QMARG

A Program for Generating Marginal Counts and Distributions

User's Guide

Revision: 21.1.5

QMARG is the data marginal count component of the QUIP System

QUIP System software is available from:

www.quipsoftware.com

QMARG documentation by

Jan Werner Data Processing

TABLE OF CONTENTS

INTRODUCTION

The QUIP System is a software package that allows almost any data set to be tabulated and the results presented as fully annotated tables. It includes programs to handle data manipulation and conversions, validity checking, marginal counts and weighting, as well as cross-tabulation.

The QUIP System was designed to meet the needs of data processing professionals. Programs are script-driven using a common structure. This allows for batch processing or for users to create their own custom environments, using QUIP System programs as processing engines.

The programs in the QUIP System share a common specification language that allows users to describe extremely complex combinations of logical conditions and numeric expressions. This language is compiled in memory at run-time, so all programs run very quickly on large data files.

This guide provides an introduction to QMARG, a component of the QUIP System that provides users with a quick way to summarize the contents of each position in a fixed field data file and to get frequency distributions on the contents of both numeric and character fields. QMARG also provides advanced features such as weighting and selective sub-basing.

The following conventions are used throughout this document:

Angle brackets:	<...>	A mandatory item to be specified.
Square brackets:	[...]	An optional item to be specified.
Vertical bar:	..|..	Only one of the listed items should be specified.

QMARG keywords are in uppercase and the minimum abbreviation for each in bold uppercase, lowercase letters are used where a number should be entered (e.g., **WID**TH nnn).

1 FLAT FILES AND MARGINALS

Marginal counts allow a user to get a quick overview of the contents of a flat file by position, for data validation, topline information, or to verify more analytical results such as cross-tabulations.

1.1 Flat Files and Data Formats

Flat files, also called fixed field files, consist of multiple records in which each item of data is located at a constant position within the record (as opposed to delimited files, where items are located by counting the number of delimiters from the beginning of the record). A flat file can be thought of as a table in which each record is a row and data fields are organized in columns. Relational database systems store data internally as a set of flat files along with a dictionary and rules that describe the contents of each field and the relationships between tables. Since the contents of almost any database can be exported to, or imported from them, flat files are the lingua franca of data processing.

Each field in a flat file must be wide enough to store the largest item that can occur. To keep the overall size of the file as small as possible, data that can take on a limited number of values is often coded. For example, the gender of a person might be coded as "1" or "M" for male, and "2" or "F" for female, using a single position, rather than reserving 6 positions for each entry.

When a variable can take on multiple values for a single record, such as a list of items which may be selected in any combination, each value becomes a boolean (e.g., yes/no) variable which must usually be dummy coded. Data stored as characters or numeric values requires one or more bytes for each such item, but a substantial gain in storage efficiency is obtained by setting individual bits on or off to represent the boolean items. The most widely used standard

for storing bit-mapped data is called column-binary and uses the image of the Hollerith punch card that was widely used in the early days of computers and data processing.

In column-binary, data are stored in blocks of 80 columns of 12 "punches" consisting of the ten numeric digits 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 plus two codes traditionally labeled "X" and "Y" that are represented in ASCII by the characters "-" and "&." The card image is easy to understand and to visualize, so character data is often stored in a similar configuration, particularly in fields that use coded values extensively, such as survey research, although multiple "punches" cannot be used within a single ASCII "column."

1.2 Marginal Counts

Marginal counts, also called hole counts, are simple tables that display all of the codes that appear in given locations of a data file. Because they are easy to run and to read, marginals are useful for obtaining topline information, verifying data validity and checking the contents of more elaborate cross-tabulations or reports.

QMARG marginals count "punch" codes in selected columns of column-binary and character data and can also count all possible character codes in character data files. Many options are provided for percentaging, annotating, filtering or weighting the output.

1.3 Frequency Distributions and Character Arrays

Some data types, such as names or numbers other than single-digit integers, cannot be easily read from marginal counts, QMARG also provides the user with the ability to show frequency distributions of the contents of multi-column numeric fields in both character and column-binary data, along with such simple statistics as mean, standard error and median. In character data files, arrays can be displayed listing the frequency of all entries appearing in multi-column character fields.

2 THE SPEC FILE

QMARG instructions are called specifications, or "specs" for short. QMARG reads the specs from an ASCII text file called a spec file. The spec file contains all the information the program needs to process the data and produce the desired output.

Specs must be written using a text editor that can write files that do not contain any embedded control codes or tab characters. Case is ignored for specs and keywords, but is preserved in annotation, tags, dictionary definitions and literal strings within quotes.

Spaces are used as separators between keywords, but the number of spaces does not matter. Tab characters are not allowed in specs and are treated as non-blank characters in label annotation. Position on the text line is significant in base variable definitions, where a specific column is used to separate the filter specifications from title text.

Any line that begins with an asterisk (*) in column 1 is treated as a comment. It will print in the spec listing but otherwise be ignored by the program, as will any completely blank line.

QMARG compiles the spec file in a single pass so forward references are not allowed. Dictionary entries must be defined before they can be referenced elsewhere in the spec file.

After a spec file has been written, QMARG may be run from a command prompt by entering:

```
QMARG <filename> [command line directives|parameters]
```

QMARG will read the spec file named on the command line and compile it, checking for syntax errors and writing a listing file back to disk as it proceeds. If no fatal errors are found, QMARG will read the input data file(s), tabulate the marginal counts and distributions as specified, then write out the report file and optional table of contents file.

2.1 Elements of the Spec File

Every QMARG spec file must contain three basic sets of specifications, and one optional set, which should appear in the following order in the file:

DATA DIRECTIVES:	Instructions that describe the input (data) and output (listing, tables and contents) files.
GLOBAL LEVEL OPTIONS:	Instructions and parameters that apply to an entire QMARG run.
[BASE VARIABLES, etc.]:	Optional instructions that specify filters to be applied to entire tables or specific columns of tables. Numbersets and weights are also placed in this section.
TABLE DEFINITIONS:	Instructions that tell QMARG which variables to cross-tabulate by each other.

Dictionaries may appear anywhere before the table definitions section.

2.2 Keywords and Statements

Most options in QMARG are specified using keywords separated by spaces. Many keywords accept options which follow after one or more spaces. Other keywords, such as those used in the specifications area of variable definitions, are followed immediately by their parameters. When multiple options or parameters are used, they are separated by either commas(,) or semi-colons(;) depending on the keyword, and with no spaces allowed in the option list.

A statement is a line of text beginning with a keyword in column one and often followed by a name and/or various options.

An END statement begins with the keyword END in column one and contains no other text.

The sections and elements of a QMARG spec file begin with a statement. Some may be entirely defined in a single statement line while others may contain many lines of specs, terminated by an END statement. When the end of one element is clearly defined by the statement beginning the next element, an implicit END statement is generated, and the physical END statement may be omitted, as, for example, between variables and numbersets.

An explicit END statement must always be used immediately before the table definitions begin, to end a dictionary, and before and after any section controlled by #INCLUDE or STOPSPEC.

3 DATA DIRECTIVES

Data directives identify the input data files and the text output files created by QMARG. Each data directive begins with a keyword followed by the file name.

3.1 File Names

QMARG accepts the following data directives:

INTAP[n]	<[path\]filename.ext (lrecl> [V]	Input data file.
LISTING	<[path\]filename.ext>	Job spec listing file.

REPORT	<[path\]filename.ext>	Output tables file.
CONTENTS	<[path\]filename.ext>	Table of contents file.

If no LISTING file is specified, QMARG will create one in the current directory, generating a unique name for it. All other output files are created only if that data directive is present in the spec file. If no explicit path is given, QMARG always reads from and writes to the current directory. QMARG accepts any valid DOS or Windows long file names with no embedded spaces.

INTAP names an input data file followed by a left parenthesis and the record length in bytes. Record length is assumed fixed unless specified as variable, meaning that each consecutive block of bytes of the specified record length is considered a separate record and QMARG checks that the file size is a multiple of the record length, issuing a warning if it is not.

The letter "V" after the record length indicates variable length records, each of which must be terminated by an end-of-line marker (Carriage Return-Line Feed). For variable length records, the specified record length must be equal or greater to that of the longest record in the file, not counting the 2 bytes for the CR-LF.

A QMARG run may have any number of INTAP files (usually numbered INTAP1, INTAP2, etc.). Data files are processed in the order they appear in the spec file, regardless of the numbering used for the INTAP keywords. All files should have the same record length specified.

3.2 Command Line Data Directives

Data directives may optionally be specified on the command line using the following switches:

/I[n]= <[path\]filename.ext>	INTAP file name
/IL[n]= <nnn>	INTAP record length
/L=<[path\]filename.ext>	LISTING file name
/S[n]=<[path\]filename.ext>	SPEC file name
/T=<[path\]filename.ext>	REPORT (tables) file name
/TC=<[path\]filename.ext>	CONTENTS file name

Multiple data directives are separated by spaces (e.g., /X=xx /Y=yy /Z=zz). The total length of the command line (including program invocation) may not exceed 127 characters. Up to 4 spec files may be listed on the command line; they are read in numeric sequence and treated as a single spec file. Data directives contained in the spec files always override those provided on the command line.

Up to 4 INTAP files may be specified on the command line and each must have a matching record length (/ILn=). Variable length record files cannot be specified in command line data directives.

4 GLOBAL OPTIONS

Global options apply to an entire QMARG run. Most are specified on a RUNOPS line which should appear immediately after the data directives at the beginning of the spec file. Most run options will default to a specific setting unless they are explicitly defined otherwise.

4.1 Address Mode

QMARG requires a data address mode to be defined before any specifications can be processed.

The SETUP ADDRMODE statement determines whether specs refer to 1-byte ASCII characters or the 2-byte column-binary format used mostly in marketing research. It also specifies whether the data are addressed using card/column or direct offset notation.

The concept of the address mode is fundamental to understanding how QMARG processes data: The address mode determines how the program interprets the specification of data locations and their contents in the remainder of the spec file. For more on how ADDRMODE affects data addressing, see section 5.1.1.

To set the address mode, use one of the following statements at the beginning of the spec file:

SETUP ADDRMODE B	Column-binary data addressed by card and column.
SETUP ADDRMODE C	Character data addressed by position in record.
SETUP ADDRMODE D	Character data addressed by card and column.
SETUP ADDRMODE E	Column-binary data addressed by position in record.

If address mode is not explicitly stated, QMARG defaults to ADDRMODE B (column-binary data, card/column addressing).

4.2 The RUNOPS Line

The RUNOPS line begins with the keyword RUNOPS. Run options modify program defaults and are invoked by keywords separated by spaces, some of which are followed by one or more parameters. The RUNOPS line options are listed below with the applicable parameters.

ERROR ALL|nn	Set level for warning or error messages to prevent QMARG from running after compilation. ALL forces an end if any messages were generated. Level 4 allows warning messages.
LENGTH nnn	Maximum page length nnn lines. Default is 63.
NAME C|R|L|A	Print table name Center/Right/Left/Alternate sides of each page. Default is Left.
NOCONTINUED	Do not print 'continued' under the table name on continuation pages if a table overflows the first page.
NOCHECK	Do not check for spec references beyond the input data file record length (note that this can cause the program to crash at run time if locations outside the data area are actually accessed).
NOMULT	Do not print the 'multiples' line at the bottom of tables in column-binary address modes.
NOWTMSG	Do not show individual warning messages for each case with a record level weight that evaluates to zero (see section 5.3).
PAGENO R|L|A[,nnn]	Number pages at the top Right/Left/Alternate side, starting optionally with page nnn. Default is Right, starting at 1.
PCELLS	Modify default punctuation for all tables (see section 7.3.1).
STAR 'x'	Put the character 'x' in non-zero cells rounding to zero (default '0').
TEST SPECS|nnn|ALL	Stop after testing specs/Run only on first nnn cases/all cases. Default is run ALL cases.
ZCELLS **'x'**	Put the character 'x' zero cells (default '-').

PCELLS, STAR and ZCELLS parameters set in RUNOPS may be overridden in individual table definitions (see section 7.3). NOCONTINUED and NOMULT do not take any parameters and thus cannot be overridden for individual tables when set in RUNOPS.

4.3 Run Titles

Run titles, a block of text that will appear at the top and/or bottom of every page of output, may be specified by a RUNTITLES statement, followed by the desired text, and terminated by an END statement. The format is:

```
RUNTITLES   TOP|BOT   R|L|C
Text goes here
(as many lines as needed)
END
```

The parameters indicate whether the text is to be printed at the top or bottom, and at the right, left or center of each page. Two separate RUNTITLES sections may be specified in a spec file: one for the top, and one for the bottom of each page.

4.4 Summary Row Labels

QMARG automatically computes and displays three rows of summary counts at the top of each column in all tables of marginals: total cases, number answering and number not answering.

For column binary data, a fourth row showing total mentions (total number of punches) is also displayed at the top of the table and a summary row showing the number of multiple punches (cases with more than one punch in this position) appears at the bottom, unless suppressed by the NOMULT run option.

For character data reported as punch values, a summary row showing the number of other answers (characters that do not correspond to the 12 punch values 1-9, 0, - or &) is displayed.

QMARG assigns default labels to these summary rows, but alternate labels may be specified for rows only in a LABELTEXT section.

The summary row names and their default labels are:

	Row Name	Column-binary	Character
Total	TT	TOTAL	TOTAL
No Answer	NA	NO ANSWER	NA
Total Answering	TA	TOTAL ANS	TA
Total Mentions/ Other Answers	TM	TOTAL MENT	OTHER ANS
Multiple punches	SM	MULTIPLES	

The section starts with a LABELTEXT statement and is terminated by an END statement. Definitions begin with the vector name in cols. 1-2 and text starts in col. 5. Continuations on a 2nd line are indicated by the letter C in column 4. Up to 12 characters are allowed for each of the two lines of text . The following example shows the use of LABELTEXT:

```
LABELTEXT
TT  Total Sample
NA  No Answer
TA  Total
TA CAnswering
END
```

5 THE QUIP SPECIFICATION LANGUAGE

The QUIP specification language was designed to allow nearly any condition to be described in nearly any kind of data. QMARG reads data sequentially, applying the instructions in the spec file to each record and counting the cases that meet each condition specified.

5.1 Data Specifications

This section describes the basic elements of the QUIP specification language as they are used to tell QMARG what to count in the data. Section 5.2 describes how to use these elements to specify test conditions in QMARG base variable specifications.

5.1.1 Addressing Data

QMARG addresses data using absolute column locations within each record. The location may be specified directly as the number of columns offset from the beginning of each record or using the card/column notation favored by many statisticians and marketing researchers. Data may also be defined in character mode (e.g., ASCII), where each column occupies a single byte, or in column-binary mode, where each column occupies two adjacent bytes.

QMARG provides four addressing modes to handle the possible combinations (see section 4.1).

In address modes C and D, a column occupies a single byte, while in address modes B and E, a column occupies two bytes. Address modes C and E use direct offset notation, so in address mode C, column x is byte x of the record, while in address mode E, column x consists of the two consecutive bytes 2x-1 and 2x.

In card/column notation, the column number within the card is a two-digit number ranging from 01 to 80 and must be preceded by a card number. The first 80 columns of a record are written as 101 through 180, the next 80 columns as 201 through 280, and so on. The data in columns 180 and 201 would be addressed as columns 80 and 81 in direct offset notation.

QMARG interprets data locations according to the current address mode automatically. In marginals, the address mode also determines whether column-binary punches or character representations of punches are tallied. In frequency distributions and variable specifications, any data type may be specified explicitly regardless of the address mode. Even-numbered bytes cannot be addressed directly in column-binary address modes since all addresses correspond to 2-byte column locations.

5.1.2 Fields and Numbers

A data address followed by one of the codes listed below will be treated as a field or a number. QMARG recognizes the following field or numeric codes (either upper or lower case):

An Column number or field, 'n' digits or columns long, where column depends on address mode. This translates to Bn if the address mode is B or E, and to Cn if the address mode is C or D.

Cn Character number or field, 'n' digits or characters long

Bn Column-binary number or field, 'n' digits or columns long

Xn Extended column-binary number, 'n' digits long, where an 'X' punch repeated 'n' times represents 10 to the power n (e.g. for X2, 'XX' = 100).

H Short, signed integer (sometimes called a 'halfword')

I Short, unsigned integer

F Long, signed integer (sometimes called a 'fullword')

G Long, unsigned integer

Q Single-byte, signed integer (sometimes called a 'quarterword')

R Single-byte, unsigned integer

E Single-precision floating point (occupies 4 bytes)

D Double-precision floating point (occupies 8 bytes)

Pn Packed decimal number 'n' digits long; 'n' is an odd number between 1 and 15

Fields (codes A, B, C, X) may be from 1 to 15 characters long for numbers or field operations. Column-binary fields (A, C) may be up to 80 columns long and character fields (A, C) may be up to 250 bytes long when testing for their being blank or non-blank, or 24 bytes long when used in a character array table (section 7.2.3).

Leading blanks are allowed in numeric fields, embedded blanks and non-numeric characters are not, except for a leading minus sign to indicate a negative value. To handle situations when numeric fields may contain non-numeric values, codes A, B, C and X may be followed by a suffix indicating how to handle special conditions: "B" indicates that blanks are to be given the value zero, and "E" indicates that any non-numeric value be given the value zero (e.g., 101be2 represents a 2 column column-binary field in which any non-numeric entry would translate to 0).

Decimal points are interpreted as expected in numeric fields so that in a field defined as 001c4, the entries "0002" and 02.0" will evaluate to the same numeric value in distributions and tests, although the marginal counts would be not be the same for those locations.

Numeric fields of data type A may have an additional suffix "X" that works like the data type X (extended column-binary) for either column-binary of ASCII data. In ASCII data, an X-punches are represented in data by minus signs is used in data for X-punches, so for a field defined as 21AEX3, an entry of "---" would translate to 1000, "--1" to zero, and "-10" to minus 10.

5.2 Testing Conditions

QMARG reads data sequentially, one record at a time, into a work area and applies the specs to the contents. A case is counted by QMARG when specified conditions test true. Field and

numeric tests require an operator. Punch tests apply to single columns and no operator is used. Complex conditions may be created by using Boolean operators to combine simple tests.

5.2.1 Columns and Punches

An address with no field code identifies a single column that contains "punch" data and may contain only the values that correspond to the punches in a column of a Hollerith (or IBM) card. The punches are 1 through 9, 0, X and Y, in that order. In ASCII modes, the X punch refers to a minus sign (-) and the Y punch to an ampersand (&) in the actual data.

In column-binary data, a column may contain multiple punches in any combination. This allows as many as 12 different codes to be stored in a single 2-byte column in column-binary data, as compared to the 12 bytes that would be required to store the same information as ASCII. For this reason, column-binary is often used for data sets containing many multiple response items.

5.2.2 Literals, Constants and Numeric Expressions

Literals are constant values used in comparisons and usually follow a logical operator. They are indicated in specs by single quotes. QMARG treats literals differently depending on the context in which they appear. In column tests, they are punch lists; in numeric tests, literals are numeric values and in field comparisons they are character strings. In some situations, multiple literals may be specified within a pair of single quotes (see section 5.2).

Constants are values used in numeric expressions and are indicated by prefixing a number by the letter "K", thus K132 represents the constant value 132.

Numeric expressions are built from numeric fields and constants using the four standard arithmetic operators: **+**, **-**, ***** and **/** for addition, subtraction, multiplication and division. Parentheses **( )** may be used to provide precedence among operators (up to 15 levels).

QMARG converts all numeric values internally to double-precision for evaluation or computation, so spec writers do not need to be concerned with mixing data types or multiplying values to preserve their precision.

5.2.3 Field and Numeric Tests

The following operators test a field for the conditions indicated:

B	Test field for being Blank (blank means that the field contains all spaces for ASCII data, all nulls for column-binary data).
P	Test field for being Packable (the field contains a valid numeric value).
Q	Test the field for being either Packable or Blank.
U	Test the field for being Unpackable (the field is not a valid numeric value).

The following operators test the numeric value of a field against one or more number literals:

='nn[;mm]'	Equal to a numeric value. A maximum of 4 numeric values may be specified for the literal, separated by semi-colons.
G'nn'	Greater than or equal to a numeric value.
L'nn'	Less than or equal to a numeric value.
>'nn'	Greater than a numeric value.
<'nn'	Less than a numeric value.
R'nn;mm'	Falls within a numeric Range (from;to).

The following operator compares the contents of an ASCII field to one or more string literals:

@[-n]'mm'	Character string compare [optionally for "n" contiguous fields] (used with ASCII data only). Accepts up to 4 literals separated by semi-colons, all of which must be of the same length as the field being compared.

The following operators test the value of an ASCII field against the ASCII string value of literals (the relative value of an ASCII string is determined by its sort sequence):

@**G**'aa'	Greater than or equal to an ASCII string value.
@**L**'aa'	Less than or equal to an ASCII string value.
@**>**'aa'	Greater than an ASCII string value.
@**<**'aa'	Less than an ASCII string value.
@**R**'aa;bb'	Falls within a Range of ASCII string values (from; to).

The sense of all field and numeric tests may be reversed (negated) by the "Not" operator **N**. For field and numeric tests, the N is placed immediately in front of the operator. For the field compare test, the N is placed immediately before the literal, outside the quote. For string value tests, the N is placed between the @ sign and the comparison operator.

5.2.4 Punch Tests

Punch tests are specified by a column address followed immediately by a literal containing a list of punches to be tested for. The list of punches is called a punch mask and the test is satisfied if any of the punches in it are present in the column. A range of consecutive punches may be specified using a hyphen (-), so '1-Y' is equivalent to '1234567890XY' and '1-39-X' is the same as '12390X'. Note the order of punches in the mask (see section 5.2.1). Punch tests may test for the absence of punches by prefixing either the literal or the punch mask itself with an N.

For column-binary data only, a column may be tested for the number of punches it contains by using the tally operator:

T<op><n>'pp'	Where <op> is a numeric comparison operator (=,<,>), n is a number from 0 to 12 and 'pp' is a list of the punches to be counted in the tally.

5.2.5 Boolean Operators and Compound Tests

The following Boolean operators are used to combine conditions into compound tests:

&	Logical AND: The result is true if, and only if, both conditions are true. (Ampersand)
!	Logical OR: The result is true if either condition, or both, are true. (Exclamation point)
( )	Parentheses: The entire expression within a matching pair is treated as a single condition or value in a Boolean or arithmetic expression.
{ }	Braces: Allows an arithmetic expression to be used in a test as if it were a single numeric field in a test condition.

The maximum length of a QMARG specification is 500 bytes, and expressions may be nested in parentheses up to 15 levels deep, so very complicated conditions may be specified. There is no Boolean "not" operator, so compound expressions cannot be negated as a whole.

Any numeric field or arithmetic expression that is not generate a valid number will cause the entire specification in which it appears to immediately evaluate as false.

5.2.6 Multiple Test Conditions

Two shorthand methods may be used to specify multiple test conditions on the contents of a single data location. These generate a separate base condition for each test and thus should only be used in variables to be used as spreads in marginals (see sections 6.3 and 7.3.2):

, (comma)	Indicates a new punch test condition in the same data location.
: (colon)	Generates a separate punch test condition for each of a series of consecutive punches in a column-binary or character data location.

5.2.7 Examples of Test Conditions

The following examples illustrate data tests using the QUIP specification language. Note that tests on single column fields and punch tests can almost always be used interchangeably, but the punch tests are often easier to specify and are more readable:

11'125'	Test for punches 1, 2 or 5 in column 11.
11C1R'1;2'!11C1='5'	Same result, using value tests.
11C1R'1;5'&11N'3;4'	Same result, different logic.
21C5='125'	Test a 5 byte field for the numeric value 125 (the test will be true for 125, 0125, 00125).
21C5@'00125'	Test a 5 byte field for the string "00125" (the test will be false for " 125" or " 0125").
21C3B	Test for a 3-column field being blank.
21C3P	Test for a valid number in a 3-column field.
101'1-8'	Test for the presence of punches 1 through 8 in column (any combination will satisfy this in column-binary data).
101N'1-8'	Test for the absence of punches 1 through 8 in column (any punches other than 1-8 may be present).

101'N1-8'	Same as the preceding.
101T=1'1-8'	Test for the presence of exactly 1 punch in the range 1-8 in column (column-binary data only).
101T>2'1-8'	Test for the presence of more than 2 punches in the range 1-8 in column (column-binary data only).
{21C3+24C3}='100'	Test that the sum of the values in the 3-column fields beginning in 21 and 24 is equal to 100.
{21CB3+24CB3}='100'	Same as above, but set the value to a blank field to zero so that it does not cause the entire specification to fail.
21C3@-3'100'	Test for the character string "100" beginning in columns 21, 24 or 27.
21C1@R'A;Z'	Test for any uppercase letter in a 1 character field.
121X2='100'	Test for a value of 100 in a 2 column field in column-binary data (represented by an X punch in both columns).
121AX2='100'	Same as above, but for either column-binary or ASCII data, depending on the address mode.
ALL	Count all cases in the input file. This dummy condition cannot be combined with any other specifications.
101'1:3,4-6,78'	Generates five separate tests in location 101 as follows: punches 1, 2 and 3 each generate a test, punches 4-6 are tested as a group, punches 7 and 8 are tested as a group.
101C2='1,10,20'	Generates three separate numeric tests, one for each of the values specified in the literal.

5.3 Weights

A weight is a numeric value assigned to each record in a data set that is used as a multiplier whenever the record is counted and is used for such purposes as to adjust counts to known population characteristics or to tabulate usage rather than cases.

QMARG allows for weights that apply at the table level, specified by a WEIGHT statement and applied by a table definition option (see section 7.3). The statement begins with the keyword WEIGHT or WT, followed by a name of up to 8 characters and a valid numeric field, constant or numeric expression (see section 5.2.2):

WEIGHT <name> <expression>

When the value specified consists of a numeric field in the data, this value can be adjusted to account for implicit decimal places by adding a period and the number of decimals by which the field value should be adjusted. This notation (which can also be accomplished by dividing the value by a constant of ten to the desired power) is available only in weight statements.

QMARG will stop evaluating an expression if any part does not generate a valid numeric value in the data. In a weight definition, this will cause the value to be set to zero. Each weight that evaluates to zero generates a warning message on screen and in the spec listing every time that weight is applied to a table, unless the NOWTMSG run option is specified (see section 4.2), and a count of zero weight messages is included in the run statistics at the end of the listing.

Some examples of weight definitions:

WEIGHT adults 171CE3.2	The weight is the value found in the 3 character field beginning in 171 read as having 2 implied decimal places, and with invalid values set to zero
WT wght 171ce3/k100	Defines the same weight as above
WT K k20	The weight is a constant value of 20

5.4 Numbersets

In the QUIP specification language, a numberset is a list of constant numeric values.

A numberset is identified by the keyword NS in column one, a name up to eight characters long, and a series of numbers separated by blanks. The letter "X" may be used instead of a number to indicate that a row is to be excluded from computations in statistical subtotals. A backslash (\) may be used to indicate that the list of numbers continues on the following line.

Numbersets are used in QMARG to assign weight values for means on marginal tables counting punch codes. Each numberset should have exactly 12 entries corresponding to the 12 punches in the sequence 1-9, 0, X, Y.

Here are some examples of QMARG numbersets:

NS Scale1 1 2 3 4 5 6 7 8 9 0 10 x

NS Scale2 -10 -5 -3.33 -1 0 1 3.33 5 10 X X X

6 BASE VARIABLE DEFINITIONS

A variable in QMARG is a set of instructions that defines filters to be applied to the counts in a table definition and provides table title text labels. QTAB users will recognize QMARG variables as BASE class variables.

Every QMARG variable begins with a VAR statement giving it a unique name and providing basic information about class of variable and text options. All spec lines following the VAR statement, up to the next END, VAR, NS or WT statement, are used to define the cell contents and title information for that variable.

Following the VAR statement, each line in the body of a variable specification is divided into two distinct parts: the left-hand side is used to specify cell contents (specs), while the right-hand side may be used for title text that will appear at the top of any tables to which the variable is applied, immediately below any title text from the table definition. The two parts of each spec line are separated by the "spacer column" in the Label parameters of the VAR statement.

6.1 The VAR statement

The VAR statement begins with the keyword VAR followed by name up to 8 characters long, the LABEL keyword and label options which identify the class of variable and spacer column. Since QMARG only uses the BASE class of variable, the format is:

VAR <name> **L**ABEL **B**[<ww>]|<Equate>[,equate,...]

The variable name may use any printable ASCII characters. Case is ignored. Both the variable class (which is always B in QMARG) and the spacer column must be specified, either explicitly or implicitly through an E type equate to another variable as described below.

6.1.1 Variable Equates

LABEL options may include "equate" codes telling QMARG that specs or titles are not defined in this variable, but should be copied from some other variable previously defined in the spec file. Each equate code must be followed by the name of a valid variable with no spaces in between. Multiple elements may be equated, separated by commas.

The equate codes are:

E<var>	Copy label information from the named variable to this variable, not including title or footnote text.
T<var>	Copy title text from the named variable to this variable.
F<var>	Copy footnote text from the named variable to this variable.
Z<var>	Copy both title and footnote text from the named variable to this variable.
Vnn;<var>	Vector offset: copy the vector definitions from the named from the named variable to this variable, offsetting all specifications by nn data columns.

6.1.2 Examples of VAR statements

The format of a simple VAR statement is illustrated by the following example:

`VAR women LABEL B20`	A base variable named "women", with title text starting in column 21 of each line.

The following examples illustrate the uses of variable equates:

`VAR c1 L V2;a1,Eb2`	A variable named "c1" of the same class as a variable named "b2", previously defined. The specs are copied from variable "a1", also previously defined, incrementing all data addresses by 2 columns from those specified in "a1".
`VAR c2 L B30,V-10;a1,Ta1`	A variable named c2 with title text starting in column 31, with specs copied from a previously defined variable named "a1" at a negative offset of 10 columns, and with titles also obtained from "a1".

6.2 Base Annotation

QMARG treats all text to the right of the spacer column of a variable as annotation. Codes in the spacer column tell what the annotation applies to.

A spec line in a variable definition with no annotation in the spacer column or beyond is ignored for labeling purposes. To generate a blank line, there must be a code in the spacer column or a non-printing character (such as a tab character) in the text area.

6.2.1 Titles and footnotes

A "T" or a "Q" in the spacer column tells QMARG that all text to the right of the spacer column represents a title and will print at the top of the table, above the header annotation. An "F" in the spacer column tells QMARG that text to the right of the spacer column is a footnote and will print after the last row of the table.

Title and footnote lines may be up to 132 positions wide and print in the order they appear in the variable definition..

A variable may have up to 10 lines of titles and 10 lines of footnotes.

6.2.2 Summary Row Annotation

Base variables may specify annotation for the table summary rows to be printed instead of the default or LABELTEXT labels (see section 4.4). These are indicated by an asterisk (*) in the spacer column followed by a letter code identifying the applicable summary row. These are technically titles, so multiple lines are used to print multiline labels. The same rules and limits apply as described in section 4.4 above

***T** TT row - Total
***N** NA row - No Answer
***A** NA row - Total Answering
***M** TM row - Total Mentions/Other Answers

QMARG will only apply summary row annotation when a variable is used as a base. When a variable is used as a spread (see section 0), any summary row annotation supplied through a variable is ignored.

6.3 Base Variable Specifications

A variable must specify at least one test condition that will be used to filter data either at the table or at the column level. Section 5.2 describes how to define test conditions. Section 7.3.2 describes how to use base variables in table definitions.

A variable used as a table base should have only a single condition specified and can be used in all table definitions. If more than one condition is specified in a variable used as a base, the filter applied will be cases that meet any one or more of the conditions specified.

A variable used as a spread should have as many conditions specified as there are columns specified in the table definitions to which it will be applied.

The area to the left of the spacer column in each line of a variable definition is reserved for the test condition specifications. If a specification is too long to fit on a single line, a backslash (\) may be used to indicate that it continues on the following line. Specifications must always start in column one of lines, including continuation lines.

7 TABLE DEFINITIONS

The final section of a QMARG spec file, after variables and other elements have been defined, contains the table definitions that describing the desired tables and how to format the output. The beginning of the table definition section is indicated by a TABLE statement (a line with the keyword TABLE starting in column 1 and no other text) following an END statement. The table definition section ends with another END statement or the end of the spec file.

At the minimum, a table name and a column list or field identifier are required to define a table. One or two bases and, for marginals, a spread variable may also be specified. Formatting and statistical options are specified by keywords on the table definition line and table titles may be provided immediately following each table definition.

7.1 The Table Definition Line

Each table definition line must begin in column one with a table name up to eight characters long, followed by either a list of columns to indicate marginal counts or the field identifier for a frequency distribution or a character array, and optional keywords calling for bases, formatting or statistics. If the line is too long to read easily, a backslash may be used to continue on the next physical line, which will be treated as a continuation of the first line. There may be any number of continuation lines, and text in continuation lines may start in any column.

Table names do not have to be unique, but consecutive tables cannot have the same name, since the second table line would be interpreted as title text for the first table defined. If a table name is longer than eight characters, those after the eighth are dropped by QMARG, but other specifications must still follow at least one blank space after the name. Specifications and options may appear in any order on the line, but parameters must follow the keywords to which they apply and for marginals defined by a list of locations, the order in which they appear on the line determines the order in which they will print on the page in tables.

Up to 10 lines of title text may be entered immediately after the table definition line. A title line must start in column one with the same name as the table definition line, followed by one or more blanks. All text from the first non-blank character after the table name until the end of the line will be used as title text. Table titles are centered on the page and print above titles from base variables.

7.2 Table Types

QMARG provides three types of tables: marginals, frequency distributions and character arrays. Only one type of table can be specified in a single table definition line.

7.2.1 Marginals

Marginals are simple tables that count the punches or character codes in a set of data locations. A table of marginals is specified either by listing up to ten individual data locations separated by one or more blanks, or by specifying the start and end of a set of consecutive data locations joined by a colon (:) or the letter "T." Each QMARG table displays up to ten data locations, so the second method will actually generate as many tables as necessary to display the entire set of consecutive data locations. A third method, useful only with spreads (see section 7.3.2), repeats a single location up to 10 times by specifying the location and the number of repetitions joined by an asterisk (*). The different methods cannot be mixed on a single table line.

In output, data locations are identified as "columns" in column-binary address modes (B and E) and as "bytes" in character address modes (C and D) (see section 4.1). By default, marginals tabulate only the standard punch codes 1-9, 0, X and Y (see section 5.2.1) in all address modes. In column-binary modes, the two high-order bits of both bytes of a column are ignored, and in character modes, any characters other than those for the punch codes will be counted in the "Other Answers" summary row (see section 4.4).

For marginals counting punches, the keyword **MEAN** tells QMARG to compute the mean of the weighted values of the punches and display it at the bottom of each table. By default, punches 1-9 and 0 received their numeric values, X is assigned the value 10 and Y is excluded from the computation of the mean. If different values are desired as weights for the mean computation, the MEAN keyword accepts a numberset name as an optional parameter to assign the values from that numberset to the 12 punches instead of the defaults.

In address modes C and D, the keyword **CHAR** may be used to tabulate the actual character codes instead of punches. This tells QMARG to tabulate all possible character codes in each byte and list them in the output in ascending numeric sequence, identified by the value in hexadecimal notation and where applicable the standard ASCII character. Rows for codes that do not appear in any byte listed on a page are suppressed from the output. Means cannot be computed on marginal tables using the CHAR option.

The address mode only tells QMARG how data locations are specified, so marginals using address modes C and D and the CHAR option can be used with any fixed record length data file to examine its contents by position. This feature can be used for such purposes as checking text files for the presence of invalid characters that cannot be displayed on screen.

7.2.2 Frequency Distributions

A frequency distribution counts the unique values appearing in a numeric field and displays them in ascending numeric order, along with their frequency in the data, the percentage of the total cases and the cumulative percentage. A frequency distribution is called for by specifying a numeric field prefixed by a semi-colon (;) and followed, after one or more spaces, by the maximum number of unique values expected.

As with weight definitions (see section 5.3), a single-digit integer appended to the numeric field after a period indicates an implied number of decimal places and divides the value by that factor of ten before counting it. Decimal places explicitly defined in the data by a period are handled automatically. Values are displayed in the table rounded to whole numbers by default, but one to three decimal places can be shown using the PCELLS keyword (see section 7.3.1 below).

QMARG can display up to 6,000 cells in a frequency distribution table, but enough memory must be reserved in the table definition to hold as many unique values as appear in the data. This memory is allocated whether or not it is used in the tabulation phase, so the maximum expected number should not be set arbitrarily high. For example, a 2-digit numeric field can only contain positive numbers in the range of 00 to 99, or 00 to 100 if an extended number type is used (see section 5.1.2), so there is no need to allocate memory for more than 101 unique values in most frequency distributions defined on a two-digit numeric field.

Cells are assigned in the order that new values occur in the data. If data contains more unique values than cells were reserved in the table definition, the new values encountered after all cells were filled are discarded and those records counted as invalids. A message will also print at the top of the frequency distribution table warning that data was discarded.

In addition to listing the values found in the data, frequency distributions show the number and sum of values tabulated, the number answering, not answering, and invalid. A mean of values can be requested with the keyword **MEAN** and a median of values (computed using the value of the median cell as the lower bound) with the keyword **MEDIAN**.

7.2.3 Character Arrays

A character array is similar to a frequency distribution, but instead of displaying numeric values, it counts the unique character strings that appear in a character field and displays them in ASCII sequence, along with their frequency, percentage and cumulative percentage. Character arrays are specified as a character field up to 24 bytes wide prefixed by the symbol @ and followed, after one or more spaces, by the maximum number of strings expected.

Character arrays are subject to the same limits as frequency distributions (see section 7.2.2) and show most of the same standard summary information, omitting the sum of values. There is also no option to show a mean or median for a character array.

7.3 Table Options

The following keywords may be used in QMARG table definitions:

BASE <VAR>	Apply this variable as a base to this table. See section 7.3.2 for details on using bases.
MEAN [numberset]	In marginals counting punches, compute a weighted mean using punch values or assigning the numbers in the numberset named to punches as values. In frequency distributions, compute a weighted mean of the values displayed in the table.
MEDIAN	In frequency distributions, compute the median value using the value of the median cell as the lower bound in the computation.
NOCONTINUED	Do not print 'continued' under the table name on continuation pages if a table overflows the first page.
NOMULT	Do not print the 'multiples' line at the bottom of tables in column-binary address modes.
PCELLS	Modify default punctuation for all tables. See section 7.3.1 for details on available options.
SPREAD <VAR>	Apply this variable as a spread to this table. See section 7.3.2 for details on using spreads.
STAR 'x'	Put the character 'x' in non-zero cells rounding to zero (default '0').
TEST SPECS|nnn|ALL	Stop after testing specs/Run only on first nnn cases/all cases. Default is run ALL cases.
WEIGHT|WT <weight>	Multiply each case counted in the table by the weight value named. See section 5.3 for how to specify weights.
ZCELLS **'x'**	Put the character 'x' zero cells (default '-').

7.3.1 Print Cells and Punctuation

By default, tables of marginals show frequencies and percents based on the table total with no decimal places and a % sign following the percents.

By default, frequency distributions and character arrays show the following four columns: unique values or character strings found in the data, the frequency with which each appears, the percent based on the number answering and the cumulative percent, both to one decimal place.

The PCELLS keyword allows these defaults to be changed using a single parameter consisting of a list of cells to print and their punctuation codes, separated by commas. Cell punctuation codes are specified in the general format C[B].[P]D, where C is the cell, B an optional base for percentage cells, P a punctuation code and D the number of decimal places (0-3) to display.

The four cell designations are:

F Frequency

P or **D** Percents

C Cumulative percents (frequency distributions and character arrays only)

V Values (frequency distributions only)

In tables of marginals only, the percent cell can be suppressed from the output by specifying the PCELLS keyword and omitting the percent cell from the list. Otherwise, any cells not explicitly specified will print using their default settings. Character arrays always use the actual strings found in the data as labels and this cannot be changed.

The following special punctuation codes may be used:

C Use commas to punctuate numbers 1,000 or larger

M Toggle % signs on/off (normally only shown for whole percents)

P Put parentheses around numbers

+ Add leading signs to all numbers (not just negatives)

$ Add leading dollar signs

The following example prints integer frequencies to one decimal place with a dollar sign in front, and percents to 2 decimal places based on the total answering and with the % sign turned on.

```
PCELLS  F.$1,PTA.M2
```

The next example would show the values field in a frequency distribution to 3 decimal places, and both the percents and cumulative percents as whole numbers with a % sign.

```
PCELLS V.3,D.M0,C.M0
```

7.3.2 Bases and Spreads

QMARG allows one or two base variables to be applied to any table as a filter. Each table base is specified by the keyword BASE followed by the variable name. Title text from the variable will print below title text defined in the table definition, and text from the second base will print below text from the first base.

The test condition specified in the base variable is applied before data are counted in the table and if it is not satisfied, the table is skipped altogether for the current case. A variable used as a base should have only a single test condition specified. When multiple conditions are specified in a variable used as a base, cases that test true for any condition will be counted in the table, but since this is not guaranteed, it is preferable to use OR logic to combine the conditions.

A SPREAD applies a list of separate test conditions individually to each of the columns in a table of marginals. A base variable used as a spread should always have exactly as many test conditions as there are data locations specified in the table definition to which it will be applied. If more conditions are specified than data locations, the extra conditions are ignored, but if fewer conditions are specified than data locations, the extra locations will not be tabulated at all.

7.4 Sample Table Definitions

Here are some examples of QMARG table definitions with descriptions as title text:

```
MARG1  101t180  MEAN  pvals1
MARG1  - a table of marginals with means applying values from numberset pvals1

MARG2  001t080  CHAR  BASE MEN  BASE NEW
MARG2  - a table of character marginals with variables MEN and NEW as bases

MARG3  101t180  BASE men
MARG3  - marginals using the variable named men as a base (generates 8 tables)

MARG4  101*10  SPREAD 102PUN  BASE 103_1
MARG4  - marginals showing column 101 10 times, filtered by the test conditions
MARG4  - defined in variable 102PUN (which should have 10 conditions) and based
MARG4  - overall on variable 103_1 (which should have 1 condition).

FREQ1  ;121c4  MEAN   MEDIAN
FREQ1  a frequency distribution of values in field 121c4 with mean and median

FREQ2  ;121c4.2
FREQ2  a frequency distribution of values in field 121c4 divided by 100

ARRAY1  @31c5
ARRAY1  an array of the character strings found in 31c5
```

8 DICTIONARIES

A Dictionary is a lookup table by which labels are assigned definitions that will be substituted for them whenever referenced in the spec file. Dictionary substitutions may be used anywhere in the spec file, including in subsequent dictionary definitions. A QMARG spec file may contain any number of dictionaries anywhere before the table definition section.

8.1 Dictionary Labels and Definitions

A dictionary begins with a DICT statement and ends with an END statement. Every line between them, except for comments and blank lines, must be a dictionary entry beginning with a label in column 1, followed by two definitions separated by spaces and enclosed by double quotes ("). All text after the second definition is treated as a comment and ignored, although it will appear in the listing file. The format is:

DICT
<LABEL> <"Definition 1"> <"Definition 2">
...
END

For each entry, the contents of the first definition will be substituted whenever the label appears in the specification area of a variable or weight definition. The contents of the second definition will be substituted for the label whenever it appears anywhere else in the spec file.

The contents of a definition are only checked if and when a substitution is made in the spec file. A dictionary definition may refer to another dictionary item previously defined in the spec file.

Labels are case sensitive and may be of any length, terminated by a space. Labels are read sequentially from left to right and each label must be unique in a spec file. QMARG checks

labels for uniqueness as it compiles the spec file: if ABC is a label, then ABCD will be rejected, because it will be recognized as ABC after the first three characters have been read, whereas ABD will be accepted, because AB was not recognized as a label.

8.2 Using Dictionary Substitutions

Once a dictionary entry has been defined, it may be used anywhere in the spec file by entering its label, prefixed by a double pound sign (##), exactly where the substitution is desired.

Substitutions take place immediately, as the spec file is compiled, and work exactly as if the contents of the definition had been entered where the label appears in the specs. The following example illustrates how dictionary entries are defined and substitutions are called for in specs:

```
DICT
STATE      "62"     "GEOGRAPHIC LOCATION"
EAST       "01"     "EASTERN REGION"
SOUTH      "456"    "SOUTHERN REGION"
CENTRAL    "23"     "CENTRAL REGION"
WEST       "78"     "WESTERN REGION"
END

VAR REGION             L B24
##STATE'##EAST'         ##EAST
END
```

The result of the preceding would be exactly as if the variable REGION had been written as:

```
VAR REGION             L S24
62’01’                  EASTERN REGION
END
```

Note how the first definition is always used when a substitution is made on the specification side of a variable definition line, while the second definition is always used in the label area.

QMARG parses the text immediately following the ## keyword for a valid label, and continues until it finds the longest one so far in its dictionary as it processes the spec file. This allows labels to be concatenated with additional text, or even other labels, to determine which dictionary entry will be used for a substitution.

8.3 Built-in Dictionary Labels

QMARG provides several built-in dictionary labels for which the definitions are generated by the program when it runs. These are:

##SYS_DATE	The current date at the time of execution, as it appears at the top of the listing, in the format: Mth dd yyyy (e.g., Dec 16 1998)
##SYS_TIME	The time at which QMARG began execution, as it appears at the top of the listing, in the format: hh:mm (e.g., 15:30)
##SYS_INTAP	The name of the input file as it appears in the first INTAP data directive specified
##SYS_REPORT	The name of the report file as it appears in the REPORT data directive

These built-in dictionary entries have the same contents for both spec and text definitions.

8.4 Command Line Parameters

QMARG provides for pairs of parameters and their arguments to be passed to the program on the command line. Each parameter becomes a dictionary entry label, and its argument is used as the contents of both spec and text definitions for that dictionary entry. The format is:

QMARG specfile **/# Parameter1 Argument1 Parameter2 Argument2 .../#**

The following example demonstrates how a command line parameter can be used with a table of dictionary entries to select the output format of a QMARG run:

```
QMARGD QBDEMO.QBS  /# VERSION A /#              <--- Command line
...
DICT
PROJ_A  ""  " 25 25 25 25 0″
PROJ_B  ""  " 40 30 20 10 0″
END
...
NS  PROJECT  ##PROJ_##VERSION
...
```

In this example, the argument for the command line parameter "VERSION" is "A" which will be substituted for ##VERSION in "##PROJ_##VERSION " in the numberset definition. QMARG will parse this to recognize the dictionary label PROJ_A and use the text definition for that dictionary to complete the numberset definition. When using command line parameters in this manner, care must be taken that any argument supplied is accounted for in the specs, otherwise the reference will not resolve, causing a fatal error at run-time.

9 FLOW CONTROL AND BATCH PROCESSING

QMARG is strictly script-driven, as are most of the programs in the QUIP System. It is designed for a production environment and optimized for repetitive processing tasks. QMARG can also be used as a tabulation engine for "user friendly" systems built with environments or languages that can generate text instructions and run external processes, such as FoxPro, Delphi, Visual Basic, and most standard programming languages.

Through judicious use of dictionaries and templates, it is possible to design "generic" spec files that can be used for a number of similar projects, needing only minimal information to be added or modified for each individual set of tabulations. QMARG provides flow control features to assist with batch processing and process automation situations, and allow building modular libraries of spec files that can be selected and included into other files at run-time.

9.1 Includes

The include statement allows a spec file to call another spec file. QMARG processes an included spec file as if its entire contents appeared in the calling file beginning at the line containing the include statement, and then returns to processing the calling file at the next line. Included spec files can in turn call other spec files, but the flow of specs will always return through all nested levels to the original file.

An include statement begins with the keyword #INCLUDE in column 1, followed by the name of the spec file to be included. The format is:

#INCLUDE <[path]specfile.ext>

An include statement must always follow an END statement and any spec file that will be called by an include statement must also be terminated by an END statement, so an include cannot appear within a variable definition or a dictionary.

9.2 Stopping and Starting Specs and Listings

QMARG provides special statements that allow the flow of spec processing to be turned on or off when reading a spec file, and to suppress the output of those specs processed to the listing file. The flow control statements all consist of a single beginning in column 1 with a keyword and, if allowed, its parameter, with no other text on the line. They are:

STOPSPEC	Stop processing the spec file from this point until a STARTSPEC statement is read.
STARTSPEC	Resume processing the spec file if it has been stopped by a STOPSPEC statement.
STARTSPEC IF <##X=Y>	Resume processing the spec file if it has been stopped by a STOPSPEC statement and if the definition of dictionary entry X matches the character string "Y".
STOPLIST	Stop sending any output to the listing file from this point until a STARTLIST statement is read.
STARTLIST	Resume sending output to the listing file if it has been stopped by a STOPLIST statement.

All of these statements must appear after an END statement and take effect immediately and unconditionally, except for STARTSPEC IF, which applies only if the condition evaluates true. STARTSPEC IF may be used with any dictionary entry, however it is most effective when used with command line parameters (see section 8.4) to control inclusion or exclusion of sections of spec files at run time.

In the following example, one or the other of two spec files (or neither) will be included in the calling spec file depending on the value of the dictionary entry labeled RUNCODE:

```
STOPSPEC
STARTSPEC IF ##RUNCODE=A
#INCLUDE subfile1.qbs
END
STOPSPEC
STARTSPEC IF ##RUNCODE=B
#INCLUDE subfile2.qbs
END
STARTSPEC
```

10 WHERE TO GET ADDITIONAL INFORMATION

For information about QMARG and other QUIP System programs, contact:

Jan Werner Data Processing

www.jwdp.com