THE EVQS INFORMATION ACCESS SYSTEM

A COMPLETE APPROACH TO HANDLING COMPLEX FORMAT DATA, APPLIED TO AN ENVIRONMENTAL QUALITY SURVEY.

by

Richard L. Roth

A project Submitted to the Graduate

Faculty of Rensselaer Polytechnic Institute

in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF ENGINEERING

Approved:

_____

Frank DiCesare, Thesis Adviser

_____

Paul Daitch, Engineering Science Curriculum Chairman

Rensselaer Polytechnic Institute

Troy, New York

June 1974

CONTENTS

List of Tables ix

List of Figures xi

Information Access Guides. xiii

Acknowledgement. xvi

Abstract xvii

1. Introductions. 1

1.1 Introduction to Environmental Quality Survey. 1
1.2 Introduction to Data Formating. 5
1.3 Introduction to EVQS Punch Format 9
1.4 Introduction to Verification and Final Processing . . 11
1.5 Introduction to EVQS On-Line Analysis System. 13

2. Finales. 15

2.1 Results 15
2.2 Conclusions 20
2.3 Discussion of Further Possibilities 22

3. User's Manuals 24

3.1 Keypunch. 24

3.1.1 Introduction 24
3.1.2 Punctuation Symbol Summary 31
3.1.3 General Format 32
3.1.4 Abbreviations. 34
3.1.5 Special Cases. 39

3.2 Data Verification 45

3.2.1 Introduction 45
3.2.2 Obvious Corrections. 48
3.2.3 Obvious Error Clues. 49
3.2.4 Full Listing 49
3.2.5 Final Processing & Later Errors. 50

3.3 EVQS On-Line Analysis System. 51

3.3.1 Introduction 51
3.3.2 Initializing the System. 51

3.3.3 System Operation. 51

3.3.3.1 Locating A Response. 51
3.3.3.2 Processing A Response. 53

3.3.4 Command Syntax. 56
3.3.5 System Organization 57
3.3.6 Using the Commands. 57

3.3.6.1 ASSIGN 57
3.3.6.2 QUERY. 59
3.3.6.3 SCAN 60
3.3.6.4 Evaluations. 61

3.3.6.4.1 TALLY 63
3.3.6.4.2 TRANSLATE 64
3.3.6.4.3 Conditional testing . . . 66
3.3.6.4.4 Multiple Translations . . 67
3.3.6.4.5 Executing 67

3.3.7 Analysis Output Units 70

3.3.7.1 Wierd,99,NR and Binary Tally Format. 71

3.3.8 Command Descriptions. 72

3.3.8.1 ASSIGN Command 72
3.3.8.2 COPY 73
3.3.8.3 IF: Conditional Test. 74
3.3.8.4 QUERY Command. 75
3.3.8.5 SCAN Command 76
3.3.8.6 TALLY Command. 77
3.3.8.7 TRANSLATE Command. 78

3.3.9 Command Summary 79

3.4 Annotated Questionnaires 80

3.4.1 Introduction. 80
3.4.2 A3. 81
3.4.3 A4. 86
3.4.4 A5. 91
3.4.5 A7. 96
3.4.6 B2. 101
3.4.7 B3. 107
3.4.8 B4. 112
3.4.9 B5. 117
3.4.10 C4 123

3.4.11 C5. 129
3.4.12 D4. 135
3.4.13 D5. 141
3.4.14 E4. 146
3.4.15 E5. 152
3.4.16 F2. 158
3.4.17 F3. 164
3.4.18 G1. 170
3.4.19 G2. 172

3.5 Code Sheet Page Summary Tables. 175

3.5.1 Introduction 175
3.5.2 Forms A3 and A5. 176
3.5.3 Form A4. 178
3.5.4 Form A7. 180
3.5.5 Form B2. 183
3.5.6 Forms B3 & B4. 185
3.5.7 Form B5. 188
3.5.8 Form C4. 191
3.5.9 Form D4. 194
3.5.10 Form E4 196
3.5.11 Form F2 199
3.5.12 Form F3 202
3.5.13 Form G1 206
3.5.14 Form G2 208

4. Programmer's Guide. 211

4.1 Introduction 211
4.2 Main Computer Runs 212
4.3 Programs 213

4.3.1 Compiling/Assembling. 213
4.3.2 PREPNH. 213
4.3.3 Invoking Terminal System. 214

4.4 Procedures 214

4.4.1 Verification. 215
4.4.2 Terminal Job Control Language 216

4.5 Data Sets. 217

4.5.1 Description of Data Files 218
4.5.2 Terminal System Documentation File. 218

5. Program Logic Manuals 219

5.1 PREPNH PLM 219

5.1.1 Parameter List. 219

5.2 OBCORR 220
5.3 VRFY 224

5.3.1 Special Actions 224

5.4 RENUM PLM. 224
5.5 EVQS System Programmer's Guide and Analysis Program Logic Manual 228

5.5.1 Introduction to System Description 229
5.5.2 The On-Line Analysis System 231
5.5.3 MACRO's and DSECT's 234
5.5.4 Command Handler 235

5.5.4.1 Routine Identifier 236
5.5.4.2 Keyword Parameter Syntax Parser. . . 236

5.5.4.2.1 Logic 236
5.5.4.2.2 Operation 237
5.5.4.2.3 Grammer Syntax. 240
5.5.4.2.4 Syntax Parser Set-Up. . . 244

5.5.5 Initializing. 247
5.5.6 I/O Operations. 247

5.5.6.1 Data Control Blocks. 248
5.5.6.2 Associating DCB's with Files 248
5.5.6.3 I/O Macro's. 249
5.5.6.4 DCB locations CSECT Name: EVQFILES. 251
5.5.6.5 ASSIGN Command Logic CSECT Name: EVQASIGN 252
5.5.6.6 File Scanning. 254

5.5.6.6.1 SCAN Command Logic CSECT Name: EVQSCAN. 256

5.5.7 The Analysis Routines 258

5.5.7.1 Translation Work Lists 262

5.5.7.1.1 Building Prototypes . . . 265
5.5.7.1.2 Building Sequence 265

5.5.7.2 Flag Description Tables. 267

5.5.7.3 Specific Work Lists 271
5.5.7.4 IF & TALLY Input Handlers 275

5.5.7.4.1 Logic Tables 275
5.5.7.4.2 Variables. 277
5.5.7.4.3 IF Input Logic 277
5.5.7.4.4 TALLY Input Logic. . . . 278
5.5.7.4.5 Storage Manager. 280

5.5.7.5 Evaluative Routines - EVQEOQR . . . 282

5.5.7.5.1 Action Routines. 283

5.5.7.5.1.1 QLLIST-List 284
5.5.7.5.1.2 QLTALLY - Tally . . 285
5.5.7.5.1.3 QLBINARY - Binary List 286
5.5.7.5.1.4 QLCOND. . . 288

5.5.7.5.2 Translation Routines . . 289

5.5.7.6 Analysis Outputs. 290

5.5.7.6.1 Tally Output Routine -- EVQTYOUT. 290
5.5.7.6.2 FORTRAN Print Routine - EVQFORPT. 291

5.5.8 Documentation Function - QUERY 293
5.5.9 System Programmer's Functions. 295

5.5.9.1 BEGIN Command 296
5.5.9.2 List Command COPY 297
5.5.9.3 Error Recovery Commands 297
5.5.9.4 Logic Notes. 299

5.5.9.4.1 Error Facilities 299

5.5.9.4.1.1 SPIE. . . . 299
5.5.9.4.1.2 PDUMP . . . 300

5.5.9.4.2 Minor Changes to Load Modules. 301

5.5.9.5 BEGIN Command Logic 302
5.5.9.6 COPY Command Logic. 302

5.5.10 Adding Functions. 303
5.5.11 Adding Options. 304

5.5.12 Conventions, MACRO's & DSECT's. 304

5.5.12.1 MACRO Uses. 306

5.6 Implementation & Debugging Notes. 307

5.6.1 Keypunching & Verifying. 307
5.6.2 RENUM Program. 308
5.6.3 EVQS Terminal System 309
5.6.4 Doing Dumps from Terminal. 311
5.6.5 QLCOND Section of EVQEOQR. 312
5.6.6 Test Sequence. 313

5.7 Additional Possibilities Logic. 315

5.7.1 QUERY. 315
5.7.2 Numeric Parameters 315
5.7.3 NMC/AMC Translators. 315
5.7.4 Misc 316
5.7.5 Pattern Parser 317
5.7.6 Other Implementations. 318

APPENDIX. 319

Literature Cited. 393

LIST OF TABLES

Table	Title	Page
1.1	Questionnaire Categories	2
1.2	Response Types	3
1.3	Response Type Translators	4
1.4	Command Summary	14
2.1	State of Punching	16
3.1	Drum Card Set-Up	30
3.2	Table of Indicators	31
3.3	Abbreviations	35
3.4	Questions referred to by Special Formats	44
4.1	Main Computer Runs	212
4.2	Computer Programs	213
4.3	Computer Procedures	214
4.4	Computer Data sets	217
4.5	Present Data Set Names	217
5.1	Terminal Routines	233
5.2	Summary of DSECT's	234
5.3	Summary of Registers	234
5.4	EVQS Standard Command Format	235
5.5	Grammar Syntax	240
5.6	Grammar Symbols	241
5.7	Action Symbols	242
5.8	Command Description Form	246
5.9	Planning Offset Structure	246
5.10	Standard DCB Address Variables	248

5.11 File Options. 248
5.12 Detailed File Usage 249
5.13 SVC Option Bytes. 250
5.14 File Deginitions. 251
5.15 Default Files 252
5.16 DCBLIST DSECT 252
5.17 FSAM Control Block. 254
5.18 List of Translators 259
5.19 'ACTIVE' Flags. 267
5.20 QTAB Table & QTLIST DSECT 268
5.21 Source of Entries in QTAB 268
5.22 Individual Question Flags, QFS in QTLIST. 269
5.23 IF Input Handler. 275
5.24 TALLY Input Handler 276
5.25 Function Output Units 290
5.26 System Programmer's Functions 295

LIST OF FIGURES

Figure	Title	Page
1.1	System Flow Chart	xviii
1.2	Lake Parameters	1
3.1	Response Divisions	26
3.2	Keypunch Drum Card	29
3.3	Sample Code Sheets	36
3.4	Terminal system organization	58
5.1	PREPNH Flow Chart	219
5.2	OBCORR Flow Chart	221
5.3	VRFY Flow Chart	222
5.4	Access Number Format (11 characters)	225
5.5	RENUM Flow Chart	226
5.6	EVQS Terminal System Control Sections	232
5.7	Syntax Parser Forms	239
5.8	Standard Files	247
5.9	Scan Pattern Format	255
5.10	SCAN GO Routine Flow Chart	257
5.11	Overview of Analysis Routines	258
5.12	Analysis Routines	260
5.13	Work List Linkage	261
5.14	Basic Work Lists	263
5.15	Conditional List Variable & List	270
5.16	Specific Work Lists	271
	5.16a CHAR	271
	5.16b MATCH	272

5.16c NMC. 273

5.16d AMC. 273

5.16e RANGE. 274

5.17 Evaluative Routines 282

5.18 List Action Routine 284

5.19 Tally Action Routine -- QLTALLY 285

5.20 Binary List Action Routine -- QLBINARY. 286

5.21 COND Evaluation Routine -- Logic QLCOND 288

5.22 CHAR Translator -- EVQCST 289

5.23 QUERY Flow Chart. 294

INFORMATION ACCESS GUIDES

(Important Information & the Sections for each.)

I)	For the Keypuncher Keypuncher's Manual --Section		3.1
	Basic Data Divisions		3.1.1
	Corresponding Punch Divisions		3.1.2
	Punch Card Layout		3.1.3
	Abbreviations		3.1.4
	Special Purpose Forms		3.1.4
	Sample Code Sheets	Figure	3.3
	Punched Sample Code Sheets	Appendix	A
	Keypunch Rules & Formats Updates	Appendix	B
	Other Aspects of the Keypuncher's Manual		
II)	For the Verifier Verifier's Manual --Section		3.2
	All the Keypuncher's Information		GuideII
	Introduction to Verification		1.4.3.2.1
	Obvious Correction Approach		3.2.2
	Sample of OBCORR listing (have data manager run one) c		
	Typical Obvious Errors		3.2.3
	Correction Punch Format		3.2.2
	Verification Computer Control Cards		4.1.1
III)	For the Terminal User Terminal User's Manual Section		3.3
	Introduction to Questionnaires		1.1
	Introduction to Terminal System		3.3.1
	Locating A Response		3.3.3.1
	The Questionnaire Package (skim)		3.4,3.5

Command Syntax 3.3.4

The Command Summaries (preview) 3.3.8 esp. 3.3.9

Command Descriptions 3.3.6

Alpha time-sharing System Manual (skim) from R.P.I. OCS

Invocation Commands 3.3.2

Remainder of Terminal User's Manual

IV) For the Non-Terminal Data User

To understand the data format, read the

Keypuncher's Information.

To locate data.

Locating a Response 3.3.3.1

Access Number Definition 4.4.1

The Questionnaire Package (skim) 3.4,3.5

V) For the Data Manager

Introductions (skim) 1

Keypuncher's Manual (skim) 3.1

Verifier's Manual (skim) 3.2

Introduction to the Terminal User's Manual 3.3.1

File Operations 4.1

Initial Set-Up Processes 4.2

Prepunching cards 4.3.2

Verifying Process 4.4.1

Final Processing 4.4.1

Back-up Files and Recovering from Errors 5.7.4

Complete report excluding Program Logic Manuals (READ)

VI) For the System Programmer

Introductions 1

User's Manuals	(skim)	3
Programmer's Guide		4
System Programmer's Manual	(skim)	5
Implementation and Debugging Notes		5.6
Additional Possibilities Logic		5.7
Program Logic Manuals	(Read)	5
Remainder of the Report		

ACKNOWLEDGEMENT

In addition to all those who helped with various details, I'd like to thank three special people:

Jack and Yash who have had a large part in molding this project; esp. Jack for reading the rough draft and Yash for doing all the data keypunching.

And most important; without whose help neither the project or this report would ever have been finished--going so far as to put off our honeymoon until this is complete--my wife, Sharon, who was both typist and editor.

ABSTRACT

The objective of this project is to develop a complete scheme for converting a large unwieldy limited/free format data source into an easily accessed information supply. The system consists of two parts:

Digitization--converting the data into a computer readable form; and verifying its accuracy,

On-line analysis--a computer system to allow easy access to the previously encoded data.

The whole process was originally developed for, and partially tested on, a series of questionnaires used by K. Jack Kooyoomjian, of the Fresh Water Institute, for research into perception of water quality on a recreational area.

The digitization stage has three parts: keypunching, verifying, and access number generation. The last two are done using three specially written computer programs.

The analysis system was written for use with Rensselaer Polytechnic Institute's Alpha Time Sharing System on an IBM 360/50 but will work on any IBM OS facility of equal or higher compatable level. A locally developed access method--FSAM--is also used to increase the scanning efficiency.

The encoded data was developed as part of his doctoral work, by K. Jack Kooyoomjian. The work was done, as part of Rensselaer Polytechnic Institute's Fresh Water Institute, at Lake George on perception of water quality in recreational areas.

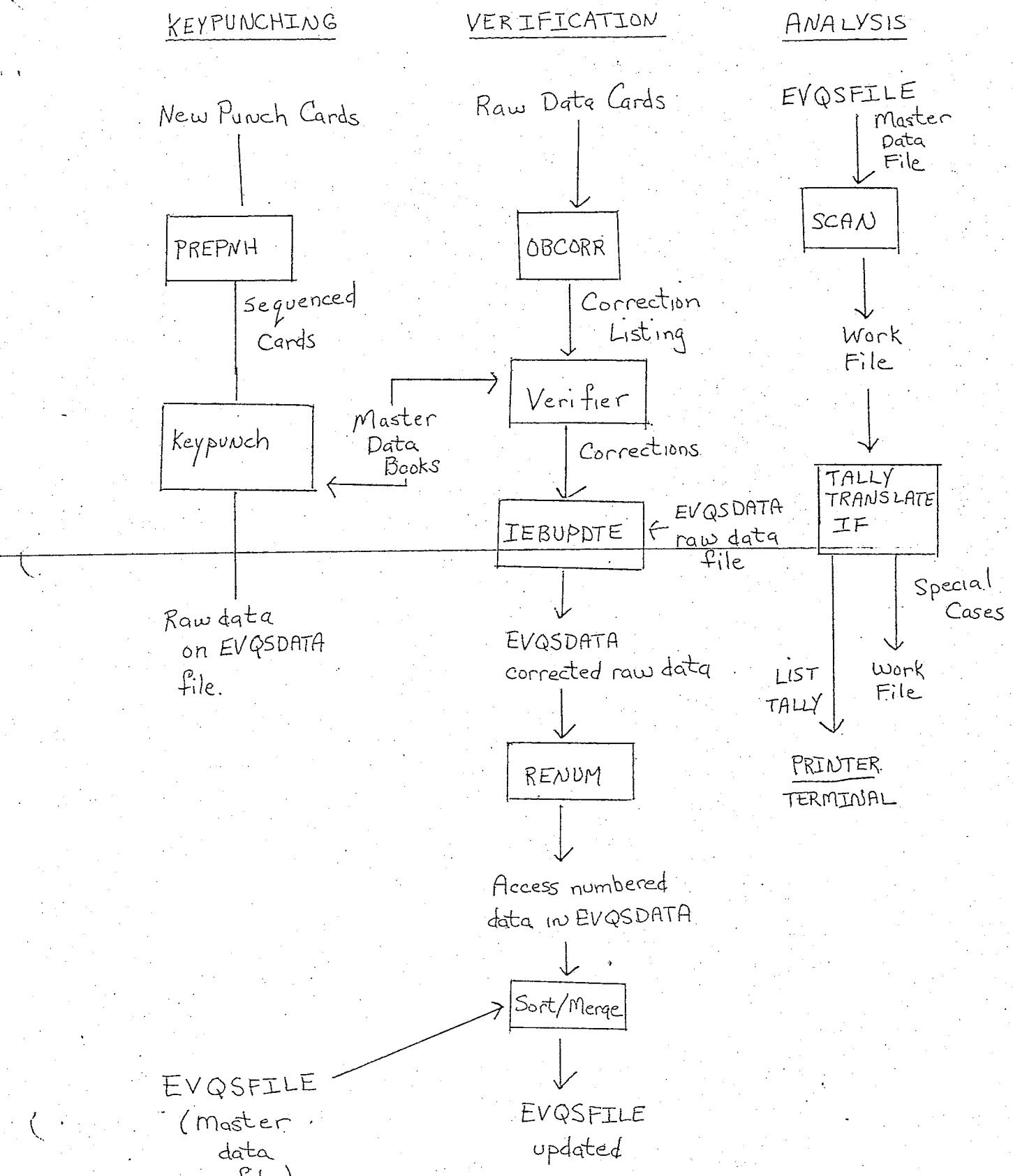

Figure 1.1 System Flow Chart

PART 1

INTRODUCTIONS

1.1 Introduction to Environmental Quality Survey

The ENVIRONMENTAL QUALITY SURVEY (EVQS) was developed by K. Jack Kooyoomjian as the basis for a Phd in Environmental Engineering. The survey was conducted under the auspices of the Fresh Water Institute, an Environmental Research Center stationed on Lake George and affiliated with Rensselaer Polytechnic Institute.

This survey research program has concentrated on the study of the various impacts of water quality by various users on recreational lakes. Four lakes were selected: Lake George, (a large oligotrophic lake), Oneida Lake, (a large eutrophic lake), Schroon Lake and Saratoga Lake, (intermediate-sized oligotrophic and eutrophic lakes respectively). Three are in the Adirondack Forest Preserve of Upstate New York and the fourth, Oneida, is further west in north central New York, near Syracuse. These four were chosen because they comprise various pairs of two conditions:

1) Large, developed for recreation lake vs. intermediate sized lake,

2) Clean, oligotrophic lake vs. nutrient-rich eutrophic lake.

	CLEAN	NUTRIENT-RICH
Large (20 miles)	Lake George	Oneida Lake
Intermediate (4-9 miles)	Schroon Lake	Saratoga Lake

Figure 1.2 Lake Parameters

53,391 questionnaires were given out around the lake to various groups during the years 1970 - 1971.

TYPE	CATEGORY	NUMBER DISTRIBUTED	RETURNED	PER CENT
A	Recreationist	29,574	4,368	14.77
B	Cottage & Home Owner	7,151	859	12.01
C	Hotel - Motel	242	55	22.73
D	Commercial, Non-Lodging	304	70	23.03
E	Marina	50	14	28.00
F	Fishing	5,453	390	7.15
G	Boat Owners (Addendum to A's and B's)	10,617	1,067	10.05
TOTALS		53,391	6,823	12.77

Table 1.1 Questionnaire Categories

The questionnaires were phrased in a highly open manner which was highly beneficial, because the amount of information, added to the questionnaire, by annotations and comments was substantial. (NOTE: These are relatively environmentally conscious areas.) The adding of such comments makes computerizing much more complex as provision must be made to handle them. The returned questionnaires were transcribed into master data books according to the codes given in a later section. These nineteen master data books are being used, through hand tallies for Mr. Kooyoomjian's thesis work and were used as the basis for keypunching the data.

The variable number of question types, and the necessity of using the master data books directly, required the special format subsequently developed.

The question types can be broken down into three major categories: character strings, numerics, and multiple choice. The last category can again be split into two forms: Choose one, i.e. sex--male/female; or check those applicable, i.e. recreational activities.

The following table shows the proportions of the questionnaire comprised by each type and the quantity of responses resulting from each. The proportions are different because: a) Character string responses average three parts, i.e., town--county--state; b) "Check those applicable" responses average five check marks and may be as high as thirty.

TYPE	% OF QUESTIONS ON QUESTIONNAIRE	% OF RESPONSE
Character String	10	15
Multiple Choice--Choose One	50	25
--Check Applicable	20	50
Numerics	20	10

Table 1.2 Response Types

A number of special handling approaches were used to deal with the many variations on the major categories. The relationships and proportions are shown below.

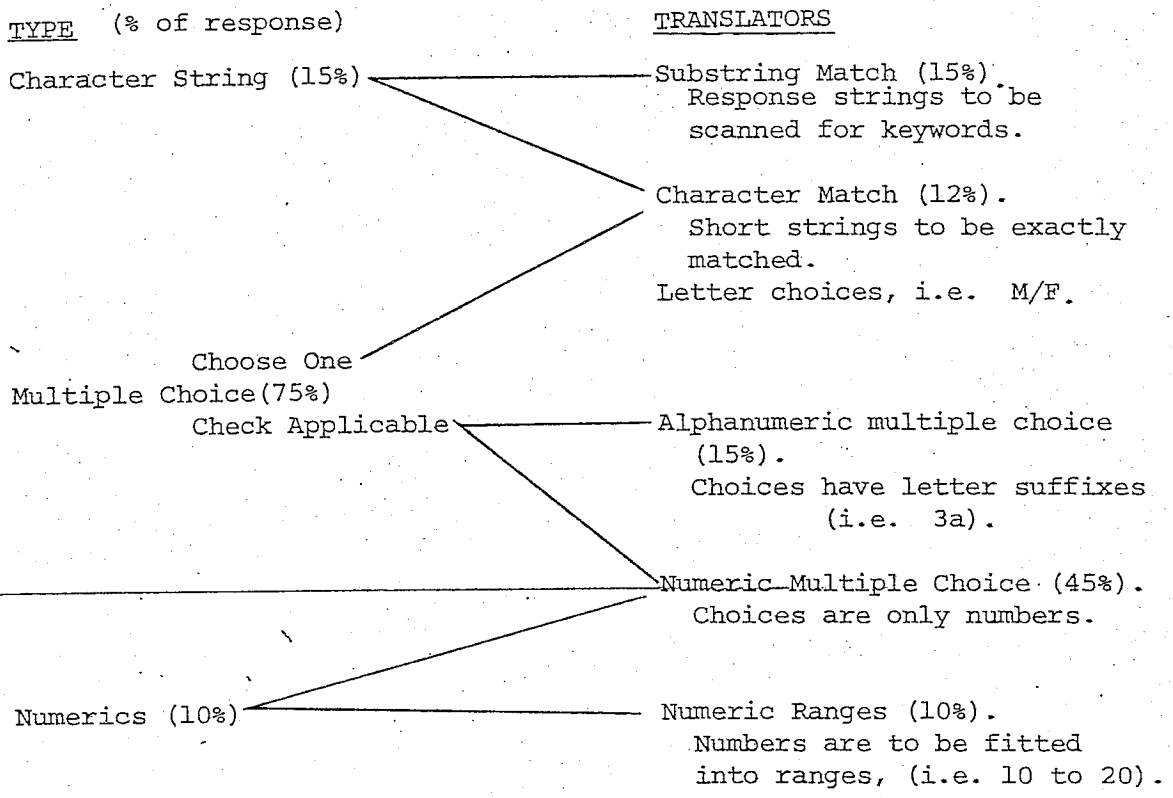

Table 1.3 Response Type Translators

1.2 Introduction to Data Formating

There have been developed a large number of approaches to data management, and the purpose of all of them is to associate a description with a specific data item. The extent of the description required and the number of data items associated with one description is what distinguishes a Fortran format read statement from a Management Information System (MIS). Regardless of the complexity of arriving at the description of the data item, the number of actual descriptions is rather limited: fixed, limited, and free formats. (The image of a punch card is used in these illustrations.)

FIXED FORMAT--A data item is defined to have a specific width and occur in a particular column of a set of identically formated cards (eg. Fortran formated read).

LIMITED FORMAT--A data item is one of a limited number of forms and occurs after a specified number of delimiters from the beginning of a card, in a set of identical cards (eg. Fortran Read / Print Statements).

FREE FORMAT--A data item consists of a string of characters containing a requested keyword--out of a set of such strings, each of which has a specified length (eg. accessing an abstract of a scientific paper).

(These are, of course, simplistic descriptions of the three possible data formats.)

The fixed format is the most widely used in the computer field; for the simple reason most data is numbers. And whether scientific, real or integer notation is used, fixed column and width are easily met criteria. The purpose of data management systems working with such data is to use some data items on each card (or record) as descriptors to determine if the other items on the record are those requested by the user. Easy input descriptions, clear output tables and analysis functions are other features usually provided.

The free format has relatively recently been developed, as computers begin to aid in the functioning of libraries. Text materials usually are keypunched directly from originals. The only formating being eighty character cards. More sophisticated techniques use prefixes containing important words relating to each article or complex indices to more quickly reference the text selection. Once the individual selection is located, the only analysis that may be done in character scanning for other critical keywords.

Limited Formating has not been vastly used in the computer world for data entry and it is only used to introduce beginners to programming or for small data requirements. What is usually done is to allow real, integer or exponential numbers to be inputed using commas or blanks to separate them. A direct correspondence is then established between the input values and the variable list.

The development of complex information management systems using structured sub-files in the last few years has been oriented chiefly in two directions:

a) Management Information Systems (MIS's), designed for industrial executives. These use mainly fixed format, even rarely where characters strings are used, a fixed length is established and any space not used is left blank names;

b) Library access systems, creating on-line card catalogs and libraries, free format is used for the text with the indices stored internally or in small fixed files.

The major work with fixed format was previously developed using simple files, for business/accounting applications. The use of computers in the social sciences, is also a newly emerging field; yet all systems developed fall in the two extreme catagories: fixed format for data handling and free format for text recovery or syntactical analysis of natural languages. Since most social scientists are beginner programmers, analysis packages occasionally allow for limited format data entry.

The preference of fixed over both free and limited formating is an obvious choice. Converting printable numeric data to internal form and storing in the proper location is greatly simplified if the beginning and end of the character representation of the number is known. Limited format requires searching for the delimiters and free format requires a total search. Handling character strings are very different: Fixed format requires allowing a fixed length for the string--if the string is too short, the rest is wasted; too long, nothing can be done without complex correction procedures. Free Format provides little information beyond what may be gained through syntactical content analysis and is usually a fairly complex procedure. Limited format allows associating

various internal meanings with variable length character strings, at the expense of the time required to separate the strings by finding delimiters and the programming required to handle such variability.

1.3 Introduction to EVQS Punch Format

Computerizing a data bank as complex as this one requires some special approaches. The complexity lies in the variability of responses; rather than in the relationship between data items. The latter is the problem of Management Information Systems (MIS's) where much work has been done. Unfortunately, the majority of the data handled by MIS's is fixed format data. The format of each set of data items is rigidly fixed. The work that has been done to accommodate free format data is attempting to use the computer as a natural language in order to access texts. The approach used is to work strictly with character strings, segmenting texts by various parameters: author, title, important words, and identifying further by straight work matching techniques.

Neither of these techniques, MIS nor text handling, can fully encompass the EVQS Data bank. The rigid format of the MIS requires the data to be reduced to a fixed format before digitizing; defeating, to a large extent, the goal of allowing computer analysis of all the data. This is so because any reduction attempts must be based on specific knowledge of the data and the user's orientation. The first cannot be manually obtained, from a data base this large, without excessive work: the second brands any reduction attempt of limited usefulness. The library approach is primarily aimed at natural languages (eg., English) and must be capable of handling not only variable length character strings (words) but dealing with the complex syntaxtical structure--either by parsing it, in a rough manner, or simply searching for keys and displaying for user analysis. The texts units

are usually considerably longer than those in EVQS--where one to four words is typical. Considering a questionnaire as a text unit fails to use the information known about the organization in each question.

Analysis of the collected data lead to the use of the most basic and ancient form of data organization as a basis for the EVQS encoding system; namely, punctuation delimiters. Just as periods, commas and semi-colons separate the responses into logical units; so special delimiters were chosen to separate the responses into logical units. On a higher level, books, chapters and paragraphs serve similar purposes using the sentence as the basic unit. Similar divisions were developed using a code-book line as the basic unit--it being a fifth or so of the questionnaire. The basic analysis operation is still character searching; however, the limited format allows only the delimiters to be looked for until the requested question is found.

Since character searching is a time-consuming operation, especially if it must be repeated many times; it was decided the basic goal of the analysis system was to convert the raw data, encoded using the punctuation technique, into a fixed format. This fixed format may then be submitted to other mathmatical programs to establish the underlying relationships in the data. Note that, since ALL the data are accessable through the computer, the conversion to fixed format is under COMPLETE CONTROL of the user, rather then based on random samples or a rough feel for the data and hand tallies. If one such conversion does not yield useful relationships, a new conversion is merely a matter of another session on the terminal with analysis system; rather then keypunching another set of data.

1.4 Introduction to Verification and Final Processing

Necessity of Verifying Data The immense complexity and variability of the EVQS data is such that errors in keypunching can happen fairly easily and unless someone has extensive experience, with the raw data, they would never see the discrepancies. This is expecially true once the data are being used with the analysis package; bad data could simply be matched or skipped as a wierd value and never noticed. Therefore it is essential the final data base be as accurate as possible; it is so large that corrections to it are expensive; and illogical (although possible) if data can be corrected in the small units it is entered in. The usual verification procedure of punching twice is foolish for data as complex and time consuming as this and so other checks have been designed.

The verification process has two stages: quick, obvious errors and full, exact comparisons. Preliminary tests have shown that once the professional keypuncher has gained a few weeks experience with the limited format, 90 - 95% of the punch errors will be caught by the obvious error techniques. The full comparison is so time-consuming to both computer and personnel, that it will probably only be used in training and for complex errors. The remaining 5 - 10% is often transcription/interpretation errors from the original questionnaire and sometimes may not even be resolved by going back to the original data.

Quick Corrections This process is mainly accomplished by a trained scanner who looks over special listings of the raw punched data for areas that do not look proper. Since the set of cards for each page is similar, most discrepencies stand out under fairly careful

scanning. The listings are made by a computer program that aids the scanners by marking various high level punctuations so they may verify the card corresponds to the appropriate markings. This program is reasonably fast and the high level punctuation checked is placed in the data so that most errors will disrupt proper markings, giving obvious indication of the error.

Exact Comparisons The program which provides the listings for exact comparisons basically duplicates the master data book so that column for column comparisons can then be made. This takes a lot of computer time, however, to do the necessary formating and print-out, and considerable personnel time to do the comparisons. It is best used at the beginning of implementing this system or training of a keypuncher and for the finding of major errors.

Final Processing The keypunch format was designed so that the keypuncher could most accurately, and with the least necessary duplication, transfer the data onto cards. A number of the characteristics are not conducive to high speed access, however. The use of header cards to avoid duplicating common data is a notable example. The punched format is also incompatable with the high speed file scan method (FSAM) available in the FIDDLE system implemented at the Fresh Water Institute. To further ease processing, verification and correction is done in steps of two to four thousand cards.

The final processing step combines all the identifying data, from the header cards and line cards, and adds an access number to each card that uniquely identifies it. The newly processed data is then merged into the main data file previously produced.

1.5 Introduction to EVQS On-Line Analysis System

Functional Summary This system was written to allow the user to scan the data bank created in previous steps. It allows general purpose scans, both by direct listing of requested questions and by tallies based on specific requests from the terminal user. Once basic information is determined using the tally/list options--more complex translations will generate output files which a statistical analysis system, such as SPSS, may use to perform cross tabulations and in-depth statistical analysis.

The basic approach is one of providing the terminal user with a tool to peruse the complete data base which can reduce the data, under operator control, for input to a statistical program. The functions provided allow the user to manipulate his input-output files; list or tally based on a further division of the previous subset; or translate the responses to a specific question into binary form for tally and later use.

Main Features A number of specific features gives the system its' flexibility. They are listed briefly as follows:

FSAM--A high-speed file scanning access method, which is used to create the subset of the master data file.

Translations--Converting any question from character representation to binary flags--allows tallying, conditional testing, and linking to statistical programs. (These match the translators mentioned in the Introduction to the Environmental Quality Survey.)

Command Handler--A master control program to perform all communications with the terminal--it allows the function programs to

concentrate on their purposes, while affording ease of operation to the user.

Syntax of Commands All commands are of the same basic structure: A command keyword, followed by various keyword parameters. The specific form of the parameters is one of the four.

EXAMPLE:

```
TALLY:    GO,QUES=1Ø,SPECIAL=(IGNORE,SAVE),RANGE=(5-1Ø,16-3Ø)
```

For all keywords, commands and parameters, only the first letters are significant; similiarly for most options. The above example may be condensed:

```
T:  G,Q=1Ø,S=(I,S),R=(5-1Ø,16-3Ø)
```

Command Summary The following table gives a preview of the functions provided.

ASSIGN--manipulate input/output file assignments,

COPY--transfer data between work files and to printer,

IF--request conditional testing by tally routine,

QUERY--produce helpful documentation listing about questionnaires,

SCAN--produce subset of master data file,

TALLY--do tallies/generate listings,

TRANSLATE--request translations.

Table 1.4 Command Summary

NOTE: A special on-line debug package also exists for use by EVQS systems programmers.

PART 2

FINALES

2.1 Results

As of late June, 1973, a year after this project was begun, design has been completed and implementation is well underway. Keypunching commenced with the smaller lakes, Schroon and Saratoga, and has continued up to and excluding 1971 Lake George A7 forms--in total eight boxes remaining. Comments have been skipped in punching, to be saved until more detailed work can be done on handling them. A comment's punch form is herein described, however it was decided more experience with the more structured data was advisable before the comments are tackled.

Verification procedures have been nearly completed. The need for updates to handle major keypunch errors, although rare, has greatly obstructed programming and serious consideration should be made of updating the cards directly in hard copy rather than on disk. This has one major problem; namely, converting all EOP's to one type, i.e. @ to #. This is necessary to reduce analysis scanning time.

The first card box keypunched contains samples of all form types (excluding form E4) for Schroon Lake, and has been used to test the necessary procedures. It is advised that the next data manager start by reprocessing this box, since full error listings have been provided. This will both give the manager familiarity with the data and double check the process.

Present State of Punching Summary Table Experienced keypuncher using two program drum card--600 cards/day or 80 cards/hour.

Man-weeks	Lake	Boxes**	Sequence numbers
1	Schroon	2	1- 3,727
1	Saratoga	2	3,728- 7,376
3	Oneida	6	7,379-18,178
1.5	George -- 1970 form	3	18,179-23,355
1.5	-- A5	3	23,356-28,904
1.5	-- other '71'*	2½	28,905-33,695
9.5 = 2.5 man months		18½	
	Prepunched cards --	5½	33,696-44,000
	No data	24	0-44,000

*1971 Lake George forms are keypunched up to and including G2; A7's haven't been punched and neither are any comments.

**2000 cards per box

Estimates Lake George A7 forms should require about 7½ boxes of cards.

That required for comments is indeterminable offhand.

Table 2.1 State of Punching

The specific bugs are twofold:

1) The OBCORR generated translation; entered into the EVQSDATA PDS; loses sections of data, and the reason is presently unknown. It might be wise to rewrite the program, either in the original assembler or PL/I.

2) The RENUM program should be double-checked to make sure it handles all cases correctly.

The core of the analysis system is complete, as follows:

1) The command handler and syntax parser are fully operational. The one possible expansion in all routines is the ability to handle variable length numeric parameters.

2) The work file handling and ASSIGN command are functional.

3) The COPY command has one main problem; namely, it does not return control to the main program properly and so abends. It also will copy garbage if the files has not been written on and it does not print the record count as requested.

4) The QUERY documentation command is working, but it should be double-checked. The documentation entries must also be entered into the documentation file.

5) The pattern building section of the SCAN command is operational, while the respondent number range handler is not. The operational section causes an error in FSAM, and this should be referred to John Fisher for correction.

6) The major work remains in the TALLY/TRANSLATE/IF routines. The input handler is fully written, but only partially debugged.

a) All the options of TALLY work initially, but the TYFILL routine has not yet been checked.

b) The TRANSLATE work list building routines have been partially checked and do not yet work.

c) The IF work list builder is being checked but is not completed as of this writing, however the special options flags do work.

The operational analysis section has only been desk debugged since both the input section and the SCAN command must be working before test data may be obtained. The logic of this section should be checked to assure that all possible cases are handled; eg., if a card has all NR's at its end, the last EOQ's are not punched, but should be filled in the QTAB; presently they are not. Also the exact place of setting of each flag was not documented and there may be some inconsistencies, or even some missing settings. The only translator written is the CHAR string translator, although others have been designed. The TALLY output routine, written in FORTRAN, is only a sample of what may be done and should be expanded.

The main problem with debugging the EVQS system is inherent in its original goals. It is an on-line system, which at R.P.I. means using ALPHA. This may be fine for users, but not for programmers. Any fatal error abends the system and the programmer must wait until the next day to get his SYSUDUMP back to determine the error. The BEGIN command's PDUMP option is a partial solution to this, where it allows the programmer to dump registers and core while running. Also '@' as the first character of a command line

prevents the GO routine from being executed. These two allow all but fully disasterous errors to be evaluated. The EVQERR routine would be a full solution if it were working, but work on it was dropped in favor of designing the analysis routines. The routine intercepts program exceptions before they abend the system and allows the user to quickly determine the cause, but corrections must still be done in batch. EVQERR, however, allows the user to reinitialize the system and to check out other things with no delay. All but the simplest corrections must be entered through a new compilation, which is another day's loss. Minor errors may be entered, via ZAP, which could also be called from the BEGIN command's LINKS parameter, which has not been checked as yet. Hence, debugging is a two hour per day, for many days, process.

This author's advice is to space out completion tasks so the time required for a dump is accounted for and to concentrate on getting EVQERR and all of BEGIN's options operational.

This will not take very long, but once done will greatly speed debugging. The work time should be two or three hours on the terminal for one day, allowing for two or three days interpreting the dumps and recompiling the corrections. The next day is on the terminal, and so on. This allows optimal use of the terminal without getting hung-up on "instant corrections".

2.2 Conclusions

As can be seen, the EVQS system is a comprehensive approach to a highly complex data source. The ideas have come from many sources; but this author can truthfully say that no other systems were found to be capable of handling such a data base. Even though the system has a while to go before it is complete, or even fully operational as conceived, the core is working and the remaining foundations are designed.

Looking back it is easy to see this would have been more appropriate as a doctoral project in computer science than as a master's project, then full implementation would have been possible by the original designer. While a few more stages of debugging could be accomplished by this author; it was decidedly more advisable to spend the time explaining the concepts to those to take over programming. In this way the documentation, though extensive, could be backed up and corrected by direct discussion.

The one main fault was the necessity of working with a previously developed coding system. Since the data was so vast, it was impossible to recode the data. The code was developed by others, with not enough thought* for computer entry and hence greatly hinders the system. Two specific things stand out:

*While consideration was given to the codes, the logical problems were apparently not evident to those computer people who advised.

a) alphanumeric multiple choice--without this beast, translation and descriptions would be greatly eased, b) intermingled data. If variable and fixed form responses were separated, a linked recored form could have been used, greatly speeding access of the fixed form data. Both could be handled by translating the already punched data. The alphanumeric to numeric conversion would require special programs, but the limited to fixed form translation is inherent in the EVQS system.

The main hinderance has been the R.P.I. "computer facility". While fine for educational computing exercises, handling a data base of this size, on a configuration such as R.P.I.'s, is very difficult. Likewise, debugging a system as complex as the EVQS, on this time-sharing system is very difficult. The problems encountered are overcome mainly by greatly increased programming and handling time, and a sharp rise in the aggrivation factor.

A wiser way to proceed would have been to leave the on-line system for later system programmers, and to concentrate upon getting the data encoded and corrected. A minimal batch program could have been developed to translate the most prevelent forms of data; straight character match, numerics and numeric multiple choice, to a form useable by SPSS. This would have, at least, made the data quickly available.

The failure of these design and planning decisions came from a lack of design management experience and a failure to appreciate the time requirements of debugging such a system as EVQS on an instillation such as R.P.I.'s.

2.3 Discussion of Further Possibilities

This is the section where one's imagination may run wild. Your author will, however, consider mainly those additions that can easily expand the system without major revisions or additional design. Three aspects of the system call for further work as follows:

1) Filling out system capacities by way of completing the remainder of the translators,

2) Easing terminal input requirements by providing automatic reference tables to convert from question number to page and index number for a specified form type. Translations could also be specified automatically; where the stored definition would be fixed form, and thus be much faster than the free form command syntax.

3) Completing output forms by expanding the binary and formated list options so multi-card records could be produced containing all types of data, binary translated, fixed columnular, and free form. This expanded list option will allow effective use of advanced statistical analysis packages, such as SPSS and BMD.

Two more advanced possibilities could also be considered, but careful trade-offs must be made as to how inclusive the EVQS system is to be. These are pattern recognition actions: the first is to recognize the most complex data forms, while the second is to recognize patterns within the data base. Both are extensive proposals individually and this author remains of the opinion that

these actions, in their complete form, best be left to special external systems. The previously described expanded list option will allow sufficient linkage to advanced analysis routines, providing procedures are developed to link the available systems in an on-line form. A wise action would be to add to EVQS the ability to handle its own special formats (SPF). This could be accomplished in two ways:

1) design translators for each form, and

2) design a general format parser.

Perhaps the same principles used to design the command syntax parser could be used.

Some of the designs mentioned have been included at the ends of the Implementation and Debugging Notes Section.