Structured Message Parsing

by Ian S. Worthington

Introduction

Developing a high quality user interface for new messages has always been a problem. The provision of suitable parameters (and synonyms), and handling the parsing of combinations of positional, keyword, and complex structured parameters has traditionally been very much an afterthought – and a difficult, time consuming one at that – and one of the reasons for the plethora of difficult and diverse messages formats which continue to populate ALCS and TPF systems worldwide

Fortunately ALCS provides some nice facilities to help us easily implement high-quality user interfaces, though, as is somewhat regrettably par for the course, it keeps its glory well hidden: a check through the manuals will reveal that there do appear to be some built-in facilities for message parsing, but the documentation offered doesn't shed much light on how to use them.

Phased approach

The ALCS approach is to separate the message parsing into distinct phases:

1. **Lexical analysis**: convert the message block into a structured format, without regard for its contents, checking only the validity of its textual structures (such as quoted strings and parenthetical elements)
2. **Syntactical analysis:** examine the structured format to identify, in turn, each of the parameters contained in the input message
3. **Semantic analysis:** validate the correctness of the parameters in context, examine their interdependencies, check for exclusiveness and contradictions.

ALCS provides services to help with both the lexical and syntactical analysis of input messages. The semantic analysis is very application dependent, differs greatly between programs, and is left entirely to the application program itself.

Lexical analysis

Input messages are very easily converted. The most straightforward way to achieve this is to allow ALCS to do it for you, by specifying PARSE=YES on the FMSGC macro used to build your functional message entry in ACM0 (or your input editor). This macro also has parameters to specify:

- translation of dashes to equals
- translation of slashes to parenthesis
- folding to uppercase
- valid system states
- valid terminal priorities

For example:

```
FMSGC xxxx,                 define the action code
   PROG=xxxx,               define the service program
   PARSE=YES,               parse the message
   TDASH=YES,               translate dashes to equals
   TBRKT=YES,               translate slashes to parenthesis
   MINLEN=5,                minimum valid command length
   TERM=CRAS,               valid terminals
   SYST=(MESW,NORM)         valid system states
```

However if you don't want to use FMSGC, are testing your program using ZDRIV, or wishing to further parse complex substrings of the input message it is useful to be able to call the parser package itself. This is easily done by an ENTRC to CFMP (or CFMQ if you wish to parse the message block starting from byte 1 rather than byte 6).

The package returns the converted message on level 2 and also sets various error indicators, which would normally be tested for you by the input editor and appropriate error messages issued to the user. If though you are calling the parser package directly it is useful to wrap it in another program which performs the same function. In ALCS this function is performed in program CFMS and an example of the required code is shown in Appendix A.

Syntactical Analysis

So once we've got our parsed message on level 2, what exactly have we got, and how do we go about processing it? Lets look at an example.

Let's use the input message:

```
zdriv xxxx quick age=90 complex=(this,that,'and the other')
```

This message consists of:

- a positional parameter ('quick'),
- a keyword parameter ('age=90'),
- a complex keyword parameter consisting of a parenthetical value clause containing three positional parameters, the third being a quoted string ('(this, that,'and the other')')

The parser has transformed the input message from:

```
D0  0000  D4C90000 00000000 00000000 00000000  MI..............
D0  0010  00340200 02D8E4C9 C3D240C1 C7C57EF9  .....QUICK AGE.9
D0  0020  F040C3D6 D4D7D3C5 E77E4DE3 C8C9E26B  0 COMPLEX.(THIS.
D0  0030  E3C8C1E3 6B7DC1D5 C440E3C8 C540D6E3  THAT..AND THE OT
D0  0040  C8C5D97D 5D4E0040 40000000 00000000  HER.)..  .......
```

to:

```
D2  0000  0000003F 00030000 00000000 00000005  ................
D2  0010  D8E4C9C3 D2000000 00000000 00000000  QUICK...........
D2  0020* 00000000 00000000 00000000 00000000  ................
D2  0040  00000000 000003C1 C7C50000 00000002  .......AGE......
D2  0050  F9F00000 00000000 00000000 00000000  90..............
D2  0060* 00000000 00000000 00000000 00000000  ................
D2  0080  00000000 000007C3 D6D4D7D3 C5E7001B  .......COMPLEX..
D2  0090  4DE3C8C9 E26BE3C8 C1E36B7D C1D5C440  (THIS.THAT..AND
D2  00A0  E3C8C540 D6E3C8C5 D97D5D00 00000000  THE OTHER.).....
D2  00B0* 00000000 00000000 00000000 00000000  ................
```

This format is described by dsect macro CO0PR. It starts with a fixed section:

Symbol	Offset	Type/Length	Description	Value in this example
PRSW1	0	BL2	Error bits	x'0000'
PRNMX	2	H	Available entries	x'003F'
PRNUM	4	H	Used entries	x'0003'

This is followed by a variable number (PRNUM) of 64 byte fields:

Entry	**PRENT** (One entry) CL64			
	PRKEY (Keyword field) CL9		**PRPRM** (Parameter field) CL53	
	PRKLN AL1 (length)	**PRKWD** CL8 (keyword)	**PRVLN** AL1 (length)	**PRVAL** CL52 (parameter)
1	0		5	QUICK
2	3	AGE	2	90
3	7	COMPLEX	27	THIS,THAT,AND THE OTHER

To assist in the analysis of this ALCS provides:

- A macro to define the expected parameters and synonyms
- A macro to process the input data fields in conjunction with this defined data

Processing the expected parameters

ALCS macro CM0ND builds a data structure containing valid parameters and their synonyms. For ease of processing it is recommended to specify positional and keyword parameters in different tables. Thus for our example, if we wished to allow:

Positional parameters:

- QUICK (with synonyms QUIC, QUI, QU, Q)
- TEMP (with synonyms TEM, TMP, TE, T)

Keyword parameters:

- AGE (with synonyms AG, A)
- COMPLEX (with synonyms COMPLE, COMPL, COMP, COM, CO, C)

we would code two macros, such as these:

```
PPL01  CM0ND ((QUICK,Q,QU,QUI,QUIC),
              (TEMP,TEM,TE,T))

KPL01  CM0ND ((AGE,AG,A),
              (COMPLEX,COMPLE,COMP,COM,CO,C))
```

Each of these parameters requires a subroutine to process it. The addresses of these are specified, in the same order as the parameters for CM0ND, in an associated branch table. ALCS allows this to be placed inline but, for ease of maintenance, it is recommended to locate it immediately adjacent to the parameter definition macros. The first entry in the branch table is reserved for a routine to process invalid parameters. For example:

```
PPE01   B      PPERROR                Handle invalid positional parameter
        B      PQUICK                 Handle 'QUICK'
        B      PTEMP                  Handle 'TEMP'

KPE01   B      KERROR                 Handle invalid keyword parameter
        B      KAGE                   Handle 'AGE'
        B      KCOMPLEX               Handle 'COMPLEX'
```

Finally we come to the processing of the input parameters. Having defined all our data blocks, the processing couldn't be simpler. CM0PR is invoked for each parameter, specifying the location of:

- The address of the valid parameter table
- The address of the branch table
- The address of the data element to process

For example:

```
         L     R03,CE1CR2                LOAD ADDRESS OF PARSED MESSAGE BLOCK
        CO0PR REG=R03                    USE R14 AS PARSED MESSAGE BASE

        #DO   TIMES=(R01,H/PRNUM),       LOOP FOR EACH PARAMETER
        #     WHILE=(R01,NZ)             IF THERE ARE ANY PARAMETERS
              LA    R07,RETADR           DEFINE A RETURN ADDRESS
              #IF   PRKLN,NONZERO        PROCESS KEYWORD?
                    CM0PR LIST=KPL01,EXIT=KPE01,DATA=PRKWD
              #ELSE                      OR PROCESS POSITIONAL PARAMATER
                    CM0PR LIST=PPL01,EXIT=PPE01,DATA=PRVAL
              #EIF
RETADR        EQU   *                    RETURN FROM PROCESSING ROUTINE
              LA    R03,L'PRENT(,R03)    SKIP TO NEXT ENTRY
        #EDO
        RELCC LEVEL=D2                   RELEASE PARSED BLOCK
        DROP  R03
```

The actions of each individual parameter-processing subroutine, and the subsequent validations required, are, of course, highly specific to the application in question, and not discussed in detail here, other than to say that it is usually more convenient to separate the syntactic and semantic analysis. In practice this results in simpler parameter processing subroutines, simply setting bits and storing values (possibly after conversion), then checking the validity of everything later. This simplifies matters, avoids having to duplicate cross-validation and, perhaps more important for your users, makes it easy to issue messages for all error conditions at one time, avoiding the frustration we've all experienced of finally figuring out the correct format for one parameter, only then to be told we can't use it in the context we're trying to use the entry!

It is also worth commenting that CM0PR may also be invoked again within any keyword processing subroutine in order to identify any fixed keyword values.

A full example is shown in Appendix B.

Summing up

ALCS supplies all the tools required to implement a high-quality input message parser. The required steps are:

1. Parse the message into CO0PR format using an FMSGC definition in your existing input editor, or by indirect entry to CFMP/CFMQ.
2. Define command synonym tables using CM0ND.
3. Define a table containing branches to each parameter processing subroutine. Don't forget to specify an error routine.
4. Use macro CM0PR within a processing loop to identify and process each parameter or its synonym.
5. Keep each parameter processing subroutine as simple as possible. Delay cross-validation of parameters.
6. When completing the semantic analysis and cross-validation, don't just issue one error message and exit. It may help your users if you issue all the error messages you can.

Thanks to…

My grateful thanks to Tony Shortman of IBM's ALCS Development and Support Group, London, for his review and comments on the draft of this article. Any errors that remain are of course mine alone. He has asked me to point out that the interfaces described here are, officially speaking, part of the unpublished ALCS interfaces and, whilst they remain so, IBM is at liberty to modify and/or enhance them in such a way that the changes may impact code any you develop based on them. My thanks also to

Aer Lingus in general, and Paul Keogh in particular, whose requirement was the original inspiration for this article.

Ian S. Worthington is an independent ALCS and MVS systems consultant, currently looking for project work. If you would like to offer him any, or contact him for any other reason, you may do so via email at IanWorthington@usa.net, or via fax through his virtual offices in New York (+1.212.208.4463) and London (+44.20.7900.2916).

Copies of this article and the code appendices may be obtained from the System Technology International Website at http://247sti.tripod.com/documents

---

Appendix A: Generic Message Parser

```
         BEGIN NAME=XPQ1,VERSION=00,AMODE=31,TYPE=MONITOR,              -
               SHR=NONE,XCL=NONE
*
*=====================================================================*
*         GENERIC MESSAGE PARSER                                       *
*=====================================================================*
*
*_____________________________________*
*                NEW SEGMENT - EXTRACTED FROM CFMS                     *
*_____________________________________*
*
*=====================================================================*
*         XPQ1 ENTRY POINT                                             *
*=====================================================================*
*
XPQ1     EQU   *
         ENTRC CFMQ                PARSE MESSAGE ONTO LEVEL 2
*
         L     R01,CE1CR0          RESTORE MESSAGE BLOCK BASE ADDRESS
         L     R14,CE1CR2          LOAD ADDRESS OF PARSED MESSAGE BLOCK
         CO0PR REG=R14             USE R14 AS PARSED MESSAGE BASE
*
         CLC   PRSW1,=XL(L'PRSW1)'00'  HAVE ANY ERRORS OCCURRED
         BE    XPQ1F10             NO - BRANCH
*
         LA    R03,10              LOAD ERROR MESSAGE NUMBER
         TM    PR1MAX,L'PR1MAX     TO MANY INPUT PARAMETERS ?
         BO    XPQ1ERR             YES - ERROR
*
         LA    R03,12              LOAD ERROR MESSAGE NUMBER
         TM    PR1UMP,L'PR1UMP     UNMATCHED PARENTHESES
         BO    XPQ1ERR             YES - ERROR
*
         LA    R03,14              LOAD ERROR MESSAGE NUMBER
         TM    PR1LNG,L'PR1LNG     PARAMETER TO LONG ?
         BO    XPQ1ERR             YES - ERROR
*
```

```
         LA    R03,16              LOAD ERROR MESSAGE NUMBER
         TM    PR1MSK,L'PR1MSK     MISSING KEYWORD ?
         BO    XPQ1ERR             YES - ERROR
*
         LA    R03,18              LOAD ERROR MESSAGE NUMBER
         TM    PR1KLN,L'PR1KLN     KEYWORD TO LONG ?
         BO    XPQ1ERR             YES - ERROR
*
         LA    R03,20              LOAD ERROR MESSAGE NUMBER
         TM    PR1QOT,L'PR1QOT     UNMATCHED QUOTES ?
         BO    XPQ1ERR             YES - ERROR
*
         LA    R03,22              LOAD ERROR MESSAGE NUMBER
         TM    PR1KBD,L'PR1KBD     INVALID KEYWORD ?
         BO    XPQ1ERR             YES - ERROR
*
         SR    R03,R03             SET INVALID COMMAND
         B     XPQ1ERR             GO OUTPUT ERROR MESSAGE
*
         DROP  R14                 DROP CO0PR BASE

*-----------------------------------*
*        NO ERRORS, RETURN                                          *
*-----------------------------------*
*
XPQ1F10  EQU   *
         BACKC ,
*
*-----------------------------------*
*        MESSAGE IN ERROR - SEND RESPONSE AND EXIT                  *
*-----------------------------------*
*
XPQ1ERR  EQU   *
         LH    R04,XPQ1INDX(R03)   LOAD ERROR MESSAGE NUMBER
         SPACE 1
         WTOPC NUM=(R04),          ISSUE ERROR MESSAGE               -
               PREFIX=CMD,                                           -
               TIME=YES,                                             -
               SUB=(CHARA,EBW000)  SECONDARY ACTION CODE
*
         EXITC ,
*
*====================================================================*
*        PROGRAM CONSTANTS - DIRECTLY ADDRESSED                     *
*====================================================================*
*        TABLE CONTAINING ERROR MESSAGE EQUATES
*-----------------------------------*
         SPACE 1
XPQ1INDX DC    0F'0'
         DC    Y(#DXCCMD001)       INVALID COMMAND FORMAT
         DC    Y(#DXCCMD002)       UNKNOWN COMMAND
```

```
         DC    Y(#DXCCMD003)       WRONG SYSTEM STATE
         DC    Y(#DXCCMD004)       PRIME CRAS ONLY
         DC    Y(#DXCCMD005)       CRAS ONLY
         DC    Y(#DXCCMD031)       TOO MANY PARAMETERS
         DC    Y(#DXCCMD007)       UNMATCHED PARENTHESES
         DC    Y(#DXCCMD008)       PARAMETER TOO LONG
         DC    Y(#DXCCMD009)       MISSING KEYWORD
         DC    Y(#DXCCMD010)       KEYWORD TOO LONG
         DC    Y(#DXCCMD011)       UNMATCHED QUOTE
         DC    Y(#DXCCMD012)       INVALID KEYWORD
         DC    Y(#DXCCMD019)       PRIME CRAS OR AT1 - 16
*
         FINIS ,
         END   ,
```

Appendix B: Full Example

```
         BEGIN NAME=XPQ0,VERSION=00,SHR=NONE,XCL=NONE,LPW=YES
         TRANV NAME=XPQP,ENTRY=XPQP
*
* ****** DEFAULTS
*
MAXDAYS  EQU  90           REQUEUE TO DAAD IF OVER THIS AGE
DEFTOQ   DC    CL5'DDDD '  DEFAULT 'TO' QUEUE
DEFTMPQ  DC    CL5'MCTC '  DEFAULT 'TEMP' QUEUE
*
* ****** DESCRIBE WORK AREAS
*
EB0EB    DSECT
         ORG   EBL000
ALTPRT   DS    XL3         ALTERNATE PRINTER CRI
WORK     DS    XL8         PACK WORK AREA
*
QUICK    DS    0XL(B'10000000')
ERROR    DS    0XL(B'01000000')
AGE      DS    0XL(B'00100000')
SINGLE   DS    0XL(B'00010000')
CONFREQD DS    0XL(B'00001000')
         DS    XL1
*
LEBL     EQU   *-EBL000,L'CE1WKA-(*-EBL000)     ERROR IF TOO BIG
*
*
*
         ORG   EBX000
UCDVWA   DS    CL18        WORK AREA FOR UCDV
LEBX     EQU   *-EBX000,L'CE1WKA-(*-EBX000)     ERROR IF TOO BIG
*
         RSECT ,
         .
         .
```

```
XPQP     EQU   *
*
* ****** SET DEFAULTS
*
         NI    ERROR,X'FF'-L'ERROR  NO ERRORS YET
         NI    CONFREQD,X'FF'-L'CONFREQD  CONFIRMATION NOT YET REQUIRED
         NI    QUICK,X'FF'-L'QUICK  NORMAL PROCESING
         NI    SINGLE,X'FF'-L'SINGLE      PROCESS ALL RSDS
         MVC   MSGAGE,=AL2(MAXDAYS) DEFAULT MESSAGE AGE
         MVC   FROMRSD,=CL5' '
         MVC   TORSD,DEFTOQDEFAULT 'TO' QUEUE
         MVC   TEMPRSD,DEFTMPQ      DEFAULT 'TEMP' QUEUE
*
* ****** ALTERNATE ENTRY FROM TIME INITIATED TABLE?
*
         #IF   EBW000,EQ,X'63',AND,
         #     CLC,EBW002(4),EQ,=C'XPQP'
*
               #IF   CLC,EBW006(2),EQ,=C'/Q'
                     OI    QUICK,L'QUICK  PROCESS IN QUICK MODE
               #EIF
*
               B     XPQ0233
         #EIF
*
* ****** PROCESS INPUT MESSAGE
*
         ENTRC XPQ1        PARSE MESSAGE ONTO LEVEL 2
*
         L     R03,CE1CR2  LOAD ADDRESS OF PARSED MESSAGE BLOCK
         PUSH PRINT
         PRINT GEN
         CO0PR REG=R03     USE R14 AS PARSED MESSAGE BASE
         POP PRINT
*
         #DO   TIMES=(R01,H/PRNUM)
               LA    R07,XPQ0231
*
               #IF   PRKLN,NONZERO
                     CM0PR LIST=KWPL01,EXIT=KWPE01,DATA=PRKWD
               #ELSE
                     CM0PR LIST=PPL01,EXIT=PPE01,DATA=PRVAL
               #EIF
*
XPQ0231        EQU   *
               LA    R03,L'PRENT(,R03)
         #EDO
*
         RELCC LEVEL=D2    RELEASE PARSED BLOCK
         B     XPQ0233
*
*
```

```
KWPL01   CM0ND ((FROM,FRO,FR,F),                                        -
               (TO),                                                    -
               (SORT,SOR,SO,S,SORTBY,BY),                               -
               (TEMP,TEM,TE,TMP),                                       -
               (AGE,AG,A))
KWPE01   B     KWPERR      INVALID KEYWORD PARAMETER
         B     PFROM
         B     PTO
         B     PSORT
         B     PTEMP
         B     PAGE
*
PPL01    CM0ND ((QUICK,QUIC,QUI,QU,Q))
PPE01    B     PPERR       INVALID POSITIONAL PARAMETER
         B     PQUICK
*
*
KWVL01         CM0ND ((RSD,RS,R),                                        -
               (LASTSEND,LAST,SEND,LS,L,S),                             -
               (CUMM,CUM,CU,C))
KWVE01         B     KWVERR          INVALID KEYWORD VALUE
               B     PRSD
               B     PLS
               B     PCUM
*
*
KWPERR   EQU   *
         WTOPC NUM=188,LET=E,                                           -
               TEXT='Unknown keyword parameter ........',               -
               SUB=(CHARA,PRKWD),COMP=YES
         OI    ERROR,L'ERROR
         BR    R07
*
*
PPERR    EQU   *
         WTOPC NUM=195,LET=E,                                           -
              TEXT='Unknown positional parameter ..................', -
               SUB=(CHARA,PRKWD),COMP=YES
         OI    ERROR,L'ERROR
         BR    R07
*
*
PQUICK   EQU   *
         OI    QUICK,L'QUICK
         BR    R07
*
*
PFROM    EQU   *
         MVC   FROMRSD,=CL5' '      CLEAR SLOT
         XR    R02,R02     CLEAR REGISTER FOR INSERT
         IC    R02,PRVLN   GET LENGTH
```

```
         BCTR  R02,0       PREPARE FOR EXECUTE
         MVC   FROMRSD(1),PRVAL     TARGET MVC
         EX    R02,*-6     EXECUTE MVC
         OI    SINGLE,L'SINGLE      PROCESS JUST SPECIFIED RSD
*
         BR    R07
*
*
PTO      EQU   *
         MVC   TORSD,=CL5' '  CLEAR SLOT
         XR    R02,R02     CLEAR REGISTER FOR INSERT
         IC    R02,PRVLN   GET LENGTH
         BCTR  R02,0       PREPARE FOR EXECUTE
         MVC   TORSD(1),PRVAL TARGET MVC
         EX    R02,*-6     EXECUTE MVC
*
         BR    R07
*
*
PSORT    DC    0H'0'
         SAVEC PUSH=GPRS                  SAVE REGS
               LA    R06,XPQ0352
               CM0PR LIST=KWVL01,EXIT=KWVE01,DATA=PRVAL
*
XPQ0352        DC    0H'0'
         SAVEC POP                        RESTORE REGS
         BR    R06
*
*
PRSD     DC    0H'0'
         MVC   DSORTCOL,=AL2(RSD-HDR)
         MVC   DSORTLEN,=AL2(L'RSD)
         BR    R06
*
*
PLS      DC    0H'0'
         MVC   DSORTCOL,=AL2(LSEND-HDR+L'LSEND-4)  sort on year
         MVC   DSORTLEN,=AL2(4)
         BR    R06
*
*
PCUM     DC    0H'0'
         MVC   DSORTCOL,=AL2(CUM-HDR)
         MVC   DSORTLEN,=AL2(L'CUM)
         BR    R06
*
*
PTEMP    EQU   *
         MVC   TEMPRSD,=CL5' '      CLEAR SLOT
         XR    R02,R02     CLEAR REGISTER FOR INSERT
         IC    R02,PRVLN   GET LENGTH
```

```
         BCTR  R02,0       PREPARE FOR EXECUTE
         MVC   TEMPRSD(1),PRVAL     TARGET MVC
         EX    R02,*-6     EXECUTE MVC
*
         BR    R07
*
*
PAGE     EQU   *
         SAVEC PUSH=GPRS
         #IF   PRVLN,EQ,X'0'
               WTOPC NUM=141,LET=E,                                    -
               TEXT='AGE parameter not specified'
               OI    ERROR,L'ERROR
*
               B     PAGERET
         #EIF
*
         XR    R02,R02     CLEAR REGISTER FOR INSERT
         IC    R02,PRVLN   GET LENGTH
         BCTR  R02,0       PREPARE FOR EXECUTE
         TRT   PRVAL(0),TRT1  (EXECUTED INSTRUCTION)
         EX    R02,*-6     TEST FOR NUMERIC
*
         #IF   NZ
               WTOPC NUM=144,LET=E,                                    -
               TEXT='AGE parameter not numeric'
               OI    ERROR,L'ERROR
*
               B     PAGERET
         #EIF
*
         EX    R02,EXPACK  EXECUTE PACK
         CVB   R02,WORK    CONVERT TO BINARY
         STH   R02,MSGAGE  SAVE IT
*
PAGERET  EQU   *
         SAVEC POP
         BR    R07
*
EXPACK   PACK  WORK,PRVAL(1)  TARGET PACK
*
* ****** ALL PARSED UP - ANALYSE WHAT WE'VE GOT
*
XPQ0233  EQU   *
         #IF   TM,ERROR,L'ERROR,O   ON SYNTAX ERRORS
               ICM   R14,B'1111',CONFTOKN
               LA    R15,4 CANCEL
               ENTRC CONF  ANY CONFIRMATION REQUEST
*
               EXITC ,     EXIT
         #EIF
*
```

```
         LH    R02,MSGAGE
         #IF   R02,LE,=F'60'  IF AGE REQUESTED IS DANGEROUS
               OI    CONFREQD,L'CONFREQD  CONFIRMATION REQUIRED
         #EIF
*
         #IF   TM,CONFREQD,L'CONFREQD,O   IF CONFIRMATION REQUIRED
               XR    R01,R01
               ICM   R14,B'1111',CONFTOKN
               SR    R15,R15
               ENTRC CONF  ASK FOR CONFIRMATION
         #ELSE
               ICM   R14,B'1111',CONFTOKN
               LA    R15,4 CANCEL
               ENTRC CONF  ANY CONFIRMATION REQUEST
         #EIF
*
         #IF   TM,ERROR,L'ERROR,O   ON SEMANTIC ERRORS
               ICM   R14,B'1111',CONFTOKN
               LA    R15,4 CANCEL
               ENTRC CONF  ANY CONFIRMATION REQUEST
*
               EXITC ,     EXIT
         #EIF
*
* ****** OK TO PROCEED
*
         .
         .
         .
*
CONFTOKN DC    CL4'XPQ0'
*
* ****** TRANSLATE AND TEST TABLE - TEST FOR NUMERIC
*
TRT1     EQU   *
         DC    256X'FF'
         ORG   TRT1+C'0'
         DC    XL(C'9'-C'0'+1)'0'
         ORG   ,
*
         FINIS ,
         END   ,
```