ø
jn55:     current-node: = forward (current-node)
          if forward pointer does not indicate return then go to jn15
          unstack (current-node)
          go to jn5Ø
*                    MISMATCH - NO ALTERNATE.
jn6Ø:     if alternate indicates a return then go to jn8Ø
          call error (error-number)
          go to jn1Ø
*                    SUB-GRAPH OR ANYTHING.
jn7Ø:     if current-node matches anything then go to jn5Ø
          stack (current-node)
          current-node: = symbol (current-node)
          go to jn3Ø
*                    ALTERNATE RETURN.
jn8Ø:     if return is successful then go to jn1ØØ
          unstack (current-node)
          go to jn4Ø
jn1ØØ:    unstack (current-node)
          if current-node has no action-routine then go to jn1Ø5
          call action-routine (operand)
jn1Ø5:    if current-node indicates a return then go to jn1ØØ
          current-node: = forward (current-node)
          go to jn3Ø
```

2.4 Statement Scan Routine (JCLSCAN, JCLKEY)

The statement scanning routine in AEJCL gets the next syntactic element on an input statement. It operates in six modes:

1 Character-by-character scanning without any analysis,

2 Recognises 1-8 character names or keywords as well as integer

constants,

3 Recognises any alphanumeric string of up to 14Ø characters as the parameter field on the EXEC statement,

4 Recognises a dataset name - an alphanumeric string possibly containing a '.',

5 Recognises the programmer's name on the job card - an alphanumeric string possibly containing '.' or '-', and

6 Recognises a hexadecimal unit address.

To allow for the above modes, JCLSCAN is governed by a multiway switch JSCAN1SW. For case 2 above when a name has been recognised, the keyword search routine (JCLKEY) is called to determine whether or not the name is a keyword and if so, returns the internal code of the keyword.

2.5 Character-by-Character Input (JCLNEXT, JCLGET,JCLSAVE)

For every character, JCLSCAN calls on a routine JCLNEXT which returns each character on the card up to and including column seventy-one and then returns a special end of statement character which the analysing routine recognises. In addition, before returning the end of statement internal code, JCLNEXT initiates the printing of the card it has just finished processing (JGETPRT), and on the next entry to JCLNEXT it reads the next card (JCLGET) and returns the first character on that card.

While assembling the multicharacter groups above JCLSCAN sometimes, to establish the end of the group, needs to read one character past the end of the group. When this situation arises JCLSCAN calls upon a routine JCLSAVE which saves the extra character so that on the next call to JCLNEXT the same character is returned.

2.6 The Language for Describing Directed Graph Nodes

To make the process of describing the syntax of JCL easier, a language for describing a node in the graph was used. Statements in this very simple (one statement) language were then translated to PDP/9L assembly statements by a simple SNOBOL program (see Appendix 1). The use of the language gave a number of distinct advantages.

1 The syntax was specified more easily and compactly which reduced the initial coding times as well as making the task of debugging the syntax a much more manageable task.

2 Because of the relative compactness of the coding in the simple language the number of simple coding errors was greatly reduced and these were quickly recognised by either the simple SNOBOL program or in the debugging phase.

3 The language description of the syntax in AEJCL affords a well documented, complete description of the syntax.

4 The syntax was specified in a machine independent form so that implementation on a machine other than a PDP/9L would be greatly simplified as only the SNOBOL program would need to be rewritten.

The syntax of the simple language is:

[label] [symbol] [(action[(operand)])][→forward][;alternate][/ comment]

where [ ] indicates an optional field,

label - the name of the node,

symbol - i a single character in quotes
ii the name of an unprintable character, or a group of characters,
iii the name of a keyword,
iv null implies the node will match anything
v '#' followed by the name of the sub-graph,

action - the name of the action routine associated with the node,

operand - an operand for the action routine

forward - i RETURN implies a forward return,
ii the name of the forward node

alternate - i the name of the alternate node,
ii '#' followed by an error number,
iii RETURN indicates a successful return,
iv #RETURN indicates an unsuccessful return.

The complete syntax description used by AEJCL is given in Appendix 8 as an illustration of the use of the simple language which may in fact be used to describe the syntax of languages other than JCL.

2.7 Error Handling in AEJCL (JERROR)

On detecting an error either in the node matching routine or in one of the action routines, control is passed to the error handling routine (JERROR) which does the following:

1 If the card in which the error was detected has not been printed, it is printed.

2 An error diagnostic number is generated on the next line of printing which has the character '$' under the column in which the error was detected.

3 The graph searching routine and the input scanning routines (JCLSCAN,JCLNEXT) are initialised and processing begins again with the next card.

For a complete list of error messages see Appendix 4.

2.8 Keywords

The keyword symbol table is set up by a simple SNOBOL program (see Appendix 3) into which the internal codes of the characters are fed followed by the keywords on cards in the form:

```
$keyword
   or
$keycode keyword1 keyword2 ... keywords
```

where in the former case the keyword has its own name as its internal keycode and in the latter case the keywords 'keyword1' to 'keywords' have the same internal keycode.

The top bit in the keycode word of a symbol table entry is set once a keyword, that may only appear once as a statement, has been recognised and this bit is cleared on the recognition of the end of the statement.

2.9 Action Routines

The action routines are used in AEJCL to:

1 check extra constraints on the syntax of statements,

2 change the mode of JCLSCAN and

3 perform control functions associated with the control cards.

Summary of Action Routines

JAØ2 - saves the operand as the type of statement.

JAØ3 - initialises ready for the next statement.

JAØ7 - saves the fact that a JCL statement had a name field.

JA1Ø - null JCL statement, saves the statement type and initialises ready for the next statement.

JA11 - entry on recognition of a JOB statement, checks for a name field on the JOB card and saves the statement type.

JA14 - entry on a PEND statement, checks that there is no name field and saves the statement type.

JA31 - changes the mode of operation of JCLSCAN to the operand value.

JA36 - checks for duplication of keywords on JCL statements.

JA45 - gives a not implemented error message.

JA49 - checks the value of a number against the operand and fails if it is out of range.

JA58 - checks that the job class is in the range 'A' to 'O'.

JA62 - checks that the sysout class is in the range 'A' to 'Z' or 'Ø' to '9'.

JA76 - saves the fact that the statement is a procedure execute statement.

JPA1 - checks that the statement is a procedure execute statement.

JTA3 - saves the time in minutes.

JTA5 - saves the time in seconds.

JRA2 - saves the region size.

JCTØ6A - initialises count for number of blanks on a continuation.

JCTØ7A - counts through the blanks on the continuation card.

JCTØ8A - checks the continuation column for a blank.

JDAØ26 - checks for mutually exclusive parameters DISP and SYSOUT.

JSERAØ3 - checks that the volume serial is no more than six characters.

JUTØ2A - checks the unit number.

JEXPA3 - checks the expiry date.

JEXPA6 - checks the retention period.

JDA12Ø - checks the unit number.

JCRØ1A - checks for list or JCL checking mode.

JCRØ4A - sets up the listing space code.

JCRØ6A - set mode to indicate JCL checking.

JCRØ7A - releases control if not a control card.

JCR11A - initialises for the twenty-four character parameter for $CALL facility.

JCR12A - accepts each of the twenty-four characters in the $CALL facility.

JCR13A - sends a $CALL message or command.

JCR14A - sends a program load request and then exits from AEJCL.

JCR19A - end of file action routine.

3. AEJCL MONITOR

3.1 The PDP/9L Link Handler Executive

The PDP/9L operating system (Richardson 1968, 1971) enables the concurrent activity of a number of separate programs by the round-robin sequential execution of the tasks. Each task is given control sequentially and it must return to the operating system to give control to the next task.

As more than one program may be active in the PDP/9L at any time and there is only one card reader and line printer it is necessary to obtain exclusive control over these facilities. This function is achieved in AEJCL by the routine JCLENQUE and the reverse process of freeing the devices for use by other programs is done by JCLDEQUE.

3.2 Task Control by the AEJCL Monitor (JCLBGND,JCLREL)

The operating system of the PDP/9L passes control to the monitor routine JCLBGND which searches a list of task control blocks (TCB) for the first

dispatchable task and then gives control to this task or returns to the operating system if there are no dispatchable tasks. When the task is given control it executes until either it is complete, in which case the task is made non-dispatchable and a return is made to the operating system, or it has exhausted its time-slice (approximately 1 milli-second), in which case a return is made to the operating system by the routine JCLREL. In the latter case the task on releasing control still remains dispatchable and will continue executing on a later entry from the operating system.

3.2.1 The task control block (TCB)

This consists of a 1-word address of the resume point of the task which has the top bit set if the task is dispatchable. AEJCL has three intercommunicating tasks which in descending priority order are JCLCARD,JCLPRT and JCLNODE. The priority structure of these tasks is given by the ordering of the TCB's in the list.

3.2.2 Tasks in AEJCL

There are 3 tasks in AEJCL:

1 JCLCARD - reads cards into buffers,

2 JCLPRT - prints lines from buffers, and

3 JCLNODE - analyses the cards and prints out the cards and any error messages.

3.3 Task Synchronisation (JCLWAIT,JCLPOST)

All internal (task-to-task) and external (interrupt handler-to-task) communication in AEJCL is done through event control block (ECB) and the routines JCLWAIT and JCLPOST. When a task wants to indicate to another task that it has completed a particular activity, which it must complete before the other task can do any more associated with that activity, it posts an ECB on which the second task waits. In practice an ECB consists of two words:

1 The status of the ECB, and

2 the address of the TCB for the task waiting on the ECB,

and the routines JCLWAIT and JCLPOST have the following structure:

```
JCLWAIT:   ECB status: = ECB status + 1 ;
           if ECB status ≠ ø then return;
           set TCB of task issuing wait non-dispatchable
           release control to background;
JCLPOST:   ECB status: = ECB status - 1 ;
           if ECB status ≠ -1 then return;
           set TCB of task waiting on ECB dispatchable;
           release control to background.
```

Thus the values of the ECB status word may be interpreted as:

ECB status = Ø → the task whose TCB address is contained in the second word of the ECB is waiting on the ECB and is non-dispatchable.

ECB status = -1 → the task is currently active.

ECB status = -n → the task is currently active and has n-1 more activations to process.

3.3.1 An example of the use of wait and post

For further explanation of the operation of the primitives wait and post the operation of the card reading task in AEJCL will be outlined as an example:

```
jclcard:          call get next buffer
                  set up buffer pointers and counters for the card reader
                       interrupt routine
                  initiate card reading and enable card reading interrupt
                       routine
                  call wait (jclcdecb)
*                 when control returns to this point a card has been read
                  add card to queue of cards to be processed
                  call post (jclndecb)
                  go to jclcard
*                 get next buffer routine
get next buffer:  call wait (jclbfecb)
*                 when control returns to this point there is a free
                       buffer
                  remove the first buffer from the queue
                  return (buffer address).
```

The interrupt servicing routine for the card reader posts jclcdecb when a card has been read and a routine called jclfrbuf posts jclbfecb when a buffer is freed, while the routine jclnode waits on the ECB jclndecb for card processing.

3.3.2 ECBs used in AEJCL

JCLCDECB
- synchronises the card reading task and the card reader interrupt handler
- posted by JCLCI - the card reader interrupt handler
- waited on by JCLCARD - the card reading task

JCLPTECB
- ECB for the synchronisation of the printer interrupt handler and the line printing task
- waited on by JCLPRT - the line printing task

- posted by JCLPI - the printer interrupt handler

JCLNDECB - ECB for the synchronisation of the card reading task and the card processing task
- waited on by JCLNODE - the card processing task
- posted by JCLCARD - the card reading task

JCLBFECB - ECB for synchronisation of the use of input/output buffers
- waited on by JCLBUFER - the get buffer routine
- posted by JCLFRBUF - the free buffer routine

JCLLNECB - ECB for the synchronisation of the card processing routine and the line printing task
- waited on by JCLPRT - the line printing task
- posted by JCLNODE - the card processing task

JCLEBECB - ECB for the synchronisation of the use of the error message buffer
- waited on by JCLNODE when it finds an error
- posted by JCLFRBUF when error message printed

JCLDECB - ECB for the synchronisation of the reading of control card. When a control card is read the card reading routine cannot read any more cards until the card has been further analysed.
- waited on by JCLCARD
- posted by JCLNODE when the analysis of the card shows that more cards can be read.

4. ACKNOWLEDGEMENT

I would like to thank Mr. R. P. Backstrom for valuable assistance and explanation of the operation of the PDP/9L monitor system and input/output devices.

5. REFERENCES

1. IBM System/36Ø Operating System (1970): Job Control Language - IBM System Reference Library Form C28-6539-9.
2. Digital Equipment Corporation (1968) - PDP/9L User's Manual - Form DEC-9L-GRVA-D.
3. Richardson, D.J. (1968) - A Generalised Computer to Computer Link - AAEC/TM485.
4. Richardson, D.J. (1971) - Generalised Computer to Computer Communication - Ph.D. Thesis, UNSW (unpublished).
5. Barry, J.M. (1972) - AESYNTAX - A FORTRAN Syntax Analysis System for the PDP/9L - AAEC/E251.

APPENDIX 1

JCLMACRO – THE DIRECTED GRAPH LANGUAGE TRANSLATOR

```
//JCLMACRO EXEC  SNOBOL,REGION=240K
//GO.FT07F001 DD SYSOUT=A
          &DUMP = 1
          &ANCHOR = 1
         TABLE = TABLE(256,10)
          ALPHANUM = 'ABCDEFGHIJKLMONPQRSTUVWXYZ0123456789'
          BLNK = SPAN(' ') \ NULL
          NUM = '0123456789'
          LABP = BREAK(' ') . LABEL SPAN(' ')
          SYMP = "'" (LEN(1) BREAK("'")) . SYM "'" NULL . NONT \
+               '#' . NONT SPAN(ALPHANUM) . SYM \
+               SPAN(ALPHANUM) . SYM NULL . NONT \
+               NULL . SYM NULL . NONT
          FORP = '->' BLNK ('RETURN' . FRET NULL . FOR \
+               SPAN(ALPHANUM) . FOR NULL . FRET ) \
+               NULL . FRET NULL . FOR
          ALTP  = '#' . ALTI ('RETURN' . ARET NULL . ERR \
+               SPAN(NUM) . ERR NULL . ARET ) NULL . ALT
+               \ 'RETURN' . ARET NULL . ERR NULL . ALT NULL . ALTI \
+               SPAN(ALPHANUM) . ALT NULL . ERR NULL . ALTI NL . ARET \
+               NULL . ALT NULL . ERR NULL . ALTI NULL . ARET
          ALTP = ';' BLNK ALTP \ NULL . ALT NULL . ERR NULL . ARET
+               NULL . ALTI
          OPERP = '(' BAL . OPER ')' \ NULL . OPER
         ACTP = '(' SPAN(ALPHANUM) . ACT OPERP ')' \ NULL . ACT
+              NULL . OPER
         COM = '/' ARB \ NULL
          MACRO = LABP SYMP BLNK ACTP BLNK FORP BLNK ALTP BLNK COM
+               POS(80)
         NERR = 0
         UPD = TRIM(INPUT)
EQUATES  CARD = INPUT
         CARD '$$$$$'                                  :S(READ)
         OUTPUT = CARD
         PUNCH = IDENT(UPD,'PUNCH') CARD
         CARD SPAN(ALPHANUM) . NAME '=' ARB '/ '
+             (LEN(1) BREAK(' ')) . VAL
         TABLE<VAL> = NAME                             :(EQUATES)
READ     CARD       = INPUT                            :F(FEND)
         OUTPUT     = CARD
         CARD '/'                                      :S(COMMENT)
         ('/' CARD) LEN(80)  . PUNCH
         CARD       MACRO                              :F(ERROR)
         L = 1
         L = DIFFER(ACT) 2
         L = DIFFER(OPER) 3
         SYM = IDENT(SYM) 0                                    :S(P2)
         SYM = IDENT(NONT) TABLE<SYM> '+400000'
P2       LABEL = DIFFER(LABEL) LABEL ','
         (LABEL '          ') LEN(9) . LABEL
         PUNCH = LABEL SYM
         ALT   = DIFFER(ERR) ERR '+400000'                     :S(P1)
         ALT = DIFFER(ARET) '600000'                           :F(P0)
         ALT = IDENT(ALTI) '700000'                            :(P1)
P0       ALT = IDENT(ALT) '.+' L + 1
P1       PUNCH = '          ' ALT
         FOR = IDENT(FOR,FRET) '.+' L
         FOR = DIFFER(FRET) '200000'
         FOR = DIFFER(ACT) FOR '+400000'
         PUNCH = '          ' FOR
         PUNCH = DIFFER(ACT) '          ' ACT              :F(READ)
         PUNCH = DIFFER(OPER) '          ' OPER            :(READ)
COMMENT  PUNCH     = CARD                          :(READ)
ERROR    OUTPUT    = '***************** INVALID MACRO ****************'
         NERR = NERR + 1                                       :(READ)
FEND     OUTPUT = '** END OF JOB ** - NO OF ERRORS WAS ' NERR
END
```

APPENDIX 2

THE JCLSCAN INTERNAL CODES

```
BLANK=   Ø                 /
DIG=     1                 / DIG
NAME=    DIG+12            / NAME
DOLLAR=  NAME+32           / $
HASH=    DOLLAR+1          / #
AT=      DOLLAR+2          / @
DOT=     DOLLAR+3          / .
MINUS=   DOT+1             / -
LB=      DOT+2             / (
RB=      LB+1              / )
COMMA=   LB+2              / ,
ASTER=   LB+3              / *
SLASH=   LB+4              / /
EQUALS=  LB+5              / =
QUOTE=   LB+6              / '
AMPER=   LB+7              / &
DOUBLEQ= LB+1Ø             / "
COLON=   LB+11             / :
QUEST=   LB+12             / ?
GT=      LB+13             / >
UNDERSC= LB+14             / ←
PERCENT= LB+15             / %
NOT=     LB+16             / ]
SEMICLN= LB+17             / ;
EXCLAM=  LB+2Ø             / !
OR=      LB+21             / \
PLUS=    LB+22             / +
LT=      LB+23             / <
CENT=    LB+24             / [
UPARROW= LB+25             / :
EOS=     1ØØ               / EOS
ROLL=    1Ø1               / ROLL
TIME=    1Ø2               / TIME
RD=      1Ø3               / RD
RESTART= 1Ø4               / RESTART
REL=     1Ø5               / REL
EXEC=    11Ø               / EXEC
PROC=    111               / PROC
PEND=    112               / PEND
DD=      113               / DD
K=       114               / K
NOYES=   115               / NOYES
PGM=     116               / PGM
DPRTY=   117               / DPRTY
PARM=    12Ø               / PARM
ACCT=    121               / ACCT
EO=      122               / EO
DATA=    123               / DATA
DDNAME=  124               / DDNAME
DCB=     125               / DCB
DUMMY=   126               / DUMMY
DSNAME=  127               / DSNAME
UNIT=    13Ø               / UNIT
UCS=     131               / UCS
VOLUME=  132               / VOLUME
LABEL=   133               / LABEL
DISP=    134               / DISP
SYSOUT=  135               / SYSOUT
```

```
SPACEX=  136                    / SPACEX
SEP=     137                    / SEP
AFF=     140                    / AFF
FOLD=    141                    / FOLD
VERIFY=  142                    / VERIFY
PRIVATE= 143                    / PRIVATE
RETAIN=  144                    / RETAIN
SER=     145                    / SER
REF=     146                    / REF
LABTYP=  147                    / LABTYP
PASSWORD=150                    / PASSWORD
INOUT=   151                    / INOUT
EXPDT=   152                    / EXPDT
RETPD=   153                    / RETPD
SNOM=    154                    / SNOM
PASS=    155                    / PASS
DKCU=    156                    / DKCU
TRK=     157                    / TRK
CYL=     160                    / CYL
RLSE=    161                    / RLSE
ACM=     162                    / ACM
ROUND=   163                    / ROUND
P=       164                    / P
DEFER=   165                    / DEFER
LRECL=   166                    / LRECL
DSORG=   167                    / DSORG
DORG=    170                    / DORG
RECFM=   171                    / RECFM
FM=      172                    / FM
BLKSIZE= 173                    / BLKSIZE
BUFNO=   174                    / BUFNO
UNITYPE= 175                    / UNITYPE
UCSPARM= 176                    / UCSPARM
JOB=     177                    / JOB
MSGLEVEL=200                    / MSGLEVEL
COND=    201                    / COND
PRTY=    202                    / PRTY
MSGCLASS=203                    / MSGCLASS
TYPRUN=  204                    / TYPRUN
HOLD=    205                    / HOLD
CLASS=   206                    / CLASS
REGION=  207                    / REGION
BUFL=    210                    / BUFL
CALL=    211                    / CALL
JCL=     212                    / JCL
LIST1=   213                    / LIST1
LIST2=   214                    / LIST2
OUTLIM=  215                    / OUTLIM
EOF=     216                    / EOF
```

APPENDIX 3

JCLKEY – THE KEYWORD TABLE PROCESSOR

```
//JCLKEY EXEC  SNOBOL,REGION=240K
//GO.FT07F001 DD SYSOUT=A
          &DUMP = 1
          &ANCHOR = 1
          TABLE = TABLE(256,10)
          TABLE<' '> = 'BLANK'
          NUM = '0123456789'
          ALPHA = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
          ALPHANUM = ALPHA NUM
          PAT = '$' SPAN(ALPHANUM) . NAME SPAN(' ') REM . CARD
EQUATES CARD = INPUT
          CARD '$$$$$'                                  :S(READ)
          OUTPUT = CARD
          PUNCH = CARD
          CARD SPAN(ALPHANUM) . NAME '=' ARB '/ '
+               LEN(1) . VAL
          TABLE<VAL> = NAME                             :(EQUATES)
READ      CARD = INPUT                                  :F(FEND)
          OUTPUT = CARD
          CARD '/'                                      :S(COMMENT)
          CARD '&' =                        :S(OTHER)
          CARD PAT                                      :F(ERROR)
          PUNCH = '/' NAME ' ' CARD
          CARD = IDENT(CARD) NAME ' '
P2        CARD SPAN(ALPHANUM) . CNAME SPAN(' ') REM . CARD
          ( CNAME '         ') LEN(1) . C1
+                 LEN(1) . C2 LEN(1) . C3 LEN(1) . C4
+                 LEN(1) . C5 LEN(1) . C6 LEN(1) . C7 LEN(1) . C8
          PUNCH = '          ' TABLE<C1> '*100+' TABLE<C2>
          PUNCH = '          ' TABLE<C3> '*100+' TABLE<C4>
          PUNCH = '          ' TABLE<C5> '*100+' TABLE<C6>
          PUNCH = '          ' TABLE<C7> '*100+' TABLE<C8>
          PUNCH = '          ' NAME
                IDENT(CARD)                 :F(P2) S(READ)
ERROR     '********** INVALID KEYWORD DEFINITION **********************'
          NERR = NERR + 1
COMMENT   PUNCH = CARD                                  :(READ)
OTHER     OUTPUT = CARD
          PUNCH = CARD                      :(READ)
FEND      OUTPUT = DIFFER(NERR) ' NUMBER OF ERRORS WAS ' NERR
          OUTPUT = ' ** END OF JOB **'
END
```

APPENDIX 4

AEJCL ERROR MESSAGES

JOB CONTROL LANGUAGE(JCL) ERROR MESSAGES.

1. '//' NOT FOLLOWED BY EITHER '*', A NAME OR BLANKS.
2. INVALID OPERATION - NOT ONE OF JOB, EXEC, DD, PROC, PEND. (SEE NOTE 1.)
3. PEND STATEMENT MAY NOT HAVE ANY OPERANDS OR COMMA OR BLANK EXPECTED ON EXEC, JOB, DD OR PEND.
4. EITHER A BLANK OR '.DDNAME' EXPECTED AFTER '//NAME'
5. NAME EXPECTED AFTER '//NAME.'
6. '//NAME.DDNAME' MUST BE FOLLOWED BY A BLANK.
7. '//NAME JOB' MUST BE FOLLOWED BY A BLANK.
10. LOCATION OR 'PIGEON' HOLE NUMBER ON JOB CARD EXPECTED.
11. ',' OR CLOSING ')' OF ACCOUNT NUMBER FIELD OF JOB CARD EXPECTED.
12. ',' AFTER ACCOUNTING FIELD ON JOB CARD EXPECTED.
13. PROGRAMER'S NAME ON JOB CARD EXPECTED.
14. A VALID JOB CARD KEYWORD (SEE NOTE 2) OR CONTINUATION EXPECTED.
15. '=' EXPECTED AFTER 'MSGLEVEL' ON JOB CARD.
16. INVALID MESSAGE CODE IN MSGLEVEL PARAMETER - ONLY '0' OR '1' ALLOWED.
17. CLOSING ')' AFTER MSGLEVEL PARAMETER EXPECTED.
20. '=' EXPECTED AFTER 'PRTY' ON JOB CARD.
21. PRTY MUST BE INTEGER BETWEEN 0 AND 13.
22. '=' EXPECTED AFTER 'MSGCLASS' ON JOB CARD
23. MSGCLASS MUST BE A LETTER (A-Z) OR A DIGIT (0-9).
24. '=' EXPECTED AFTER 'TYPRUN'.
25. 'HOLD' EXPECTED AFTER 'TYPRUN='
26. '=' EXPECTED AFTER 'CLASS'
27. JOB CLASS MUST BE A LETTER BETWEEN 'A' AND 'O'.
30. INTEGER BETWEEN 0 AND 4095 EXPECTED AS CODE IN COND PARAMETER.
31. CLOSING ')' EXPECTED AFTER '(CODE,OPER' IN COND PARM ON JOB CARD.
32. CLOSING ')' OF COND PARAMETER EXPECTED OR ',' EXPECTED.
33. OPENING '(' OF COND SUB-PARAMETER EXPECTED OR AS FOR MSG 30.
34. OPENING '(' ON NEXT COND SUB-PARAMETER EXPECTED.
35. ',' EXPECTED AFTER NUMERIC CODE IN COND SUB-PARAMETER.
36. RELATIONAL OPERATOR (EQ,NE,LT,LE,GT,GE) EXPECTED IN COND PARM.
37. INVALID TIME FIELD ON JOB CARD.
40. CLOSING ')' OF TIME PARAMETER EXPECTED OR ',' EXPECTED.
41. INVALID MINUTES FIELD OF TIME PARAMETER OR',' EXPECTED.
42. INVALID MINUTES FIELD OF TIME PARAMETER.
43. INVALID REGION SPECIFICATION ON JOB CARD.
44. '=' EXPECTED AFTER REGION.
45. REGION VALUE MUST BE INTEGER VALUE.
46. 'K' EXPECTED AFTER REGION SPECIFICATION.
47. INVALID ROLL PARAMETER ON JOB STATEMENT.
50. ROLL PARAMETER MUST BE EITHER 'YES' OR 'NO'
51. ',' EXPECTED IN ROLL PARAMETER.
52. ROLL PARAMETER MUST BE EITHER 'YES' OR 'NO'
53. ')' OF ROLL PARAMETER EXPECTED.
54. OPENING LEFT BRACKET OF ACCOUNT NUMBER ON JOB CARD EXPECTED.
55. ACCOUNT NUMBER ON JOB CARD EXPECTED.
56. MSGLEVEL STATEMENTS FIELD MUST BE EITHER '0','1','2' OR NULL.
57. MSGLEVEL STATEMENTS FIELD MUST BE EITHER '0','1','2'.
60. '=' EXPECTED AFTER 'COND'
61. OPENING '(' EXPECTED AFTER 'COND=' ON JOB CARD.
62. '=' EXPECTED FOR TIME PARAMETER.
63. INTEGER BETWEEN 1 AND 59 SPECIFYING SECONDS EXPECTED.
64. '=' EXPECTED IN ROLL PARAMETER.
65. OPENING '(' EXPECTED AFTER 'ROLL='

66. '=' EXPECTED AFTER PGM ON EXEC STATEMENT.
67. PROGRAM NAME OR REFERBACK EXPECTED AFTER 'PGM='
70. '=' EXPECTED AFTER 'PROC' ON EXEC STATEMENT.
71. PROCEDURE NAME EXPECTED.
72. A VALID EXEC CARD KEYWORD (SEE NOTE 3) OR CONTINUATION EXPECTED.
73. '=' EXPECTED AFTER 'DPRTY' ON EXEC STATEMENT.
74. DPRTY MUST BE AN INTEGER BETWEEN 0 AND 15.
75. '=' EXPECTED AFTER PARM ON EXEC STATEMENT.
76. ',' OR ')' EXPECTED IN PARM FIELD.
77. PARM FIELD MUST BE A STRING IN QUOTES, AN ALPHAMERIC STRING, OR STRINGS IN BRACKETS
100. INVALID ROLL PARAMETER ON EXEC STATEMENT.
101. INVALID REGION PARAMETER ON EXEC STATEMENT.
102. INVALID TIME PARAMETER ON EXEC STATEMENT.
103. '=' EXPECTED AFTER 'COND' ON EXEC STATEMENT.
104. 'EVEN' OR 'ONLY' OR COND SUB-FIELD '(CODE,OPER,STEP)' EXPECTED.
105. COND SUB-FIELD '(COND,OPER,STEP)' EXPECTED.
106. ')' OR ',' EXPECTED IN COND PARAMETER.
107. COND SUB-FIELD '(CODE,OPER,STEP)' OR 'EVEN' OR 'ONLY' EXPECTED.
110. CODE OF 'COND' SUB-PARAMETER BETWEEN 0 AND 4095 EXPECTED.
111. STEPNAME EXPECTED IN COND SUB-PARAMETER.
112. STEPNAME EXPECTED IN COND SUB-PARAMETER.
113. ',' OR CLOSING ')' EXPECTED.
114. '=' EXPECTED AFTER 'LABEL' ON DD STATEMENT.
115. STEPNAME OF PROCEDURE STEP EXPECTED.
116. '.' AFTER '*' IN REFERBACK EXPECTED.
117. NAME EXPECTED AFTER '*.' IN REFERBACK.
120. NAME EXPECTED AFTER '*.NAME.' IN REFERBACK.
121. NAME EXPECTED AFTER '*.NAME.NAME.' IN REFERBACK.
122. BLANK EXPECTED AFTER 'DD' ON DD STATEMENT.
123. '=' EXPECTED AFTER 'DDNAME' ON DD STATEMENT.
124. NAME EXPECTED AFTER 'DDNAME=' ON DD STATEMENT.
125. DCB PARAMETER OR CONTINUATION EXPECTED.
126. '=' EXPECTED AFTER 'DCB' ON DD STATEMENT.
127. DCB SUB-PARAMETER EXPECTED.
130. ')' EXPECTED FOR DCB PARAMETER.
131. DCB SUB-PARAMETER EXPECTED.
132. VALID DD STATEMENT KEYWORD (SEE NOTE 4) OR CONTINUATION EXPECTED.
133. '=' EXPECTED AFTER DSNAME ON DD STATEMENT.
134. NAME EXPECTED IN DSNAME FIELD (MAY ALSO BE REFERBACK OR IN QUOTES)
135. MEMBER NAME IN DSNAME FIELD EXPECTED.
136. ')' AFTER MEMBER NAME IN DSNAME FIELD EXPECTED.
137. '=' EXPECTED AFTER 'UCS' ON DD STATEMENT.
140. SYMBOLIC PARAMETERS IN IN-LINE PROCEDURES ARE NOT SUPPORTED.
141. UCS PARAMETER, 'FOLD', 'VERIFY' OR ')' EXPECTED IN UCS PARAMETER.
142. UCS PARAMETER (QN,TN,IN,PN) EXPECTED.
143. '=' EXPECTED AFTER 'VOL' OR 'VOLUME' ON DD STATEMENT.
144. 'PRIVATE', 'SER' OR '(' EXPECTED IN VOLUME PARAMETER ON DD STATEMENT.
145. 'PRIVATE', 'RETAIN', NUMBER OF VOLUMES OR REF SUB-PARM OF VOL FIELD EXPECTED
146. '=' EXPECTED AFTER 'SER'.
147. NAME OR '(' EXPECTED AFTER 'SER=' (SEE MSG 150)
150. VOLUME SERIAL NAME OF NOT MORE THAN 6 CHARACTERS EXPECTED.
151. ')' EXPECTED IN SER FIELD OR ',' EXPECTED.
152. '=' EXPECTED AFTER 'REF'
153. REFERBACK OR DATA SET NAME EXPECTED AFTER 'REF='
154. '(' OR FILE NUMBER OR EXPIRY DATE OR RETENTION PERIOD EXPECTED.
155. FILE NUMBER, LABEL TYPE (NL,SL,SUL,NSL,BLP) OR 'IN' OR 'OUT' EXPECTED.
156. '=' EXPECTED AFTER 'EXPDT' IN LABEL FIELD.
157. EXPIRY DATE OF FORM 'YYDDD' EXPECTED.

```
160. '=' EXPECTED AFTER 'RETPD'.
161. RETENTION PERIOD IN DAYS EXPECTED.
162. '=' EXPECTED AFTER 'DISP' ON DD STATEMENT.
163. '(', SHR,NEW,OLD,MOD IN DISP PARAMETER EXPECTED.
164. ')', PASS,DELETE,KEEP,CATLG,UNCATLG EXPECTED IN DISP PARM.
165. '=' EXPECTED AFTER 'SPACE' ON DD STATEMENT.
166. '(' EXPECTED AFTER 'SPACE='.
167. SPACE SPECIFICATION ('CYL,'TRK' OR NUMBER OF BYTES) EXPECTED.
170. ',' EXPECTED AFTER SPACE SPECIFICATION.
171. '=' EXPECTED IN 'SYSOUT' FIELD.
172. PRIMARY SPACE ALLOCATION EXPECTED.
173. EXTENT NUMBER OR DIRECTORY BLOCK NUMBER EXPECTED.
174. ')' EXPECTED IN SPACE SUB-PARM.
175. SPACE SIZE,',', RLSE,ALX,MAXIG,CONTIG OR ROUND EXPECTED.
176. ')' EXPECTED AFTER SPACE FIELD.
177. SYSOUT CLASS EXPECTED-MUST BE LETTER (A-Z) OR DIGIT (0-9)
200. CONTINUATION CARD EXPECTED.
201. CONTINUATION CARD EXPECTED - MUST HAVE BLANK IN COLUMN 3.
202. CONTINUATION CARD EXPECTED.
203. ATTEMPT TO USE STEP OVERWRITING ON A 'PGM=' EXEC CARD.
204. SEP PARAMETER INVALID.
205. DDNAME EXPECTED IN SEP PARM.
206. ',' OR ')' EXPECTED IN SEP PARM.
207. DDNAME EXPECTED IN SEP PARM.
210. '=' EXPECTED AFTER 'AFF'
211. DDNAME EXPECTED AFTER 'AFF='
212. '=' EXPECTED AFTER 'UNIT' ON DD STATEMENT.
213.
214. INVALID UNIT SPECIFICATION.
215. ')' EXPECTED IN UNIT FIELD OF DD STATEMENT.
216. UNIT TYPE OR NUMBER, 'AFF', 'SEP' OR '(' EXPECTED.
217. '=' EXPECTED AFTER 'DCB'
220. DCB SUB-PARM, '(', OR REFERBACK EXPECTED.
221.
222. ',',')' OR DCB SUB-PARM EXPECTED.
223. '=' EXPECTED AFTER LRECL IN DCB ON DD STATEMENT.
224. INTEGER LRECL EXPECTED.
225. '=' EXPECTED AFTER DSORG IN DCB.
226. DATA SET ORGANISATION (PS,PO,DA,IS) EXPECTED.
227. '=' EXPECTED AFTER RECFM IN DCB.
230. RECORD FORMAT (F,B,S,M,ETC) EXPECTED.
231. SECOND '/' EXPECTED IN COLUMN 2.
232. 'JOB' OR 'PROC' STATEMENT HAS NO NAME FIELD.
233. 'PEND' STATEMENT CANNOT HAVE A NAME FIELD.
234. KEYWORD PREVIOUSLY DEFINED ON STATEMENT.
235. NOT IMPLEMENTED.
236. DISP AND SYSOUT MUTUALLY EXCLUSIVE PARAMETERS.
237. VOL. SER NAME HAS A MAXIMUM OF 6 CHARS.
240. '=' EXPECTED AFTER A SYMBOLIC PARAMETER NAME.
241.
242. BLANK EXPECTED AFTER 'EXEC' ON EXEC STATEMENT.
243. BLANK EXPECTED AFTER 'PROC' ON PROC STATEMENT.
244. SYMBOLIC PARAMETER OR CONTINUATION EXPECTED ON PROC STATEMENT.
245. '=' EXPECTED AFTER 'OUTLIM'
246. INTEGER OUTLIM EXPECTED.
247.
1000. INTEGER TOO LARGE - OVERFLOW.
```

NOTES:

A 'NAME' THROUGHOUT THESE ERROR MESSAGES REFERS TO ANY VALID NAME IN JCL. A VALID NAME IN JCL CONSISTS OF NOT MORE THAN 8 CHARACTERS THE FIRST OF WHICH MUST BE EITHER ALPHABETIC OR NATIONAL('@','$','#') AND THE REMAINING MUST BE EITHER ALPHABETIC, NUMERIC OR NATIONAL.

1. THIS ERROR MAY APPEAR ON THE CONTINUATION CARDS OF A STATEMENT WITH A PREVIOUS ERROR.
2. VALID JOB STATEMENT KEYWORDS ARE:
   - MSGLEVEL
   - COND
   - PRTY
   - MSGCLASS
   - TYPRUN
   - TIME
   - CLASS
   - REGION
   - ROLL
   - RD
   - RESTART
3. VALID EXEC STATEMENT KEYWORDS ARE:
   - PGM
   - PROC
   - DPRTY
   - PARM
   - COND
   - ROLL
   - REGION
   - TIME
   - ACCT
   - RD
   - RESTART
4. VALID DD STATEMENT KEYWORDS:
   - DSNAME OR DSN
   - UNIT
   - UCS
   - VOLUME OR VOL
   - DCB
   - LABEL
   - DISP
   - SYSOUT
   - SPACE
   - SEP
   - AFF
   - OUTLIM

APPENDIX 5

JCLWAIT AND JCLPOST – TASK SYNCHRONISATION

```
/              E C B
/         2 WORDS :-
/         1. STATUS OF THE ECB
/         Ø  - WAITING
/         -1 - PROCESSING
/         -N - PROCESSING - N-1 REQUESTS OUTSTANDING.
/              P O S T
/         DECREMENT OF PARMAMETER ADDRESSED ECB
/              W A I T
/         INCREMENT OF PARAMETER ADDRFESSED ECB.
/
/         CONDITIOND :
/              IF AN ECB CAN BE WAITED ON OR POSTED BY AN INTERRUPT ROUTINE THEN
/         IT CANNOT BE POSTED IN THE BACKGROUND BUT MAY BE WAITED ON.
JCLCDECB,Ø-1                              / WAITED ON BY JCLCARD.
         Ø                                / POSTEE BY JCLCI - INTERRUTP.
JCLPTECB,Ø-1                              / WAITED ON BY JCLPRT.
         Ø                                / POSTED BY JCLPI - INTERRUPT.
JCLNDECB,Ø-1                              / WAITED ON BY JCLNODE.
         Ø                                / POSTED BY JCLCARD.
JCLBFECB,Ø-JCLNBUF-1                      / WAITED ON BY JCLBUFFER
         Ø                                / WAITED ON BY JCLBUFER (JCLPRT)
/                                         / POSTED BY JCLFRBUF (JCLPRT)
JCLLNECB,Ø-1                              / LINE PRINTER WAITING FOR LINE
         Ø                                / POSTED BY JCLNODE IN ROUTINE JCLGET.
JCLEBECB,Ø-1-1                            / ERROR BUFFER ECB
         Ø                                / WAITED BY JERROR POSTED BY FRBUF.
JCLDECB, Ø-1                              / ECB TO SYNCH THE $ CONTROL.
         Ø                                / CARDS.
JCLEFECB,Ø-1                              / THE END-OF-FILE ECB.
         Ø                                / TO CONTROL END-OF-FILE
/              J C L W A I T
JCLWAIT, Ø                                / THE ENTRY POINT.
         LAC I JCLWAIT                    / LOAD ECB ADDRESS.
         DAC   JWAIT1S                    / SAVE THE ECB ADDRESS.
         ISZ   JCLWAIT                    / POINT TO THE RESUME POINT.
         DAC   JWAIT2S                    / SAVE POINTER TO ECB.
         ISZ   JWAIT2S                    / POINT TO ECB ADDR. WORD.
         LAC   JCLCTCB                    / LOAD THE CURRENT TCB ADDRESS.
         DAC I JWAIT2S                    / SET IN THE ADDRESS WORD.
         LAC   JCLWAIT                    / LOAD THE RETURN POINT.
         AND   JWAIT1F                    / AND OUT THE TOP BITS.
         ISZ I JWAIT1S                    / W A I T   ON ECB.
/              D I S P A T C H A B L E
         XOR   JPOST2F                    / INDICATE DISPATCHABLE.
         DAC I JCLCTCB                    / SAVE AS ENW TCB POINTER WORD.
         JMP I JCLBGND                    / -------> RETURN TO BACKGROUND.
JWAIT1F, Ø17777                           / MASK   NON-DISPATCHABLE.
JWAIT1S, Ø                                / TEMPORARY.
JWAIT2S, Ø                                / TEMPORARY.
/              J C L P O S T              /
JCLPOST, Ø                                / THE ENTRY POINT.
         LAC I JCLPOST                    / LOAD THE ECB ADDRESS
         ISZ   JCLPOST                    / INCREMENT OVER THE PARM.
         DAC   JWAIT1S                    / SAVE THE ECB ADDRESS.
         LAC I JWAIT1S                    / LOAD THE EXB STATUS.
         SZA                              / IS IT WAITING ?
         JMP   JPOST1Ø                    / ---> NO.

/              W A I T I N G
         TAD   JPOST1F                    / POST THE ECB.
         DAC I JWAIT1S                    / SAVE THE ECB SYATUS.
         ISZ   JWAIT1S                    / POINT TO THE TCB ADDRESS.
         LAC I JWAIT1S                    / LOAD THE TCB
         DAC   JWAIT1S                    / SAVE TCB ADDRESS.
         LAC I JWAIT1S                    / LOAD THE TCB.
         XOR   JPOST2F                    / INDICATE DISPATCHABLEH
         DAC I JWAIT1S                    / SAVE.
         LAC   JCLPOST                    / LOAD THE RETURN POINT.
         JMP   JREL1Ø                     / RELEASE CONTROL.
/              N O T    W A I T I N G
JPOST1Ø, TAD   JPOST1F                    / POST THE ECB.
         DAC I JWAIT1S                    / SAVE.
         LAC   JCLPOST                    / LOAD THE RETURN POINT.
         JMP   JREL1Ø                     / ---> R E L E A S E
JPOST1F, Ø-1                              / -1    DECREMENT.
JPOST2F, 400000                           / THE DISPATCHABLE BIT.
```

APPENDIX 6

JCLBGND AND JCLREL – TASK CONTROL ROUTINES

```
/               T C B
/         1. WORD  :-   THE CESUME ADDRESS.
/               TOP BIT ON = DISPATCHABLE.
JCLCTCB, Ø                              / THE ADDRESS OF THE CURRENT TCB.
JCLTCB=  .                              / THE TCB CHAIN.
JCLCDTCB,JCLCARD+400000                 / CARD READER TCB.
JCLPTTCB,JCLPRT+400000                  / LINE PRINTER TCB.
JCLNDTCB,JCLNODE+400000                 / ANALYZER TCB.
JCLEFTCB,JCLEFILE+400000                / THE END-OF-FILE CHECKER
         Ø                              / THE END OF THE TCB CHAIN.
/               J C L B G N D           /
JCLBGND, Ø                              / THE ENTRY POINT.
         LAW   JCLTCB                   / LOAD ADDRESS OF 1ST TCB.
         DAC   JCLCTCB                  / SET AS THE CURRENT TCB.
JBGND1Ø, LAC I JCLCTCB                  / LOAD THE TCB.
         SPA                            / IS IT DISPATCHABLE ?
         JMP   JBGND2Ø                  / ---> YES, RESUME THE TASK.
         ISZ   JCLCTCB                  / POINT TO THE NEXT TCB.
         SZA                            / ARE THERE ANY MORE TCB'S ?
         JMP   JBGND1Ø                  / ---> YES.
/               NO ACTIVE TCB'S.
         JMP I JCLBGND                  / -------> RETURN TO BACKGROUND.
JBGND2Ø, DAC   JBGND1S                  / SAVBE THE RESUME POINT.
         JMP I JBGND1S                  / ---> RESUME
JBGND1S, Ø                              / TEMPORARY FOR THE RESUME POINT.
/               J C L R E L             /
JCLREL,  Ø                              / THE ENTRY POINT.
         LAC   JCLREL                   / LOAD THE RETURN POINT.
JREL1Ø,  RAL                            / SHIFT LEFT.
         CLL CML RAR                    / SHIFT IN THE DISPATCHABLE BIT.
         DAC I JCLCTCB                  / PUT IN THE CURRENT TCB.
         JMP I JCLBGND                  / ---> R E L E A S E.
```

APPENDIX 7

JCLNODE – THE SYNTAX CHECKING ALGORITHM

```
/                G R A P H   M A T C H
JCLNODE= .                              / THE ENTRY POINT.
JN10,    JMS     JA03                   / I N  I T I A L I S E .
         LAW     JCLGRAPH               / THE THE START OF THE GRAPH.
JN15,    DAC     JN1A                   / DEPOSIT THE CURRENT NODE.
         JMS     JCLSCAN                / GET THE NEXT INPUT ITEM.
JN30,    LAC I   JN1A                   / LOAD THE SYMBOL.
         ISZ     JN1A                   / POINT TO THE ALTERNATE.
         SMA                            / IS IT A TERMINAL ?
         JMP     JN70                   / ---> NO.
         SAD     JSCAN1S                / IS IT THE SAME AS WHAT WE GOT.
         JMP     JN50                   / ---> YES.
         SAD     JN1F                   / IS THE NODE A NAME ?
         JMP     JN45                   / ---> YES.
/                M I S M A T C H
JN40,    LAC I   JN1A                   / LOAD THE ALTERNATE.
         SPA SNA                        / IS THERE AN ALTERNATE ?
         JMP     JN60                   / ---> NO.
/                A L T E R N A T E
         DAC     JN1A                   / SET CURRENT NODE = ALTERNATE.
         JMP     JN30                   / ---> LOOP FOR THIS NEW ALTERNATE.
/                N A M E   N O D E
JN45,    LAC     JSCAN1S                / LOAD THE CHAR.
         AND     JKEY1F                 / AND OUT THE TOP BIT.
         TAD     JN2F                   / SUBTRACT EOS CODE.
         SPA                            / IS IT A KEYWORD.
         JMP     JN40                   / ---> NO, MISMATCH
/                KEY WORD MATCHES AS A NAME.
/                M A T C H
JN50,    ISZ     JN1A                   / POINT TO THE FORWARD POINTER.
         LAC I   JN1A                   / LOAD THE FORWARD POINTER.
         SMA                            / IS THERE AN ACTION ?
         JMP     JN55                   / ---> NO.
/                A C T I O N
         DAC     JN2A                   / SAVE THE FORWARD POINTER.
         ISZ     JN1A                   / POINT TO THE ACTION ROUTINE ADDRESS.
         LAC I   JN1A                   / LOAD THE ADDRESS OF THE ACTION ROUTINE.
         DAC     JN4A                   / SAVE IT.
         ISZ     JN1A                   / POINT TO THE OPERAND.
         JMS I   JN4A                   / CALL THE ACTION ROUTINE.
         JMP     JN53                   / ---> SUCCESSFUL RETURN.
         LAW     0-3                    / GET THE ADDRESS
         TAD     JN1A                   / OF THE ALTERNATE.
         DAC     JN1A                   / SET AS THE ALTERNATE.
         JMP     JN40                   / ---> LOOOP OVER THE ALTERNATE.
JN53,    LAC     JN2A                   / LOAD THE ADRRESS OF THE FOR PTR.
/                A C T I O N   C O M P L E T E   I F   A N Y
JN55,    RAL                            / SHIFT TO GET THE RETURN BIT.
         SMA RAR                        / IS THERE A RETURN ( RESTORE)
         JMP     JN15                   / ---> TAKE THE FORWARD.
/                M A T C H   F O R W A R D   R E T U R N
         LAW     0-1                    / DECREMENT
         TAD     JN3A                   / THE
         DAC     JN3A                   / STACK POINTER.
         LAC I   JN3A                   / UNSTACK THE POINTER.
         DAC     JN1A                   / SET AS THE CURRENT NODE.
         JMP     JN50                   / ---> MATCH.
/                N O   A L T E R N A T E
```

```
JN60,   RTL                         / SHIFT TO GET RETURN BIT
        SZL                         / IS THERE A RETURN.
        JMP   JN80                  / ---> YES.
/             E R R O R
        RTR                         / RESTORE THE ACCUMULATOR.
        JMS   JERR                  / CALL THE ERROR ROUTINE.
        JMP   JN10                  / ---> START A NEW STATEMENT.
/             N O N - T E R M I N A L   O R   A N Y
JN70,   SNA                         / IS IT NON-TERMINAL.
        JMP   JN50                  / ---> NO, ANY.
        DAC   JN2A                  / SAVE THE NON-TERMINAL ADDRESS.
        LAC   JN1A                  / ADDRESS OF THE CURRENT ALTERNATE.
        DAC I JN3A                  / STACK THE POINTER.
        ISZ   JN3A                  / INCREMENT THE STACK POINTER.
        LAC   JN2A                  / RESTORE THE NON-TERMINAL POINTER.
        DAC   JN1A                  / SET AS THE CURRENT NODE.
        JMP   JN30                  / ---> MATCH THE SUBGRAPH.
/             R E T U R N
JN80,   SPA                         / IS IT SUCCESSFULL.
        JMP   JN100                 / ---> YES.
        LAW   0-1                   / DECREMENT
        TAD   JN3A                  / THE
        DAC   JN3A                  / STACK POINTER.
        LAC I JN3A                  / UNSTACK THE POINTER.
        DAC   JN1A                  / SET AS THE CURRENT NODE.
        JMP   JN40                  / ---> MISMATCH.
/             M I S M A T C H   S U C C E S S F U L   R E T U R N
JN100,  LAW   0-1                   / DECREMENT
        TAD   JN3A                  / THE
        DAC   JN3A                  / STACK POINTER.
        LAC I JN3A                  / UNSTACK THE POINTER.
        DAC   JN1A                  / SET IT AS THE CURRENT NODE.
        ISZ   JN1A                  / SET THE POINTER TO THE FORWARD.
        LAC I JN1A                  / LOAD THE FORWARD POINTER.
        SMA                         / IS THERE AN ACTION ?
        JMP   JN105                 / ---> NO.
        DAC   JN2A                  / SAVE THE FORWARD POINTER.
        ISZ   JN1A                  / SET TO ACTION ROUTINE ADDRESS.
        LAC I JN1A                  / LOAD THE ADDRESS OF THE ACTION ROUTINE.
        DAC   JN4A                  / SAVE IT.
        ISZ   JN1A                  / POINT TO THE POERAND.
        JMS I JN4A                  / CALL THE ACTION ROUTINE.
        LAC   JN2A                  / RESTORE THE FORWARD POINTER.
/             A C T I O N   C O M P L E T E   I F   A N Y
JN105,  RAL                         / SHIFT TO GET RETURN BIT.
        SPA RAR                     / IS IT RETURN (RESTORE)
        JMP   JN100                 / ---> YES.
        DAC   JN1A                  / SET THE NEW FORWARD POINTER.
        JMP   JN30                  / ---> LOOP.
JN1A,   0                           / CURRENT-NODE
JN2A,   0                           / SAVE.
JN3A,   JSTK                        / THE STACK POINTER.
JN4A,   0                           / ACTION ROUTINE ADDRESS SAVE.
JN1F,   NAME+400000                 / CODE FOR NAME WITH TERMINAL BIT.
JN2F,   0-EOS                       / -EOS.
JSTK,   DS    10                    / THE STACK.
```

APPENDIX 8

AEJCL SYNTAX DESCRIPTION

```
/                J C L G R A P H
JCLGRAPH '/'    ; JCR01                        / /
         '/'(JA31(JSCAN20))  -> JG06           / //
         '*'(JA02(7)) ; JG05                   / /*
JG03     EOS(JA03) -> JCLGRAPH                 /
         -> JG03                               /
JG05     (JA02(11)) -> JG03                    /
JG06     '*'(JA02(6)) -> JG03                  / //*
         NAME(JA07) -> JG18                    / //NAME
         #JBLNK1  ; #1                         /
         EOS(JA10(10)) -> JCLGRAPH             / //     NULL
JG11     JOB(JA11(1)) ->JG24                   / //NAME JOB
         EXEC(JA02(2)) -> JG75                 / //NAME EXEC
         PROC(JA11(4))  -> JH22                / //NAME PROC
         PEND(JA14(5)) ->JG16                  / //       PEND
JG15     DD(JA02(3)) -> JDD001 ; #2            / //NAME DD
JG16     ' ' ->JG16                            / //      PEND
JG17     EOS(JA03) -> JCLGRAPH  ; #3           / //      PEND
JG18     ','   ->JG21                          / //NAME.
         #JBLNK1  -> JG11  ; #4
JG21     NAME                  ; #5            / //NAME.DDNAME
         #JBLNK1  -> JG15  ; #6
/                J O B    C A R D
JG24     #JBLNK1   ; #7
         '('   ; #54                           / (
         NAME  ; #55                           / (<ACCT>
         ','   ; JG30                          / (<ACCT>,
         NAME  ; #10                           / (<ACCT>,<LOCN>
JG30     ')'   ; #11                           / (<ACCT>,<LOCN>)
         ','(JA31(JSCAN50))  ;  #12            / (<ACCT>,<LOCN>),
         #JQ01 ->JG34                          / ,'PROG-NAME'
         NAME  ; #13                           / ,PROG-NAME
JG34     ','(JA31(JSCAN20)) ->JG36             / PROG-NAME
         ' '   ->JG03 ;JG17                    /
JG36     MSGLEVEL(JA36)-> JG47                 / MSGLEVEL
         COND(JA36)-> JG63                     / COND
         PRTY(JA36)-> JG55                     / PRTY
         MSGCLASS(JA36) -> JG57                / MSGCLASS
         TYPRUN(JA36) -> JG59                  / TYPRUN
         TIME(JA36) -> JG72                    / TIME
         CLASS(JA36) -> JG61                   / CLASS
         REGION(JA36) ->JG73                   / REGION
         ROLL(JA36) -> JG74                    / ROLL
         RD(JA45)                              / RD
         RESTART(JA45)                         / RESTART
         #JCT01  -> JG36  ;  #14               / CONTINUATION
/                M S G L E V E L
JG47     '='   ; #15                           / MSGLEVEL=
         '('   ; JG54                          / MSGLEVEL=(
         DIG(JA49(-2)) ; JG53                  / MSGLEVEL=(0\1\2
         ','   ; JG52                          / MSGLEVEL=(0\1\2,
JG51     DIG(JA49(-1))      ; #16              / MSGLEVEL=(0\1\2,0\1
JG52     ')'   ->JG34  ; #17                   / MSGLEVEL=(0\1\2,0\1)
JG53     ','   ->JG51  ; #56                   / MSGLEVEL=(,
JG54     DIG(JA49) -> JG34  ; #57              / MSGLEVEL=0\1\2
/                P R T Y
JG55     '='   ; #20                           / PRTY=
         DIG(JA49(-13))  -> JG34 ; #21         / PRTY=0-13
```

```
/               M S G C L A S S
JG57      '='(JA31(JSCAN1Ø))    ; #22     / MSGCLASS=
          (JA62(JSCAN2Ø)) ->JG34 ; #23    / MSGCLASS=A-Z\Ø-9
/               T Y P R U N
JG59      '='    ; #24                    / TYPRUN=
          HOLD -> JG34   ; #25            / TYPRUN=HOLD
/               C L A S S
JG61      '='(JA31(JSCAN1Ø))   ; #26      / CLASS =
          (JA58)  -> JG34   ;  #27        / CLASS=A-O
/               C O N D
JG63      '='    ; #6Ø                    / COND=
          '('    ; #61                    / COND=(
JG65      '('    ; JG7Ø                   / COND=((
          #JCØ1 ; #3Ø                     / COND=((CODE,OPER
          ')'    ; #31                    / COND=((CODE,OPER)
          ','    ; JG69                   / COND=((CODE,OPER),
          '('    -> JG65   ; #34          / COND=((CODE,OPER),(
JG69      ')'    -> JG34   ; #32          / COND=((CODE,OPER),...)
JG7Ø      #JCØ1 ->JG69   ; #33            / COND=(CODE,OPER
/               C O N D    S U B G R A P '
JCØ1      DIG(JA49(-7777))   ; #RETURN  / CODE
          ','    ; #35                    / CODE,
          REL    -> RETURN   ; #36        / CODE,OPER
/               T I M E
JG72      #JTØ1 -> JG34   ; #37           / TIME=(MIN,SEC)
/               T I M E   S U B G R A P H
JTØ1      '='    ; #62                    / =
          '('    ; JTØ8                   / =(
          DIG(JTA3)   ; JTØ7              / =(MIN
          ','    ; JTØ6                   / =(MIN,
JTØ5      DIG(JTA5)   ; #63               / =(MIN,SEC
JTØ6      ')'   -> RETURN   ; #4Ø         / =(MIN,SEC)
JTØ7      ','    ->JTØ5   ; #41           / =(,
JTØ8      DIG(JTA3)   -> RETURN   ; #42   / =MIN
/               R E G I O N
JG73      #JRØ1 -> JG34   ; #43           / REGION=NNNNNK
JRØ1      '='    ; #44                    / =
          DIG(JRA2)   ; #45               / =NNNNN
          K -> RETURN ; #46               / =NNNNNK
/               R O L L
JG74      #JOØ1 -> JG34   ; #47           / ROLL=(YES\NO,YES\NO)
JOØ1      '='    ; #64                    / =
          '('    ; #65                    / =(
          NOYES   ; #5Ø                   / =(NO\YES
          ','    ; #51                    / =(NO\YES,
          NOYES    ; #52                  / =(NO\YES,NO\YES
          ')'    -> RETURN   ; #53        / =(NO\YES,NO\YES)
/               C O N T I N U A T I O N
JCTØ1     ' '    -> JCTØ3                 /
          EOS    -> JCTØ5   ; #RETURN     /
JCTØ3     EOS    -> JCTØ5                 /
          -> JCTØ3                        /
JCTØ5     #JSSØ1    ; #2ØØ                / //
          ' '(JCTØ6A)   ; #2Ø1            /
JCTØ7     ' '(JCTØ7A(JCTØ8))   -> JCTØ7   ; RETURN
/               PAST COLUMN 16
JCTØ8     EOS(JCTØ8A(JCT1Ø))   -> JCLGRAPH
          -> JCTØ8                        /
JCT1Ø     #JSSØ1   ; #2Ø2                 /
JSSØ1     '/'      ; #RETURN              / /
```

```
         DISP(JDA026(JKSYSOUT)) -> JDD078
         SYSOUT(JDA026(JKDISP)) -> JDD110    / SYSOUT
         SPACEX(JA20) -> JDD088             / SPACE
         SEP(JA20) -> JDD112                / SEP
         AFF(JA20) -> JDD113                / AFF
         OUTLIM -> JDD138 ; #132            / OUTLIM
JDD031   ','    ; JDD016                    / ,
         #JCT01  -> JDD018 ; JDD018         / CONTINUATION.
JDD033   '='(JA31(JSCAN40)) ; #133          / DSNAME=
         #JF01 -> JDD031                    / DSNAME=*.STEP.PROC.DDNAME
         #JQ01(JA31(JSCAN20)) ->JDD031      / DSN='STRING'
         '&'                                / DSNAME=&
         '&'                                / DSNAME=&&
         NAME(JA31(JSCAN20)) ; #134         / DSNAME=&&NAME
         '('        ; JDD031                / DSNAME=&&NAME(
         NAME       ; #135                  / DSNAME=&&NAME(NAME
         ')' -> JDD031 ; #136               / DSNAME=&&NAME(NAME)
/                U  C  S
JDD041   '='        ; #137                  / UCS=
         '('        ; JDD049                / UCS=(
         UCSPARM                            / QN IN TN PN
         ','        ; JDD048                / UCS=(QN,
         FOLD                               / UCS=(QN,FOLD
         ','        ; JDD048                / UCS=(QN,FOLD,
         VERIFY                             / UCS=(QN,FOLD,VERIFY
JDD048   ')' -> JDD031  ; #141              / UCS=(QN,FOLD,VERIFY)
JDD049   UCSPARM  ; #142                    / QN IN TN PN
/              V O L U M E
JDD050   '='        ; #143                  / VOL=
         PRIVATE -> JDD031                  / VOL=PRIVATE
         #JSER01 -> JDD031                  / VOL=SER=AAE00N
         '('        ; #144                  / VOL=(
         PRIVATE                            / VOL=(PRIVATE
         ','        ; JDD062                / VOL=(PRIVATE,
         RETAIN                             / VOL=(PRIVATE,RETAIN
         ','        ; JDD062                / VOL=(PRIVATE,RETAIN,
         DIG                                / VOL=(PRIVATE,RETAIN,N
         ','        ; JDD062                / VOL=(PRIVATE,RETAIN,N,
         DIG                                / VOL=(PRIVATE,RETAIN,N,N
         ','                                / VOL=(PRIVATE,RETAIN,N,N,
JDD062   #JSER01                            / VOL=(,,,,REF=*.STEP.PROC.DDNAME
         ')' -> JDD031 ; #145               / VOL=(,,,,REF=*.STEP.PROC.DDNAME)
/              S E R
JSER01   SER        ; JSER08                / SER
         '='        ; #146                  / SER=
         NAME(JSERA03) -> RETURN            / SER=NAME
         '('        ; #147                  / SER=(
JSER05   NAME(JSERA03) ; #150               / SER=(NAME
         ',' -> JSER05                      / SER=(NAME,
         ')' -> RETURN ; #151               / SER=(NAME,...)
JSER08   REF        ; #RETURN               / REF
         '='(JA31(JSCAN40)) ; #152          / REF=
         #JF01 -> RETURN                    / REF=*.STEP.PROC.DDNAME
         NAME(JA31(JSCAN20)) ->RETURN ;#153
/              L A B E L
JDD064   '='    ; #114                      / LABEL=
         DIG        -> JDD031               / LABEL=N
         #JEXP01    -> JDD031               / LABEL=EXPDT=YYNNN
         '('        ; #154                  / LABEL=(
         DIG                                / LABEL=(N
```

```
          ',' ; JDD076                  / LABEL=(N,
          LABTYP                        / LABEL=(N,SL
          ',' ; JDD076                  / LABEL=(N,SL,
          PASSWORD                      / LABEL=(N,SL,PASSWORD
          ',' ; JDD076                  / LABEL=(N,SL,PASSWORD,
          INOUT                         / LABEL=(N,SL,PASSWORD,INOUT
          ','                           / LABEL=(N,SL,PASSWORD,INOUT,
JDD076    #JEXP01                       / LABEL=(,,,,RETPD=NNNN
          ')' -> JDD031  ; #155         / LABEL=(,,,,RETPD=NNNN)
/                E X P D T      R E T P D
JEXP01    EXPDT     ; JEXP04            / EXPDT
          '='       ; #156              / EXPDT=
          DIG(JEXPA3) -> RETURN ; #157  / EXPDT=YYDDD
JEXP04    RETPD     ; #RETURN           / RETPD
          '='       ; #160              / RETPD=
          DIG(JEXPA6) ->RETURN ; #161   / RETPD=NNNN
/                D I S P
JDD078    '='       ; #162              / DISP=
          SNOM -> JDD031                / DISP=SHR NEW OLD MOD
          '('       ; #163              / DISP=(
          SNOM                          / DISP=(SHR NEW OLD MOD
          ','       ; JDD087            / DISP=(SNOM,
          PASS -> JDD085                / DISP=(SNOM,PASS
          DKCU                          / DISP=(,DELETE KEEP CATLG UNCATLG
JDD085    ','       ; JDD087            / DISP=(,,
          DKCU                          / DISP=(,,DELETE KEEP CATLG UNCATLG
JDD087    ')' -> JDD031 ; #164          / DISP=(,,)
/                S P A C E
JDD088    '='       ; #165              / SPACE=
          '('       ; #166              / SPACE=(
          TRK  -> JDD093                / SPACE=(TRK
          CYL  -> JDD093                / SPACE=(CYL
          DIG       ; #167              / SPACE=(NNNN
JDD093    ','       ; #170              / SPACE=(NNNN,
          '('       ; JDD101            / SPACE=(NNNN,(
          DIG       ; #172              / SPACE=(NNNN,(NN
          ','       ; JDD100            / SPACE=(NNNN,(NN,
          DIG                           / SPACE=(NNNN,(NN,NN
          ','       ; JDD100            / SPACE=(NNNN,(NN,NN,
          DIG       ; #173              / SPACE=(NNNN,(NN,NN,NN
JDD100    ')'  ->JDD102  ; #174         / SPACE=(NNNN,(NN,NN,NN)
JDD101    DIG                           / SPACE=(NNNN,NN
JDD102    ','       ; JDD109            / SPACE=(N,(N,N,N),
          RLSE                          / SPACE=(N,(N),RLSE
          ','       ; JDD109            / SPACE=(N,(N),RLSE,
          ACM     -> JDD107             / SPACE=(,,,ACM
JDD107    ','       ; JDD109            / SPACE=(N,(N),,,
          ROUND     ; #175              / SPACE=(N,(N),,,ROUND
JDD109    ')' -> JDD031  ; #176         / SPACE=(N,(N),,,ROUND)
/                S Y S O U T
JDD110    '='(JA31(JSCAN10))  ; #171    / SYSOUT=
          (JA62(JSCAN20)) ->JDD031 ; #177 / SYSOUT=A-Z\0-9
/                S E P
JDD112    #JSEP01 -> JDD031 ; #204      / SEP=(DDNAME,...)
JSEP01    '='       ; #RETURN           / SEP=
          '('       ; JSEP06            / SEP=(
JSEP03    NAME      ; #205              / SEP=(DDNAME
          ','  -> JSEP03               / SEP=(DDNAME,
          ')'  -> RETURN  ; #206        / SEP=(DDNAME,...)
JSEP06    NAME -> RETURN  ; #207        / SEP=DDNAME
```

```
          '/'  -> RETURN    ; #231          / //
/                 E X E C    S T A T E M E N T
JG75      #JBLNK    ;#242                   / //NAME EXEC
          PGM    ; JG80                     / PGM
          '='    ; #66                      / PGM=
          #JF01 -> JG83                     / PGM=*.STEP.PROC.DDNAME
          NAME  -> JG83   ; #67             / PGM=NAME
JG80      PROC   ;  JG82                    / PROC
          '='    ;  #70                     / PROC=
JG82      NAME(JA76(1)) ; #71               / PROC=NAME
JG83      ',' -> JG85                       / ,
          ' ' -> JG03   ; JG17              /
JG85      DPRTY(JA36)  -> JG94              / DPRTY
          PARM(JA36)   -> JG98              / PARM
          COND(JA36)   -> JH13              / COND
          ROLL(JA36)   -> JH07              / ROLL
          REGION(JA36) -> JH09              / REGION
          TIME(JA36)   -> JH11              / TIME
          ACCT(JA45)                        / ACCT
          RD(JA45)                          / RD
          RESTART(JA45)                     / RESTART
          #JSYM01 -> JG83                   / SYMBOLIC PARAMETER.
          #JCT01  -> JG85  ; #72            / CONTINUATION.
/                 S Y M B O L I C  P A R A M E T E R
JSYM01    NAME   ; #RETURN                  / NAME
          '='(JA31(JSCAN30))   ; #240       / NAME=
          NAME(JA31(JSCAN20)) -> RETURN     / NAME=NAME
          #JQ01(JA31(JSCAN20)) ->RETURN ; RETURN  / NAME='FIELD'
/                 D P R T Y
JG94      #JP01                             / DPRTY.PROC
          '='    ; #73                      / DPRTY=
          DIG(JA49(-15)) -> JG83 ; #74      / DPRTY=0-15
/                 P A R M
JG98      #JP01                             / PARM.PROC
          '='(JA31(JSCAN30))   ;  #75       / PARM=
          NAME(JA31(JSCAN20))   -> JG83     / PARM=ALPHAMERIC
          '('    ; JH06                     / PARM=(
JH02      NAME   -> JH04                    / PARM=(ALPHAMERIC
          #JQ01                             / PARM=('STRING'
JH04      ','    -> JH02                    / PARM=(PARMFIELD,
          ')'(JA31(JSCAN20))->JG83;#76      / PARM=(FIELD,...)
JH06      #JQ01(JA31(JSCAN20)) ->JG83 ;#77  / PARM='STRING'
/                 R O L L
JH07      #JP01                             / ROLL.PROC
          #JO01 -> JG83   ; #100            / ROLL=(YES\NO,YES\NO)
/                 R E G I O N
JH09      #JP01                             / REGION.PROC
          #JR01 -> JG83   ; #101            / REGION=NNNNNK
/                 T I M E
JH11      #JP01                             / TIME.PROC
          #JT01 -> JG83   ; #102            / TIME=(MIN,SEC)
/                 C O N D
JH13      #JP01                             / COND.PROC
          '='    ; #103                     / COND=
          '('    ; JH21                     / COND=(
          #JE01  ; JH20                     / COND=(EVEN\ONLY\(CODE,OPER,STEP)
JH17      ','    ; JH19                     /
          #JD01 -> JH17   ; #105            /
JH19      ')'    -> JG83   ; #106           /
JH20      #JE04 -> JG83   ; #107            /
```

```
JH21     EO  ->JG83     ; #104          / COND = EVEN \ ONLY
JH22     #JBLNK1    ; #243              / PROC
         EOS(JA03) -> JCLGRAPH          / //    PROC
JH23     #JSYM01   ; JH26               / SYMBOLIC PARM.
         ','  -> JH23                   / SYM PARM.
         ' '  -> JG03 ; JG17            / BLANK.
JH26     #JCT01 ->JH23 ; #244           / CONTINUATION
/              C O N D    S U B G R A P H
JD01     '('    ; #RETURN
         #JC01 ; #110
JD03     ','    ; JD07
         NAME   ; #111
         '.'    ; JD07
         NAME   ; #112
JD07     ')'    -> RETURN ; #113
/              C O N D    S U B G R A P H
JE01     #JD01 -> RETURN
         EO  -> RETURN ; #RETURN        / EVEN\ONLY
JE04     #JC01 -> JD03  ; #RETURN
/              . P R O C S T E P N A M E
JP01     '.'(JPA1) ; #RETURN            / .
         NAME -> RETURN ; #115          / .PROCSTEPNAME
/              *.STEPNAME.PROCSTEPNAME.DDNAME
JF01     '*'(JA31(JSCAN20))  ; #RETURN / *
         '.'    ; #116                  / *.
         NAME   ; #117                  / *.NAME
         '.'    ; RETURN                / *.NAME.
         NAME   ; #120                  / *.NAME.NAME
         '.'    ; RETURN                / *.NAME.NAME.
         NAME   -> RETURN ; #121        / *.NAME.NAME.NAME
/              ' S T R I N G '
JQ01     ''''(JA31(JSCAN10)) ; #RETURN  / '
JQ02     ''''   ; JQ04                  / ''
         '''' -> JQ02  ; RETURN         / '''
JQ04     ->JQ02                         / 'X
/              DD  STATEMENT.
JDD001   #JBLNK1    ; #122              / //<NAME> DD
         '*'        -> JDD007           / //<NAME> DD *
         DATA       -> JDD007           / //<NAME> DD DATA
         DDNAME     ; JDD017            / //<NAME> DD DDNAME
         '='        ; #123              / //<NAME> DD DDNAME=
         NAME       ; #124              / //<NAME> DD DDNAME=NAME
JDD007   ','        ; JDD016            / //<NAME> DD *,
JDD008   DCB        ; JDD015            / *,DCB
         '='        ; #126              / *,DCB=
         '('        ; JDD014            / *,DCB=(
JDD011   #JDCB01    ; #127              / *,DCB=(BLKSIZE=NNNN
         ','        -> JDD011           / *,DCB=(BLKSIZE=NNNN,
         ')'->JDD016; #130              / *,DCB=(BLKSIZE=NNNN,BUFNO=N)
JDD014   #JDCB01    -> JDD016 ; #131    / *,DCB=BLKSIZE=NNNN
JDD015   #JCT01     -> JDD008 ; #125    /
JDD016   ' '        -> JG03   ; JG17    /
JDD017   DUMMY      -> JDD031           / //<NAME> DD DUMMY
         EOS(JA03) -> JCLGRAPH          /  //     D
JDD018   DSNAME(JA20)   -> JDD033       / DSNAME
         UNIT(JA20) -> JDD115           / UNIT
         UCS(JA20) -> JDD041            / UCS
         VOLUME(JA20)  -> JDD050        / VOLUME
         DCB(JA20) -> JDD131            / DCB
         LABEL(JA20) -> JDD064          / LABEL
```

```
/                A F F
JDD113    '='        ; #210                / AFF=
          NAME ->JDD031    ; #211          / AFF=DDNAME
/                U N I T
JDD115    '='        ; #212                / UNIT=
          '('        ;  JDD128             / UNIT=(
          #JUT01                           / UNIT=(TAPE DISK 280 ETC.
          ','        ; JDD124              / UNIT=(,
          P    -> JDD121                   / UNIT=(,P
          DIG(JDA120)                      / UNIT=(,DIG
JDD121    ','        ; JDD124              / UNIT=(,,
          DEFER                            / UNIT=(,,DEFER
          ','        ; JDD124              / UNIT=(,,,
JDD124    SEP        ; JDD127              / UNIT=(,,,SEP
          #JSEP01    ; #214                / UNIT=(,,,SEP=(DDNAME,...)
JDD127    ')' ->JDD031 ; #215              / UNIT=(,,,)
JDD128    #JUT01 ->JDD031                  / UNIT=00E 009 TAPE 2400
          SEP -> JDD112                    / UNIT=SEP=(DDNAME,...)
          AFF  -> JDD113   ; #216          / UNIT=AFF=DDNAME
JUT01     UNITYPE  -> RETURN               / TAPE DISK SYSDA ETC.
          DIG(JUT02A) -> RETURN ; #RETURN/ 2400 137 00E ETC.
/                D C B
JDD131    '='        ; #217                / DCB=
          #JDCB01 -> JDD031                / DCB=SUBPARM
          '('     ; JDD137                 / DCB=(
JDD134    #JDCB01                          / DCB=(SUBPARM
          ','   -> JDD136                  / DCB=(SUBPARM,
          ')'   -> JDD031   ; #222         / DCB=(SUBPARM,...)
JDD136    #JCT01 ->JDD134 ; JDD134         / CONTINUATION.
JDD137    #JF01 ->JDD031 ; #220            / DCB=*.NAME.NAME,NAME
JDD138    '='    ; #245                    / OUTLIM=
          DIG    -> JDD031   ; #246        / OUTLIM=NNNN
JDCB01    LRECL       ; JDCB04             / LRECL
JDCB02    '='         ; #223               / LRECL=
          DIG  -> RETURN   ; #224          / LRECL=NNNN
JDCB04    DSORG       ; JDCB07             / DSORG
          '='         ; #225               / DSORG=
          DORG  -> RETURN   ; #226         / DSORG=DORG
JDCB07    RECFM       ; JDCB10             / RECFM
          '='         ; #227               / RECFM=
          NAME    ->RETURN ; #230          / RECFM=NAME
JDCB10    BLKSIZE -> JDCB02                / BLKSIZE
          BUFL   ->JDCB02                  / BUFL
          BUFNO    -> JDCB02   ; #RETURN   / BUFNO
JBLNK1    ' '    ; #RETURN                 /
JBLNK2    ' '    -> JBLNK2    ;  RETURN    /
/                C O N T R O L
JCR01     #JCR02(JCR01A(JCR16))  ->JCLGRAPH ; JG05  / CONTROL
JCR02     '$'(JA31(JSCAN20)) ; JCR19       / $
          CALL   -> JCR10                  / $CALL
          LIST1(JCR04A(JPRT20)) ->JCR08    / $LIST1
          LIST2(JCR04A(JPRT30)) ->JCR08    / $ LIST2
          JCL(JCR06A(JPRT40)) ->JCR08      / $JCL
          (JCR07A(JSCAN10)) ; #RETURN      / $..
/                $ J C L
JCR08     EOS(JA31(JSCAN10)) -> RETURN     / $JCL
          ->JCR08                          / $JCL
/                $ C A L L
JCR10     ' '(JA31(JSCAN10)) ; #RETURN     / $CALL
          '@'(JCR11A(JCR13A)) -> JCR13     / $CALL @

          (JCR11A(JCR14A)) ->JCR14         / $ CALL  .
JCR13     (JCR12A) -> JCR13  ; JCR15       / $CALL @
JCR14     (JCR12A)  -> JCR14               / $CALL ........
JCR15     -> RETURN                        / $CALL @.....................
/                $ L I S T
JCR16     #JCR02(JCR01A(JCR16)) -> JCLGRAPH
JCR17     EOS   -> JCR16                   /
          ->JCR17                          /
JCR19     EOF(JCR19A) ->RETURN ; #RETURN/ EOF
```