5.5.9.4.1.2 PDUMP --FORTRAN Dump Routine The dumping routine requires IBCOM. PDUMP, IBCOM and all other FORTRAN modules required are in SYS1.FORTLIB. They should be loaded in at run time using the loader on the load module. (A full description of dumping options are in the "Fortran functions and Service subprograms "; manual.)

For FORTRAN call:

```
CALL PDUMP (A,B,Ø)
Where A is starting address
      B is ending address
      Ø indicates hex dump.
```

For ASSEMBLER call:

```
CALL PDUMP,PARAM(A,B,ZERO),VL=1
       or:
CALL PDUMP
       with Register 1 pointing to   A(A),A(B),A(ZERO)
```

A -- Label to start at

B -- Label to end at

ZERO -- XL1'8Ø',XFL3'Ø' -- End of PARM list flag and zero.

The format acually allows more then one triple of PARM's at a time.

PUDMP sends its output to the standard Fortran print units (usually unit 6 or DDNAME=FTØ6FØØ1).

If normal FORTRAN error recoveries are desired, IBCOM is initialized by:

```
L 15,=V(IBCOM#)
BAL 14,64(15)
```

5.5.9.4.2 Minor Changes to Load Modules--"ZAPS"

Program: IMASPZAP

R.P.I. version: SUPERZAP Location: SYS1.LINKLIB

(See IBM System Guide to Debugging.)

To use on terminal/JCL:

```
//EXEC PGM=SUPERZAP,REGION=24K

//SYSPRINT DD   UNIT=TERMOUT                  --output file

//SYSLIB DD DSN=CL.LLIB.SRES14,DISP=OLD       --load library

//SYSIN DD   UNIT=TERMIN,DCB=BUFNO=1          --input file
```

Commands:

```
NAME 'MEMBER NAME' {'CSECT'} member in SYSLIB file to edit.

BASE XXXXX         --For CSECT who's list does not start at Ø--
      Hex offset   allows using addresses on listing in
                         verify/rep commands.

VERIFY    XXXXX              XXXX,XXXX...
          Offset in Hex        Expected contents

          If found--reprints command; if not does dump.

REP  (Same as verify command.)--replaces contents with given data.

DUMP  'MEMBER'   {'CSECT'}         --dumps as requested
        ALL                          sets no go flag so
                                     cannot rep after a dump.
```

5.5.9.5 BEGIN COMMAND LOGIC

CSECT : EVQBGN   COMMAND : BEGIN

Letter	Return	Offset	Name	
E	1	Ø	E1	Call EVQERR; store old PICA on return
L	3	Ø	L3D	Link to name given as first parameter; move character parameter, if present, to a location to pass to linked program
F	1	12	F1	Restore old PICA
W	2	Ø	W2D	Issue STIMER with given interval
P	2	12	P2	Dump registers, as soon as passed, using PDUMP
	4	12	P3	Dump core requested; use EPHEx to convert characters given into hex
Used by P3			EPHEX	Uses translate instruction pack then shift with offset to convert EBCDIC characters to HEX

5.5.9.6 COPY Command Logic

CSECT : EVQENT ; COMMAND : COPY

```
Open WORKIN, MASPRINT

Read WORKIN

Write MASPRINT

EOF:

    Close WORKIN

    Given number of records printed

    Return
```

5.5.10 Adding Functions The procedure used by the routine identifier to locate the routine is such that adding a function is a relatively minor procedure, however four precautions must be taken:

a) The whole system should be recompiled if any change is made to the QUESLIST DSECT--except for adding at its end, providing its length does not exeed 4Ø88 bytes;

b) Do not any names used elsewhere in the EVQS system--check the assembler's symbol table.

c) Standard conventions should be used as described in any new section;

d) Only one command may start with a specific letter; and only letters are allowed.

To insert the new command:

1) Add a V-constant (external symbol), with the name of the CSECT, to COMA in EVQS, and calculate the offset from the beginning of the list.

2) Place the offset in its proper place in COMT in EVQS. Each letter, in alphabetical order, has one byte containing an offset of its routine--zero is an error routine; four says the routine exists but is not active. (This probably should be done to BEGIN for a regular user's version of the system.)

Two notes: a) COMT has lines as hex constants and as decimal constants--use the proper form depending on the line, b) there are fillers between I & J, R & S because there are gaps in the EBCDIC representations.

5.5.11 Adding Options The forms used to direct the Syntax Parser is such that adding options is easy:

1) Correct the number of character in NUM,

2) Add the new letter(s) on the end of 'LETS',

3) Add the returns flags in corresponding order,

4) Add the address constants to the list and figure their offsets,

5) Add the offsets to the offst list in the same order as the letters,

6) Add the routines required for each letter.

An example of the finished product follows the section on Syntax Parser set-up.

5.5.12 Conventions, MACRO's & DSECTS

Register Conventions --- meets IBM conventions.

Name	Register	Description	Name	Reg.	Desc.
--	0	System & PARM lists	CHAR	9	Pointer to CHAR being processed.
--	1	PARM lists	CNT	10	Work counter
REG2	2		BR1	11	Base-function subroutine -- commands
REG3	3	Work	BASE	12	Base-main CSECT/ 2nd level subroutines
REG4	4		--	13	SAVE area
REG5	5		--	14	Return address
COMBASE	6	Base-ADRLIST DSECT	BR	15	Branch address/work
DCBASE	7	Base-DCBLIST DSECT			
QUESBASE	8	Base-QUESLIST DSECT			

Register Settings

BR1--is set on initial entrance to a function routine to establish addressability for the parameter handling routines.

COMBASE--points to list of form of ADRLIST DSECT--giving the Syntax

Parser the information it needs. It too is set on initial entry to a function routine.

DCBASE--set in system initializing sequence to point to reference list for assign command--it may be used as a work register if it is reset.

QUESBASE--must never be changed--all CSECTS use QUESLIST and changing QUESBASE will destroy its addressability.

Save Areas--in QUESLIST--usually used as shown-- some CSECTS use internal save areas for lower level routines.

SAVEA -- used by 1st level subprogram

SAVE2 -- 2nd

SAVE3 -- 3rd

(IBM Standard) SAVE MACRO is used whenever registers are changed that are needed in a higher level program, especially Register 14.

SAVER MACRO is used whenever a program calls a lower level program--keyword SAVE (default SAVE=SAVEA) loads register 13 with the address of the new SAVE area, after storing the old register 13 value in the second word of the new SAVE area; -- This is standard the IBM chain back conventions; a regular SAVE is also done.

RETURN--when a normal SAVE is done, i.e., SAVE (14,12), a normal return is sufficient, i.e., RETURN (14,12). When a SAVER is done, i.e., SAVER SAVE=SAVEX, --the original SAVE area must be restored hence:

```
L 13,SAVEX+4
RETURN (14,12)
```

The interface between EVQS main CSECT and the function routines is done without saves until the function's GO routine is entered, unless I/O must be done. In the GO routine normal save conventions are followed. The work registers may be used by the function routine.

Registers CNT and BASE must not be changed without a save or specific intent. Changing CNT moves the pointer on the input command card and BASE destroys the main CSECT's addressability.

Save levels--All function subroutines are first level EVQEOQR is second level--but it uses SAVEQ (internal to itself) for its lower level routines. The translators are third level but they use only SAVE (not SAVER).

5.5.12.1 MACRO Uses Starting CSECTS is done with the BEGIN MACRO--It also titles the Assembly listing and establishes CSECT and DSECT addressibility.

Normal I/O is done with the DOIO MACRO.

EODAD's (End of File Address) are establish using the EOF MACRO.

The binary value of an input parameter is determined using the BVALUE MACRO.

Printable form is converted from binary using the EPCVAL MACRO.

Simple QSAM DCB's are generated with the QDCB MACRO. (Self-documented MACRO and DSECT listings are in the Computer Listing Appendix).

5.6 Implementation & Debugging Notes

Since this project has not complete implementation, all JCL procedures and all programs are suspect for errors, as is any system not totally field tested. The following notes should point out some particular trouble spots.

5.6.1 Keypunching & Verifying

--Consistancy in abbreviations and spelling should be checked.

--The addendums (G1,G2) have two form types: that of the addendum itself, and that of the form it is attached on. The second form type may go in one of two places but the first is easier to handle:

--As part 2 of the respondent number of the addendum on the first card of each line.

--As part 2 of the header card's form type.

The RENUM program will transfer the forms in the second place to the first, however checks should be made to assure all G forms use one of these two places; especially, not as part one of the main form's respondent number.

--A maximum of 99,999 of any form type is allowed.

--Correction cards may only have # as EOP's--@ is not allowed on these correction cards.

--Depending on the line printer character chain the listings are printed with raw data listings may look improper--the QN chain on PRINTER 1 is the proper one.

--It is requested by the second card of a job being

/*ROUTEØØPRINTØPRINTER1 (punched exactly as shown.)

--PRINTER 2, in the machine room is faster and its listings are readable; once special characters equivalents are determined--to request PRINTER 2 use: /*ROUTEȸȸPRINTȸPRINTER2

The equivalency table is:

Printer 1 (correct symbols)	Printer 2	
@	'	
#	=	
-	-	(0.8.2 is non-printable)
<	)	
>	ȸ	(No equivalent character.)

5.6.2 RENUM Program

--A word to be checked is the lake name on the header cards. Expecially Lake George; RENUM is set for 'L.G.' but the keypuncher used 'GEORGE'. This should be double checked and RENUM corrected. It is important that all header cards names are spelled the same; and the same as RENUM name list.

--On the same note: RENUM should complain if the name is not matched--it does not now.

--The previous version of RENUM loses the last characters of the first line card--the number lost is the same as the length as the respondent number. Only lines extending to more then one card lose the last characters. Corrections have to be entered in the most recent version but they should be checked.

--Handling the addendum's is another problem--The final version deletes respondent numbers present, but leaves the EOQ following the respondent number. Where addendum numbers are present, whether

--Any special case or binary list output has the modified access number of the first card--regardless of the card the question is on.

--Figure out how to build work lists:

a) So modifiable, and do not have to be totally rebuilt each use, perhaps the user could just change FORM or INDECIES.

b) So can interlace descriptions.

```
eg., I=(5,8,10)   ...
     I=(3,7,9)    which are not allowed now.
```

Building a translation description library may solve part of the problem. See Further Possibilities Logic, Section 5.7.

--The HOLD command is not essential as the system has developed-it should be set automatically in the TALLY command preface.

--Note that @ as the first character of a line stops the GO routine from running--This is useful when debugging input handlers.

--A common error in this system is to use LA (load address) rather then L (load) to retrieve a stored address--the reverse also occurs.

--No translator can handle any SPF. SPF1 is especially tricky since all indecies are in fact parts of the same response.

--Formated list can be used to combine a number of fixed form questions on one card--margins must be set for no folw over and EOQ's or other delimiter must be retained although the newly fixed columns may be sufficient.

--The SCAN routine may be tested by scanning to the terminal: issue an ASSIGN: OUTPUT=TERMINAL or A: O=T

(But only with FSAM inactive.)

as parts of header card form type; or respondent number, they are placed before this initial EOQ. If no addendum is present, the first EOQ follow a null string.

5.6.3 EVQS Terminal System

--Once debugging is complete the 'TERM' option on the loader should be removed; it will eliminate the IEW1012 messages that precede the EVQS system heading.

--To terminate execution, enter /* ; the return code has no meaning.

--EVQEOQR may not function properly if CARD=1 is given as a SCAN option. This is because cards may not have EOL which EVQEOQR needs for its operations.

--An inconsistancy in commands which may cause trouble is

```
SCAN:  FORM=A7,PAGE=Ø2  vs.
TALLY:  FORM=A7Ø2
```

--Using a normal SAVE macro can cause trouble--if it is with identifier; eg.,

```
JUNK     SAVE    (14,12),;*
```

This form assumes BALR 14,15 was used to reach it and so uses REG 15 to branch around the identifier. If BAL 14, GETSTORE is used--as was before correction--the branch will land in the middle of garbage.

--Note that dashes in the comments text or other responses must be replaced by underscores; or else, they will be mistaken for EOQ's. eg., fertilizer 20-10-5 becomes 20_10_5

--EVQIF when handling IF ~~test~~ requires the test flag in TINTFG. This was originally an IFFILL flag but the fill operation was eliminated for IF. It is still needed to determine if the bit is the first to be turned on and so need initialization or is later bits.

--Note that FSAM always skips the first record--so one blank card is written on EVQSFILE at generation time also EVQSFILE should be allocated in contigious cylinders.

--The Routine Identifier Section of the EVQS CSECT does not check for invalid symbols as first letters of the word before the colon. If an invalid symbol is entered--an illegal branch will be made--and an ØC1 system error usually results, such a check should be added.

--EVQSLLIB should be able to hold about 4ØK of programs--1 cylinder on 2314 is sufficient.

--No abbreviations are allowed in the QUERY command parameters--it would be wise to add such a reminder 'no abbreviations allowed' to the 'no such QUERY NAME' message.

5.6.4 Doing Dumps from Terminal

LOAD 1, LOC 'ENTRY ADDRESS'

(Lists entry address for loader output.)

LOAD 2,LIST 1ØØ - 3ØØ, LOC 'REGS AT ENTRY TO ABEND'.

(Lists header, comp. code, and last PSW LOC give line number of registers.)

Then LIST the number of registers line - #+300

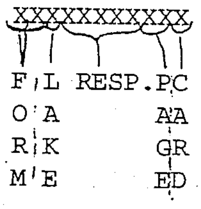
Ignore WORKSPACE EXCEEDED: it just means the whole dump is not available--uaually the end lost is not important.

```
to do ZAP:    SUB ZAP, ACCT=(11Ø1),MODE=CONV

      NAME      'MEMBER'    'CSECT'

      VERIFY    '000000'    ' ______ ', '(______)

      REP       '000000'    ______      ______
```

To do summary of SYS msg. do LOAD O,LOC 'STEP'

or LOC'CPU'; LOC'CODE'.

--This is shown in the SCAN command but not explained: FORM keyword can be used to specify LAKE (& RESP), likewise PAGE can also specify CARD, because of the layout of the access number:



5.6.5 QLCOND Section of EVQEOQR The individual tests need work, each test must compare the pattern with the translate bit flags to determine if as requested:

OR -- At least one bit that is on in the pattern is on in the translate flags.

AND -- All that are on in the pattern are on in the flags.

XOR --One and only one of the bits on in the pattern is on in the flags.

These are done by:

AND - AND pattern and flags, XOR pattern and result; then go if final to zero result.

OR - AND pattern and flags; if greater then zero go

XOR - AND pattern and flags; if equals zero stop; otherwise; shift one to right and XOR with 1 when shifted 32 if XOR'd VALUE equals one; go.

5.6.6. Test Sequence These next pages contain simple test sequences, what works and proper sections. (✓ indicates it works.)

✓	ASSIGN: OUTPUT=3
✓input function only	SCAN: FORM=A3
✓input function; not fill or go functions	TALLY: H,FORM=A302,INDEX=02,OP=3
	TRANS: CHAR=(M,F)
	IF: TEST=02
input ✓	TALLY: FORM=A305,IND=01,OP=2,GO
✓	ASGN: SCAN=2
input ✓	SCAN: PAGE=05
1→ SCAN→3	SCAN: FORM=A3
✓	A: SCAN=2,OUT=3
2→ SCAN→ 3.	SCAN: PAGE=05
✓	A: IN=3
	COPY:
✓	A: OUT=T
	SCAN: PAGE=05
✓	TALLY: H,F=A302,I=02,OP=3

The results of some of previous commands:

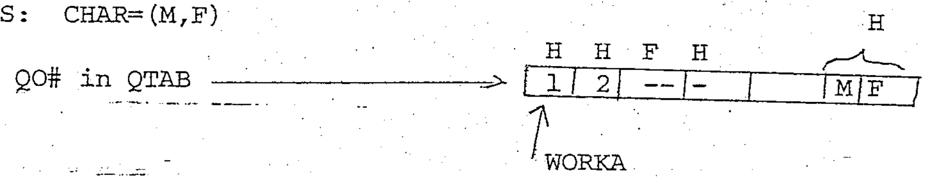
PTYPE=1,TFCHK=A3Ø2

TQNUM=1,TQUES=Ø2

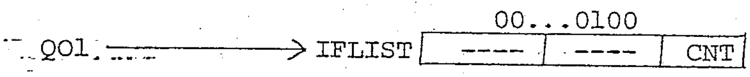
ACTIVE+2 OPTION=X'Ø4'

ACTIVE X'Ø8'

TRANS: CHAR=(M,F)

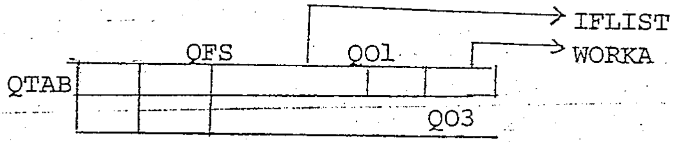


IF: TEST=02



IFNUM=4

TALLY: FORM=A305



PTYPE=2,TFCHK=▭ A302 A302

T: IND=01,OP=2

TQNUM=1 TQUES=0002

ACTIVE+2 OPTION=X'02'

ACTIVE X'0.8.2.'

??? T: GO

5.7 Additional Possibilities Logic

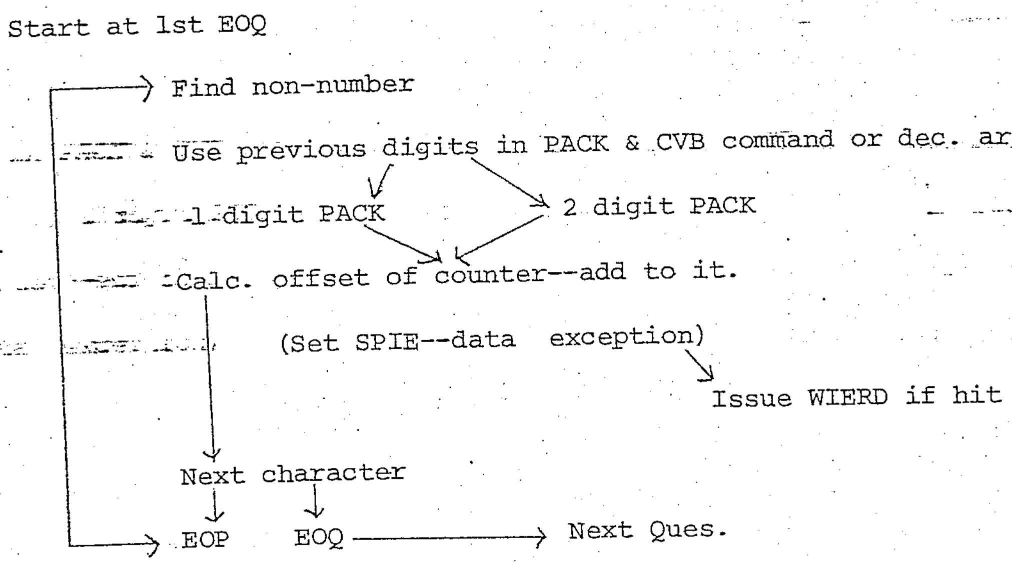
5.7.1 QUERY:

Add a 'DIRECTORY', Form 1, keyword. When given, it lists all entries in the DOCUMN PDS directory in three sets: HELP, ITEM, FORM/PAGE. ALL is an alternative keyword for the same purpose.

5.7.2 Numeric Parameters:

The BVALUE MACRO--could be changed to do an execute on the length of the parameter with a PACK instruction--however an SPIE should be issued to recover any data exceptions from the CVB (convert to Binary) instruction and bad data.

5.7.3 NMC/AMC Translators


Start at 1st EOQ
Find non-number
Use previous digits in PACK & CVB command or dec. arith.
1 digit PACK
2 digit PACK
Calc. offset of counter--add to it.
(Set SPIE--data exception)
Issue WIERD if hit
Next character
EOP
EOQ
Next Ques.

ALPHANUMERIC Same as above but have a tableto correct numeric forletters and add numeric value of letter to corrected number.

EXAMPLE:	1	a	b	c	2	a	b	3	a	b	c	d	4	a
Correction:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Table Entries:	1				5			8					13	
Numeric Values:		1	2	3		1	2		1	2	3	4		1

5.7.4 Misc. The preliminary report done by this author on this project should be reference concerning fixed format data and data sets for this data.

Provisions for generating back-up files should be added.

Automatic look-up tables could use either FSAM or Direct Access. The translation definitions should be implemented first. The entry should be reference by form and quesiton number; and should contain page, and index number and the translation definition.

Later tables could contain name scanning; i.e., SCAN: NAME=FISHERMAN calls out the requested form definition values; FORM, PAGE, & especially Respondent Number ranges.

Minor statistical figures--averages, mean, standard deviation could be added to Fortran output routines.

5.7.5 Pattern Parser

(Possible Pattern Parser Command)

Format: **Form**

LEN=XX --length of response 2

ESTIMATE: --length given is estimate otherwise assumed as maximum 1

CHAR=9'ABC+ --sets format definition characters 2

Delimiters allowed

9 -- numbers

' -- multiple of following characters

A -- letters

B -- blanks

+ -- following character is optional

C -- specific character string

$\text{CHAR=(}\underbrace{\text{string}}_{\text{C1}},\underbrace{\text{string}}_{\text{C2}},\underbrace{\text{string/string}}_{\text{C3}}\text{)}$ (either string)

Examples Using Format Parser

FORM=('9'B'A'B) matches 123b̸XYZb̸b̸

FORM=(29,'B,2A,'B,C1),CHAR=(IO/IB)

matches 72b̸ABb̸IO

not 72b̸ABCb̸OB

CHAR=(FT,HP,OB/IB)

$\text{FORM (}\underbrace{\text{29+C1}}_{1}\text{,'B,}\underbrace{\text{2A+c2}}_{2}\text{,'B,}\underbrace{\text{C3}}_{3}\text{),}$

Matches SPF3 and returns 3 segments--1 for each string matched.

5.7.6 Other Implementations

Must change length of sequence number.

Change level of punctuations's meanings.

Redo SCAN; perhaps with direct access if FSAM is not fully available.