GILSOFT

Home Computer Software

The Quill
Adventure Writing System

For the BBC B & Electron

WELCOME to the world of Adventure...

With "The Quill" adventure writing system you can create an infinity of worlds in the classic adventure style.

No knowledge of programming is required to create original machine code games - this package contains all you need to start writing adventures, including a large manual which details The Quill and its use.

The Quill has received several awards for its ease of use and has been used to write many successful commercial games - even if you don't write the next number one hit, you can still have great fun writing adventures - a must for every budding adventurer!

"The Quill" for the BBC/Electron by Neil Fleming-Smith

"The Quill" in its present form first implemented by Graeme Yeandle on the Spectrum

C O N T E N T S

What is Adventure?

ADVENTURE can be described as a computerised version of the game Dungeons & Dragons. In Dungeons & Dragons one person is nominated as the dungeon master, and he invents a dungeon for other players to explore and try to retrieve hidden treasures which are usually protected by monsters of various shapes and sizes. Each player notifies his proposed actions to the dungeon master who decides on the outcome, sometimes with the help of dice to introduce a random element.

In Adventure the computer takes the place of the dungeon master and the player or players explore a predefined dungeon. Most Adventures will contain a vocabulary of words which the computer 'understands', a variety of locations which a player may wander around and objects which have to be used in the correct way to enable the Adventure to be solved. The computer will describe a situation to the player and invite him to decide on a course of action. The computer then tells the player the result of his action.

Everyone who plays Adventure has the problem of making the computer understand their commands. The computer will only have a limited vocabulary of perhaps a few hundred words and 'finding the right words' can sometimes be a problem, for example, if you are playing Dungeons & Dragons and the dungeon master tells you "There is a lamp nearby" then if you decide to "PICK UP THE LIGHT" the dungeon master should know what you mean. If the same situation occurs when playing Adventure, the computer may understand "GET LAMP" but may not know that LIGHT means, on this occasion, the same as LAMP or that PICK UP means the same as GET. Even so, most players will very quickly get the knack of 'finding the correct words'. However, it should be noted that it is up to the Adventure designer to decide which words are included in the computer's vocabulary.

What is The Quill?

The Quill is an adventure writer that allows someone with no programming experience to create a machine code adventure which will run independently of The Quill.

The Quill consists of two parts, the Editor and the Interpreter. The Editor is used to construct a database containing all the locations, objects, messages, etc, and commands that will make it obey your instructions, it is the database that makes your adventure unique. The Interpreter acts on the database during run-time and is saved with the database when an adventure is completed.

The Quill, apart from its obvious commercial applications, has its uses in a school environment, as a precursor to high-level languages, where it provides the user with visible rewards for their labour in a short period of time, the results being dependent on his/her imagination rather than programming ability.

Getting started

To load The Quill, type CH."QUILL" <RETURN>
You will then be asked whether or not you have a printer fitted. Type either 'Y' or 'N'.

The Main Menu will now be displayed.

Part 1 of this manual will introduce you gently to The Quill, from simple location descriptions through to complex condition tests and actions. It is strongly recommended that you work all the way through part 1 before attempting to write your own adventures.

Part 2 contains a detailed description of The Quill for reference.

The Quill on the BBC and Electron

The only difference between The Quill on the BBC and Electron on cassette is the maximum size of the database is reduced by 7k on the Electron due to the lack of mode 7. This means that if you want your adventure to work on both the BBC and the Electron then the database must *not* pass location 23476 (as given by Option O on the Main Menu). If you are using an Electron you will not be allowed to go above this point anyway.

This restriction can be overcome on both the BBC and Electron by dividing the adventure into several parts, (most effective when using discs), the use of text compression within The Quill also help. (The compression routines act on lower case letters and spaces and attain about 32% compression). The more advanced user may wish to incorporate his own machine code routines to save memory, an example of which is given later in this manual.

=== Part 1 ===

The Main Menu

The Main Menu consists of 15 options lettered A-O (A-P in the case of the disc version) and an option is selected by pressing the appropriate key on the keyboard. Options A-I then print a sub-menu of options, allowing the insertion, editing and listing of entries in the appropriate table, within the database. The remaining options will be described here.

Bytes Spare

Option O prints the amount of memory remaining free for the database, and the memory location from which it starts. If the free memory starts before 24576 then the adventure will run on the Electron. (It is vital that Electron users do not pass this mark as the database may be corrupted.) If sufficient memory is left above the database the user routines may be placed there, although several pages of memory have been reserved for this purpose elsewhere in memory.

SAVE and LOAD database

Options J and K load and save the database to/from tape or disc, and in each case you will be asked for a filename. The filename must not be a null string, but may, if the disc filing system is present, contain directory and drive information. If an error occurs during saving or loading of the database then the user will be taken back to the Main Menu. If the error occurred during the loading of a database, then the database will be corrupt, and may in turn corrupt The Quill if operated on. In this case the only safe operation is to reload a database. To abort the save or load operation at any time, press the Break key. The Escape key may be used at the prompt for a filename, if the option has been selected by mistake.

SAVE and Test Adventure

Options L and M test and save the adventure. The procedure for saving the adventure is the same as saving the database, except that once saved, the adventure cannot be reloaded into The Quill. If option L is selected, then the user is asked if diagnostics are required and the adventure is started. The user can return to The Quill be typing QUIT (It is vital that the entry for QUIT is not deleted from the Event table otherwise the only way out of the interpreter is to turn the machine off and reload The Quill).

Objects Conveyable

Option N asks the user how many objects the player should be able to carry or wear in the adventure, this can be anything from 0 to 255, but a value of 10 is usually about right.

The Input Routine

The input routine is different to the normal BBC routine, in that it allows the user to move backwards and forwards through the text, inserting and deleting from anywhere in the text, word processor style. The cursor keys moves the cursor and the DELETE key deletes the letter to the left of the cursor (a '<'). During editing the ESCAPE key will take you back to a sub-menu without making any change. RETURN will finish the edit and update the database. When editing a piece of text (a message, location or object description) the COPY key, when pressed, moves the cursor onto the next line. There are, however, some constraints on this, only 18 lines of text can be entered and a space cannot precede a tab character. The program will prevent this from occurring by ignoring any key that would cause it. Another constraint during text input is that the text cannot contain a leading space and two spaces cannot be placed next to each other, this is caused by the text compression routine, which is present during text editing. This compresses lower case letters and spaces to maximise memory space. (Note that when in lower case, pressing SHIFT and the required letter will insert the capital of that letter.) Multiple spaces and leading spaces can be inserted using the 'forced space' ('@') which will appear as a space when the text is printed during game play.

When editing an entry in the Status or Event tables the function keys become active, allowing certain commands to be typed in at a single keypress. The commands are obtained by pressing the function key, or the function key and either the CTRL or SHIFT keys.

If the RETURN key is pressed and a beep is head, then the word, number or command typed in is incorrect and will not be accepted until all of the following conditions are met:

a) All words used are present in the Vocabulary.
b) All location numbers used have been confirmed.
c) Any entry referred to is present in the database.
d) The syntax for all commands is correct.

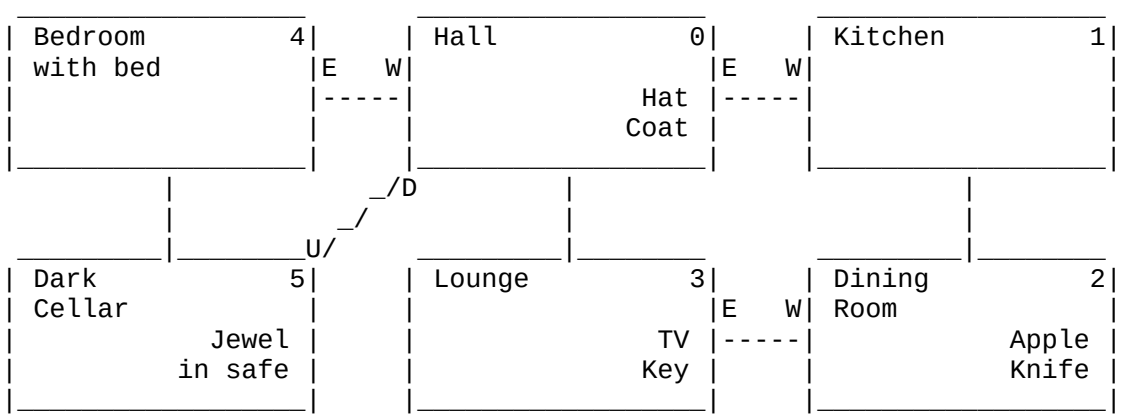
Note that <,> and ~ are not available to the user.

Setting up an Adventure

The following sections of Part 1 of this manual will give you some practical experience of using The Quill to set up an adventure. Each section depends on entries having been made to the database in earlier sections so you have to work through it step by step. If you wish to break off part of the way through, please save the database so that you can continue later where you left off.

A map of the mini adventure we are going to set up is shown in figure 1 and the objective of the adventure if to find the jewel and place it in the Dining room. The map shows all the locations in the adventure and how they are interconnected. The locations have all been given a location number, which is shown in the corner, and the position of various objects is indicated.

Figure 1 - Map of the Adventure



Before we start to set up the adventure make sure that the database is in the state it was when The Quill was loaded.

The Location Texts

The descriptions we will use for each location are as follows:-

```
Location 0
I am in the Hall, the Kitchen is to the East, the Bedroom to the
West and the Lounge to the South. Steps lead down to the Cellar.

Location 1
I am in the Kitchen. The Hall is to the West and the Dining-room is
to the South.

Location 2
I am in the Dining-room, the Kitchen is to the North and the Lounge
to the West.

Location 3
I am in the Lounge. To the North is the Hall while the Dining-room
is to the East.

Location 4
I am in the Bedroom. The Hall is to the East and a bed is against
the North wall.

Location 5
I am in the Cellar. Steps lead Up to the Hall.
```

If you select option C on the Main Menu, the Location Text Menu will be displayed, giving options 1 - 6. Option 1 inserts a location text, up to a maximum of 253. Option 2 allows you to edit a location text in the range of 0 to the total number of locations inserted into the database. Option 3 views a location text and Option 4 lists all location descriptions. (Option 5 sends a copy of this listing to the printer if fitted). Option 6 returns the user to the Main Menu.

Press 4 to list the location text table and you will see that a description is already present for location 0. The only reason for this is that the programming of The Quill was much simpler if location 0 was already present.

As a text is already present for location 0, we will have to amend it to the text required for our mini-adventure. This is done by selecting option 2 and typing 0 followed by RETURN. The present location text is then printed ready for editing, with the cursor '<' at the end, which can be moved using the cursor keys. It is worth pointing out again that the DELETE key will delete the character to the left of the cursor, the COPY key moves down to the line below, a space cannot precede a tab character or another space, and any attempt to do this will be ignored.

Getting back to our mini-adventure; delete the previous text using the DELETE key and then type in the text we need for our location 0, i.e. the Hall. Note that the computer may or may not have CAPS LOCK on, so text typed in at this stage may be in capitals. In order for the compression routine to be effective, the majority of the text description should be in lower case, and capitals are just needed to start sentences and names. Thus it may be necessary to press CAPS LOCK to change from upper to lower case.

Pressing ESCAPE at any time will take you back to the location text menu without altering the database. The database is only amended when RETURN is pressed.

The texts for the other locations in our mini adventure now have to be inserted using option 1 on the Location Text Menu. Notice that there is no need to type in a number for option 1, as the Editor automatically allocates the next location number in sequence. The Editor then places a null entry into the database for this location and then allows you to edit it. It is important that you realise that a null entry has been inserted for this location, and if you were to press ESCAPE, the null entry would remain in the database. You would have to select option 2 and specify the location number before it could be edited further.

Type in the texts that we need for our locations 1 to 5 and then list them to ensure they are correct (if you want to look at one closely, but do not want to edit it, then use option 3 to view a location description). If you have made any mistakes then please amend the texts using option 2 to correct the errors.

By now you should have inserted the texts for locations 0 to 5 and can now return to the main menu using option 6 before continuing with the next part.

The Movement Table

The interconnections between our six locations can now be entered into the database and these are placed in the Movement Table. If you select option G on the Main Menu the Movement Table Menu will be displayed and you will see that entries can be edited, viewed and listed, but not inserted. This is because when you insert a location text the Editor automatically creates a null entry for that location in the Movement Table. If you select option 4 you should see that null entries do actually exist for our locations 0 to 5.

Refer back to the map of our mini-adventure in figure 1 and you will see for location 0 that:-

EAST goes to location 1
WEST goes to location 4
DOWN goes to location 5
and SOUTH goes to location 3.

Going back to the Movement Table Menu, select option 2 and type 0 followed by RETURN to amend the entry for location 0. Now type in the following (exactly):-

```
EST 1 DOWN 5 WEST 9 SOUTH 3
```

and press RETURN.

The reason that RETURN has no effect is that there are two errors in the above entry. The first is that EST is not present in the database's vocabulary, if you replace EST by EAST this error will be rectified. The second error is caused by WEST being connected to location 9, as there are only 6 locations in our mini-adventure this is invalid, replace the 9 by a 4 and this error will also have been rectified. The entry should now read:-

```
EAST 1 DOWN 5 WEST 4 SOUTH 3
```

If RETURN is pressed the entry should be accepted. If it hasn't been accepted then the error is either caused by one of the errors mentioned above, or:-

a) The syntax of the entry is incorrect. This is caused by the proper order of word followed by location number not being followed, i.e. the entry contains a word with no number or vice versa.

b) There is no space separating the word from the location number, or there is no space separating the location number from the next word.

Note that some words in the vocabulary may have a synonym, (a word that means the same, e.g. N means the same as NORTH). It is perfectly correct to use the abbreviation in the Movement Table, and the editor, when asked to re-display the entry, will display the first synonym it comes across.

The Movement Table entries we need for our mini-adventure are:-

```
Location 0 : EAST 1 DOWN 5 WEST 4 SOUTH 3
Location 1 : SOUTH 2 WEST 0
Location 2 : NORTH 1 WEST 3
Location 3 : NORTH 0 EAST 2
Location 4 : EAST 0
Location 5 : UP 0
```

Check these with the map and then edit the Movement Table entries for Locations 1 to 5 using the Movement Table menu. You can list the entries to check they are correct if you wish.

Testing the Adventure

Now that you have entered the location texts and the movements, it is time to test the adventure, so select L on the Main Menu. You will be asked whether you require diagnostics and you should reply N. Note that the adventure always begins at location 0. You should be able to move to all our locations using the full words, e.g. EAST, or the abbreviations, e.g. E. When you are in the bedroom, try typing the following commands:-

`GO WEST`	will give "I can't go in that direction."
`GET JEWEL`	will give "I can't do that."

```
LIE ON THE BED
& GO TO THE HALL        will give "I don't understand"...
REDESCRIBE or R         will redescribe the present location
INVENTORY
or GET INV
or I                    will give a list of what you're carrying (nothing)
```

To return to the Editor type QUIT. If you have found any errors in the location texts or the movements then use the Editor to correct them. If you would like to try out the diagnostics then select L again on the Main Menu and reply Y to the prompt. The diagnostics will not help you much at this stage but note that the third number on the first row (flag no. 2) is the present location no. These are called the user flags and there are 64 of them, of which the first 3 have special meanings. The user flags will be explained in a later section.

The Objects

An object is anything that can be manipulated, e.g. a key, moved from place to place, or changed from one thing into another, e.g. an unlit torch into a lit torch. Most of the objects in our mini-adventure are shown on the map in figure 1 but a full list showing the object number, the description needed and the position of the object at the start of the adventure is as follows:-

Objno.	Text	Start location
Object 0	A lit torch.	not created
Object 1	A torch.	1
Object 2	An apple.	2
Object 3	A sharp knife.	2
Object 4	A television.	3
Object 5	A coat.	0
Object 6	A deerstalker hat.	0
Object 7	A key.	3
Object 8	A safe.	5
Object 9	A jewel.	not created
Object 10	An open safe.	not created
Object 11	A walking stick.	carried

So, the (unlit) torch starts out in the kitchen, the walking stick starts off being carried and the jewel (which can't be seen because it's in the safe) starts off as not created. Note that there are two descriptions for the safe and that it is treated as two separate objects. When the safe is opened Object 8 will be destroyed and Object 10 will be created, while the reverse will happen if the safe is closed. Similarly the torch is really two objects will be swapped over when the torch is switched on or off.

The Object Table

The descriptions of the objects are entered in a familiar way to the location texts, select option D on the Main Menu to display the Object Text Menu. You will see that it is the same as the Location Text Menu, except that all options apply to the Object Table rather than the Location Table. If you select option 4 to list the object text table you may not be surprised to find that object 0 already exists. Amend the text of object 0 so that it reads "A lit torch". Then insert objects 1-11. After you have done his and corrected any errors return to the Main Menu.

The Object Start Locations

Now that the object texts have been inserted into the database we can go about placing the objects where they will be at the start of the adventure. Select option F on the main menu to enter the Object Start Location Menu. Object start locations can be edited and listed, but cannot be inserted as this is done by the editor automatically when an object text is inserted. If you list the object start locations you will see that our twelve objects (0-11) are all 'not created' to edit the start location of an object, object no. 7 for instance, select option 2 and type 7 followed by RETURN, the present start location will be displayed, followed by a table of numbers between 253 and 255 (these have special meanings to the editor).

The editor now asks for the new start location. Pressing RETURN at this stage will leave the entry unaltered. If the start location is to be altered then delete the previous value, and type in the new value, e.g. 3 followed by RETURN.

Values of 253, 254 and 255 have special meanings. 252 means that an object is carried. 253 means that an object is worn and 255 means that an object is not created (the default value). If there is no response when the RETURN key is pressed then the number either contains a letter, is not 253, 254, 255 or is greater than the number of locations inserted so far.

Refer back to our list of objects and their start locations and amend all the other entries that need to be changed. Return to the Main Menu once this is complete.

Testing Again

It's time to test the adventure again to check that the objects are where they should be and have the correct descriptions, so select option L on the Main Menu. When you are in the adventure type INVENTORY or I to check that you are carrying the walking stick and use the map in figure 1 to check all the other objects are at the correct locations.

If you selected diagnostics you will see that the second number on the first row (flag no. 1) is now set to 1. This flag contains the number of objects being carried or worn at the present time - and you are carrying the walking stick - hence the flag has the value 1.

The Vocabulary

One section of the database contains the vocabulary and this will hold an entry for every word that the computer is to understand. The Vocabulary Menu is option A on the Main Menu and it allows for words to be inserted and deleted, for the vocabulary to be listed and for synonyms of a word to be displayed. Synonyms were mentioned earlier when we discovered that the Editor knew that NORTH and N meant the same thing. If you list the vocabulary you will see that there are over 30 words already present which will be needed by most adventures. Each word consists of up to four letters followed by a number (or word value) and words with the same word value are synonyms.

The words in the vocabulary are either whole words e.g. UP or, if the word has more than four letters then just the first four letters e.g. ASCE(ND). This has the advantage of using up only a little memory and it also reduces the amount of typing the person playing the adventure has to do

because he or she will soon learn that only the first four letters of each word are significant. The disadvantage, of course, is that you cannot have two words with different meanings which start with the same four letters. This rarely causes problems but note that when you are playing an adventure and want to go NORTHEAST you have to type in NE as the vocabulary says that NORT is a synonym of N.

Take a look through the vocabulary and you should be able to spot all the words we used in the Movement Table and in fact the Movement Table can only contain words that are in the vocabulary. Getting back to the Vocabulary Menu, select the synonyms option and type INV to see the synonyms for Inventory. Then try inserting the word ORANge with the value 200 using option 1. We haven't an orange in our adventure so delete it using option 2. Try inserting words that are already present e.g. STOP and deleting a word that isn't present e.g. COMPUTER. We'll come back to the vocabulary when we've found out what the Interpreter does when a command is typed in.

Decoding the player's command

Each time the player types in a command during an adventure, the Interpreter has to decode it. It does this by searching along the command for words which are in its vocabulary. The word value of the first word recognised is stored in a variable called W1 and the word value of the second word recognised is placed in W2. This means that commands like TURN ON THE TORCH can be reduced to ON TORCH provided that words TURN and THE are not in the vocabulary. Similarly GO TO THE EAST will mean the same as EAST if GO, TO and THE are not in the vocabulary. Thus it is important to consider which words are to be excluded from the vocabulary as well as those that are included.

If no words are recognised, the interpreter gives the reply "I don't understand...". If the interpreter recognises a word or words, but they neither cause movement (due to no entry in the movement table) nor cause an action to be performed (to be explained later) then the Interpreter gives the reply "I can't do that" if the value of W1 is greater than 12. Otherwise it replies with "I can't go in that direction". Therefore the words in the vocabulary which relate to directions should have word values in the range 0 to 12.

More Movements

The location descriptions in our adventure include statements of the form "The Hall is to the West" and up to now we have moved to the Hall with the commands WEST, W or GO WEST. We are now going to improve the adventure so that it will also obey commands of the form GO TO THE HALL or just HALL. To do this we will need an entry in the vocabulary relating to each location so insert the following entries into the vocabulary:

```
HALL                13
KITCHEN             14
DINING              15
LOUNGE              16
BEDROOM             17
CELLAR              18
```

E.g. to insert the word KITCHEN with a word value of 14 use option 1 on the Vocabulary sub-menu. List the Vocabulary to check the entries are correct then amend the entries in the Movement Table to include these new words. The Movement Table entries required are:-

```
Location 0  E 1 D 5 W 4 S 3 KITC 1 CELL 5 BEDR 4 LOUN 3
Location 1  S 2 W 0 DINI 2 HALL 0
Location 2  N 1 W 3 KITC 1 LOUN 3
Location 3  N 0 E 2 HALL 0 DINI 2
Location 4  E 0 HALL 0
Location 5  U 0 HALL 0
```

When you have changed the Movement Table, test the adventure again to check that commands such as GO TO HALL are obeyed correctly. While testing the adventure, note that when you are in the Dining Room, GO NE gives the reply "I can't go in that direction" because NE has a word value less than 13, GO TO THE HALL gives the reply "I can't do that" because HALL has a word value greater than 12.

More Words

We will shortly get to the stage where we can begin to manipulate the objects in our adventure. Before we can do that though, the words we will use to manipulate the objects have to be inserted into the vocabulary.

```
TORCH           20
APPLE           21
KNIFE           22
TELEVISION      23
TV              23
KEY             26
SAFE            27
JEWEL           28
STICK           29
UNLOCK          30
OPEN            30
CLOSE           31
SHUT            31
LOCK            31
LIGHT           32
ON              32
OFF             33
OUT             33
EAT             34
SCORE           35
BED            200
HUNGER         201
FINISH         202
COAT           224
DEERSTALKER    225
HAT            225
```

The bed is not an object in the adventure and we do not intend to make use of the word BED anywhere else in the database. However, the bed is mentioned in the location description of the bedroom so BED is included in the vocabulary to stop the Interpreter replying "I don't understand..." if a player tries to use the bed.

The Event Table

This table (and the Status Table) form the heart of the database, for it is here that the actions the Interpreter has to take to reply to a player's command are specified. Each entry in the table consists of two word values and a set of commands. When the adventure is played, the Interpreter matches the word values entered by the player (which have been stored in W1 and W2) against each entry in the table. If the word values match then the commands in that entry are executed. These commands consist of conditions and actions which can be freely mixed together. If a condition is not met then the remaining commands are ignored and execution continues with the next entry in the table. Select option H on the Main Menu and the Event Table menu will be displayed, and as with other options, entries can be inserted, amended and listed. When inserting an entry you are asked for the two words which will identify this entry. If the words are not present or ESCAPE is pressed. Once the entry has been identified then the actual commands can be inserted, and these are of the form of a command, or a command followed by one or two numbers. (Several of the commands are available at a single keypress by using the function keys, on their own or in combination with the SHIFT or CTRL keys.) Spaces must be placed between all commands and numbers.

Action INV

This is the action which prints "I have with me:-" etc. The entry in the Event Table, GET INVE, has no conditions and a single action called INV. What does this mean? Well, if you were playing the adventure and typed in TAKE INVENTORY then because the word value stored in W1 matched the word value for GET and W2 matched INVE, action INV would be performed. Remember that TAKE & GET and INVE & I are synonyms.

The next entry INVE * does the same thing. If the first word recognised by the Interpreter is a synonym of INVE then the action INV is performed. The * means that the corresponding value in W2 is irrelevant. Please make sure you fully understand the last two sections on the Event Table before reading further.

Action DESC

The entry in the Event Table DESC * means that if REDE(SCRIBE) is entered as a command then action DESC will be performed. Action DESC clears the screen and attempts to describe the current location. (If it is dark the Interpreter will print "Everything is dark. I can't see.")

Actions SAVE and LOAD

These are the actions which copy a game position to tape and restore a game position from tape. The last two entries in the Event Table use these actions and mean, for instance, that if LOAD is typed in as a command then action LOAD is performed. Do not be confused by the vocabulary word LOAD and the action word LOAD because they are not related. You can only have an entry in the Event Table of LOAD * if there is an entry for LOAD in the vocabulary. The action word LOAD is independent of the vocabulary.

Actions QUIT and END

Action QUIT asks "Are you sure?". If you reply 'N' then the adventure will be rejoined, but if you type 'Y' then the next action (usually END) is invoked. Action END is a very important action as it prints "End of game. Do you want to play again?". 'N' will return you to the Quill Editor, 'Y' will cause the game to be restarted from the beginning. It is important that you do not remove END from the Event Table, as there is no other way of returning to the Editor.

Actions GET, DROP and OK

Actions GET and DROP must be followed by an objno. and are used to carry or put down objects while action OK simply prints "OK". In our mini adventure we want the player to be able to take and leave the walking stick using the commands TAKE STICK and DROP STICK respectively. The entries needed in the Event Table for this are:-

TAKE STICK	GET 11 OK
DROP STICK	DROP 11 OK

To insert the first of these into the Event Table, go to the Event Table (option H) followed by option 1 on the sub-menu and type TAKE STIC and press RETURN. This inserts a null entry into the Event Table and then displays a cursor to allow you to enter the commands for TAKE STICK, now type GET 11 OK and press RETURN. (GET can be obtained by pressing function key 6 or can be typed out in full). Insert the entry for DROP STIC in the same way and then list the Event Table to check the entries. Note that the editor will have changed TAKE into GET as GET is a synonym for TAKE and comes before TAKE in the vocabulary.

Now test the adventure to see the effect of these entries in the Event Table. In particular try the following commands:-

TAKE STICK	when you are already carrying it
DROP STICK	when you are not carrying it
GET STICK	when the stick is at a different location

Condition PRESENT

The entries we need, to be able to GET and DROP the other objects are:-

Words	Commands
GET APPLE	GET 2 OK
DROP APPLE	DROP 2 OK
GET KNIFE	GET 3 OK
DROP KNIFE	DROP 3 OK
GET COAT	GET 5 OK
DROP COAT	DROP 5 OK
GET HAT	GET 6 OK
DROP HAT	DROP 6 OK
GET KEY	GET 7 OK
DROP KEY	DROP 7 OK
GET JEWEL	GET 9 OK
DROP JEWEL	DROP 9 OK
GET TORCH	PRESENT 0 GET 0 OK
GET TORCH	GET 1 OK
DROP TORCH	PRESENT 0 DROP 0 OK
DROP TORCH	DROP 1 OK

As you can see, the entries for the torch are not quite so simple. The problem is that the torch is really two objects so we have to have two entries. The condition PRESENT, which must be followed by an objno., checks whether the object specified is present at the current location. So the first entry for GET TORCH checks whether Object 0 is present and if it is, tries to GET it. If Object 0 is not present the condition is not satisfied so the Interpreter falls through to the next entry, which has no conditions, and tries to get Object 1.

When you have inserted the entries, list them to check they are correct and that the entries for the torch are in the correct order. Amending entries in the Event Table is similar to amending entries in the other database tables. However, if there is more than one entry present for the same words, e.g. DROP TORCH, then each entry is displayed in turn for amending and you simply press RETURN to leave an entry as it is. Try Selecting option 2 and type DROP TORC on the Event Table menu and keep pressing RETURN until you get back to the menu; both entries should be displayed in turn for possible amending.

Entries in the Event Table can be removed by deleting any existing commands, so that a null entry is left, and then pressing RETURN.

Actions WEAR and REMOVE

THESE actions enable objects to be worn and removed and must be followed by an objno. Our adventure has two objects that can be worn and the entries needed in the Event Table for this are:-

Words	Commands
WEAR HAT	WEAR 6 OK
REMOVE HAT	REMOVE 6 OK
WEAR COAT	WEAR 5 OK
REMOVE COAT	REMOVE 5 OK

Insert these entries and then test the adventure again using diagnostics. The reason for using diagnostics is to enable you to monitor the value of Flag 1 (the second flag) which has a value equal to the number of objects carried. When you are in the adventure try these commands in the order shown:-

Command	Response	Flag 1 value
WEAR HAT	I don't have it	1
REMOVE HAT	I'm not wearing it	1
GET HAT	OK	2
TAKE COAT	OK	3
KITCHEN		3
GET TORCH	OK	4
DINING		4
GET APPLE	OK	5
WEAR HAT	OK	5
WEAR HAT	I'm already wearing it	5
REMOVE HAT	OK	5
DROP HAT	OK	4
WEAR COAT	OK	4
I	I have with me...	4
REMOVE COAT	OK	4

Also make sure you can GET the key and drop all the objects you are carrying.

As you can see, the actions GET, DROP, REMOVE and WEAR include quite a bit of checking on the objects being manipulated and they can normally be used without being preceded by conditions. These actions are exceptions to the rule as most other actions will need to be preceded by conditions.

Action SWAP

This action is followed by two objnos. and simply swaps over the positions of the two objects. e.g. if Object 0 is 'not created' and Object 1 is at location 3 then the action SWAP 0 1 would put Object 0 at location 3 and make Object 1 'not created'. In our adventure we are going to use SWAP to switch the torch on and off. The entries we need are:-

```
Words              Commands
ON TORCH           PRESENT 1 SWAP 0 1 OK
TORCH ON           PRESENT 1 SWAP 0 1 OK
OFF TORCH          PRESENT 0 SWAP 0 1 OK
TORCH OFF          PRESENT 0 SWAP 0 1 OK
```

Notice that we have catered for a player giving the command SWITCH OFF TORCH or SWITCH TORCH OFF and that the condition PRESENT 0 means that the torch can only be switched off if the lit torch is present. Insert these entries into the Event Table and test them.

The Flags

The interpreter contains 64 user flags which are really just variables that can hold a value in the range 0 to 255. Like everything else in The Quill, numbering starts at 0 so the flag numbers (flagno.s) are in the range 0 - 63. There are actions which enable the flag values to be changed and conditions which allow the flag values to be tested. Some of the flags have special purposes (flags 0, 1, 2 and 63). Flag 0 is used to tell the Interpreter whether it is light or dark; if it has any value other than zero the Interpreter thinks it's light. Flag 1 as we have seen before, holds a count of the number of objects carried. Flag 2 contains the present location and flags 48 to 62 act as double byte numbers (i.e. have a range of 0 - 65535 instead of 0 - 255), but are printed as
normal flags by the diagnostics and are only printed in full by the PRINT command. Note that Flag 63 is set to 255 whenever a location is described.

Note that a flag cannot be decreased below zero and that Object 0 is a special object because the Interpreter considers it as a source of light.

There is a summary of the flags, conditions and actions that can be used on the last page of this manual. As mentioned earlier, when diagnostics are requested the values of flags 0 - 63 are printed so that you can monitor the values of them.

Light and Dark

If you take another look at our map in figure 1 you will notice that it says the cellar is dark. To make the cellar dark, we will use the condition AT and the actions CLEAR, SET, GOTO and DESC. Condition AT must be followed by a locno. and is satisfied if the player is at that location. The actions CLEAR and SET must be followed by a flagno. and change that flag's value to 0 or 255 respectively. Action GOTO is followed by a locno. and causes movement to that location.

If this entry were present in the Event Table (do not insert it):

Words	Commands
U *	AT 5 GOTO 0 DESC

It would have exactly the same effect as out entry U 0 for location 5 in the Movement Table i.e. if a player gives the command UP when he is at location 5, he is moved to location 0 and the location is described.

The Event Table entries to be inserted to make the cellar dark are:-

Words	Commands
D *	AT 0 CLEAR 0 GOTO 5 DESC
CELLAR *	AT 0 CLEAR 0 GOTO 5 DESC
U *	AT 5 SET 0 GOTO 0 DESC
HALL *	AT 5 SET 0 GOTO 0 DESC

When a player moves from the Hall to the Cellar, Flag 0 will be set to 0 (making it dark). When he moves back to the Hall, Flag 0 will be set to 255 (making it light again). When you are testing the adventure monitor the value of flag 0 and try experimenting with moving between the Hall and cellar and switching the torch on and off. You can delete the entries for D & U in Movement as they have no effect now.

Opening and Closing the Safe

In our adventure the safe can only be opened if the key is carried, the lit torch is present and, of course, the safe is not already open. To close the safe, the key is not needed but the lit torch must be present. The first time the safe is opened we want to create the jewel and give the player a score of 50% but if the player closes the safe and reopens it then we must not create the jewel again or give another 50%.

We will use Flag 11 to show whether the safe has already been opened. As the flags start off with a value of zero we will say that if Flag 11 has the value 0 the safe has not previously been opened. Then if we SET Flag 11 the first time the safe is opened we will be able to tell that the safe has already been opened. The Event Table entries we need are (note they are in the form that you would see if you List the Table - they must be inserted in the same way as the above entries, it just makes them easier to read).

```
OPEN SAFE          PRESENT     8
                   CARRIED     7
                   PRESENT     0
                   ZERO       11
                   DESTROY     8
                   CREATE     10
                   LET        30      50
                   CREATE      9
                   SET        11
                   DESC

OPEN SAFE          PRESENT     8
                   CARRIED     7
```

```
                        PRESENT     0
                        DESTROY     8
                        CREATE     10
                        OK

CLOSE SAFE              PRESENT    10
                        PRESENT     0
                        DESTROY    10
                        CREATE      8
                        OK
```

Condition CARRIED must be followed by an objno. and is satisfied if that object is carried while condition ZERO must be followed by flagno. and is satisfied if that flag has the value 0. Action CREATE is used to change an object's location to the current location while action DESTROY changes an object's location to 'not created'. Both CREATE and DESTROY must be followed by an objno. Action LET is followed by a flagno. and a value, and the flag is given that value e.g. LET 30 50 gives Flag 30 a value of 50; a bit like the BASIC command LET FLAG30=50.

The order of the two entries for OPEN SAFE for important. The first time the safe is opened the condition ZERO 11 will be satisfied so the actions in the first entry will be performed. If the safe is opened again the condition ZERO 11 will not be satisfied and the Interpreter will fall through to the second OPEN SAFE entry. Insert the above entries into the Event Table then list them to check they are correct. Note the first OPEN SAFE entry ends with action DESC so that the player can see that the jewel has been created. When you have checked the entries, test the adventure again and use diagnostics so that you can monitor the values of flags 11 and 30.

The Message Texts

Any messages which are needed in the adventure have to be entered into the Message Text Table. The Message Text menu is similar to the Location Text menu with the exception that it operates on the Message Table instead of the Location Table. If you select B on the Main Menu to display the Message Text menu and then list the messages, you will see that messages 0 to 19 already exist. These are called the system messages, and may be changed to read 'You' instead of 'I', but the meaning of the messages should not be altered. The messages we need for our adventure are:-

```
Message 20
I'm hungry!
Message 21
Ah. That's better.
Message 22
I'm dying of starvation...
Message 23
Well done. You've solved the Adventure.
Message 24
You have scored
Message 25
%
```

Insert these messages before proceeding.

Implementing a percentage scoring system

As mentioned in the section on Opening & Closing the Safe the player will have completed 50% of the adventure if he gets the jewel and this will be stored in flag 30 (it could be any flag between 3 and 47, which are all single byte flags). This percentage must be displayed at some point, and this is done by the PRINT command, which prints out a flag's contents. We will also use two of the messages inserted earlier (24 & 25).

There are two actions available to print messages. These are; MESSAGE which prints the given message and then a new line, and MES which prints the given message only, thus the routine to insert in Event is:-

```
SCORE *         MES         24
                PRINT       30
                MESSAGE     25
                DONE
```

This should also be placed in the entry for QUIT (using option 2 to Edit it), i.e.:-

```
QUIT *          QUIT
                MES         24
                PRINT       30
                MESSAGE     25
                END
```

The combination of MES, PRINT and MESSAGE can also be used to print the number of turns taken, gold collected, points scored, etc. and using PRINT with a flagno between 48 and 62 (double byte flags), values up to 65,535 can be printed.

The Status Table

This table has exactly the same format as the Event Table. i.e. each entry has two word values followed by conditions and actions, but the Interpreter uses the two tables in slightly different ways. We have already seen that the interpreter uses the Event Table after each command given by a player and matches the word values in W1 and W2 with each entry in the table. The Interpreter uses the Status Table in between turns and looks at each entry irrespective of the word values in W1 and W2. Thus the Event Table contains entries which are dependent on the commands entered by the player while the Status Table contains entries that are independent of the commands entered by the player. Note that the Interpreter does not make use of the word values present in the Status Table entries. The vocabulary words used in the Status Table can therefore be used as commends, to remind you what the entries do, or to position an entry at a particular place in the table because the entries are arranged in ascending order of word value. When you write your adventure you will find that the order of the entries in the Status Table is very important.

Entries at the start of the Status Table can be used to keep track of the number of turns taken, how long a lamp will stay alight, etc, and can make any flag increment or decrement every turn or only under certain circumstances.

Finishing the Adventure

Our mini adventure is solved when a player is at the Dining Room and the jewel is present. It doesn't matter what commands the player uses to get the jewel to the DINING ROOM so we will use an entry in the Status Table to detect when the adventure is solved. The Status Table entry required is:-

```
FINI *          AT            2
                PRESENT       9
                MESSAGE      23
                PLUS         30    50
                MES          24
                PRINT        30
                MESSAGE      25
                END
```

Note that the vocabulary word FINI(SH) is used only as a comment.

Select I on the Main Menu to display the Status Table menu and then insert this entry. Action PLUS must be followed by a flagno. and a value and it adds that value to the appropriate flag. e.g. PLUS 30 50 adds 50 to flag 30 (the score). PLUS will only count up to 255 for flags < 48, but flags over this may overflow, and if this happens the next flag up is incremented. i.e. if we were to insert an entry containing PLUS 50 100 and flag 50 already contained 200, then flag 51 would be incremented and flag 50 would contain 45 (the overflow).

When you have inserted this entry in the Status Table, test the adventure again and take the jewel to the Dining Room.

Hunger

In our adventure we want the player to become hungry after six turns, to remain hungry for the next seven turns and then to die of starvation. Eating the apple, of course, will ward off the pangs of hunger. We will set flag 12 when the apple is eaten and we will use flag 5 to count the seven turns when the player is hungry. The entries needed in the Status Table are:-

```
HUNG *          ZERO         12
                ZERO          5
                LET           5    14

HUNG *          ZERO         12
                MINUS         5     1

HUNG *          LT            5     8
                ZERO         12
                MESSAGE      20

HUNG *          ZERO          5
                MESSAGE      22
                PAUSE       100
                MES          24
                PRINT        30
                MESSAGE      25
                END
```

Note that the vocabulary word HUNG(ER) is only used as a comment.

Condition LT must be followed by a flagno. and a value, and is satisfied if the flag specified contains a value Less Than the one specified. Action MINUS is also followed by a flagno. and a value and causes the value to be subtracted from the specified flag. It will not reduce the value below zero. The action PAUSE waits for a set time. Insert these four entries into the Status Table but make sure you insert them in the order shown. The first entry sets flag 5 to 14, this only occurs once. The second entry decrements this by 1 every turn if flag 12 is 0. The third entry prints message 20 ("I'm hungry") if flag 5 is < 8. The fourth entry only executes if flag 5 = 0 (died of starvation) and prints message 22 "I'm dying of starvation", waits a while, then prints out the score and ends the game.

Eating the apple

We have already said that we will set Flag 12 when the apple is eaten. However, we only want to print message 21 if the apple is eaten when the player is hungry, i.e. if the apple is eaten before the player is hungry we just want the reply "OK". The entries required in the Event Table are:-

```
EAT APPL        PRESENT      2
                LT           5     8
                NOTZERO      5
                DESTROY      2
                SET         12
                MESSAGE     21
                DONE

EAT APPL        PRESENT      2
                DESTROY      2
                SET         12
                OK
```

The condition NOTZERO is followed by a flagno and is satisfied if the flag specified does not have the value 0. Action DONE simply stops the Interpreter falling through to the next entry in the table. Insert these two entries into the Event (Not Status) table in the correct order.

Number of Objects Conveyable

You may remember when you were testing the adventure earlier, that the player was able to carry several objects. This number can be altered by selecting option N on the Main Menu. For our mini adventure change the number of objects conveyable to 1 (one) and then test the adventure for the last time (it won't be quite so easy this time).

SAVE Adventure

This option allows complete copies of an adventure to be saved to tape (or disc) in a form that can be run using the *RUN 'filename' command, or can be run from a BASIC header, which can contain a loading screen, setup routines or load in user routines. Note that a saved adventure is *not* designed to be reloaded into The Quill.

The Object Word Table

This table is used by the actions AUTOG, AUTOD, AUTOW and AUTOR (short for Autoget, Autodrop, Autowear and Autoremove) and its purpose is to enable objects to be manipulated without you making hundreds of entries in the Event Table of the form:-

```
GET HAT             GET 6
                    OK
```

As an example, we are not going to change our mini adventure so that it uses the Object Word Table. If you look at the Event Table, you will see that we have made the following entries related to manipulating objects:

Words	Commands
GET TORCH	PRESENT 0 GET 0 OK
GET TORCH	GET 1 OK
GET APPLE	GET 2 OK
GET KNIFE	GET 3 OK
GET COAT	GET 5 OK
GET HAT	GET 6 OK
GET KEY	GET 7 OK
GET JEWEL	GET 9 OK
GET STICK	GET 11 OK
DROP TORCH	PRESENT 0 DROP 0 OK
DROP TORCH	DROP 1 OK
DROP APPLE	DROP 2 OK
DROP KNIFE	DROP 3 OK
DROP COAT	DROP 5 OK
DROP HAT	DROP 6 OK
DROP KEY	DROP 7 OK
DROP JEWEL	DROP 9 OK
DROP STICK	DROP 11 OK
WEAR COAT	WEAR 5 OK
WEAR HAT	WEAR 6 OK
REMOVE COAT	REMOVE 5 OK
REMOVE HAT	REMOVE 6 OK

Please delete these 22 entries and insert the following six entries:-

Words	Commands
GET TORCH	PRESENT 0 GET 0 OK
GET *	AUTOG OK
DROP TORCH	PRESENT 0 DROP 0 OK
DROP *	AUTOD OK
WEAR *	AUTOW OK
REMOVE *	AUTOR OK

Now select Option E on the Main Menu for the Object Word Menu and you will see that entries can be deleted or printed. Entries cannot be inserted because when you insert an object text for an object the Editor automatically inserts a null entry for that object in the Object Word Table. If you print the Object entries you should see that twelve objects all have null entries. The Object Word table is where objects are associated with Vocabulary words, e.g. to associate the word HAT with object 6, select option 2 to edit word relation and type 6, followed by RETURN and then type 'HAT'. In our mini-adventure, we need the following associations:-

```
Object     0  -
Object     1  -  TORCH
Object     2  -  APPLE
Object     3  -  KNIFE
Object     4  -
Object     5  -  COAT
Object     6  -  HAT
Object     7  -  KEY
Object     8  -
Object     9  -  JEWEL
Object    10  -
Object    11  -  STICK
```

Note that all the entries for objects 4, 8 and 10 remain null because the TV and Safe are not to be manipulated and that the Torch (objects 0 and 1) has object 1 handled by the Object Word Table while Object 0 still has to be handled explicitly in the Event Table.

If you list the Object Word Table now you should get the following information:-

```
Object    0
Object    1   TORC GD
Object    2   APPL GD
Object    3   KNIF GD
Object    4
Object    5   COAT GDWR
Object    6   HAT  GDWR
Object    7   KEY  GD
Object    8
Object    9   JEWE GD
Object   10
Object   11   STIC GD
```

The GD or GDWR signifies which of the Actions AUTOG, AUTOD, AUTOW and AUTOR will operate for each object. Whether GD or GDWR is shown depends entirely on the word values of the words used. If the word value is < 200 then GD is shown otherwise GDWR is shown. Thus objects which can be worn should be associated with words which have word values > 199. If you test the adventure now, you will find that it works in exactly the same way as before the Object Word Table was used.

Deleting Words for the Vocabulary

Note that a word will not be deleted from the vocabulary if it has no synonyms and:-

a) The word is used as a direction in the Movement Table
b) The word is sued as an association in the Object Word Table
c) The word is used in the Event or Status Tables

Other Actions

Most of the actions available in The Quill were used in our mini adventure. The action actions are detailed in Part 2 of this manual. One use of the KEY action is to provide an introduction to adventures.

If the introduction occupies two screens, then insert it into the database as the texts for locations 0 & 1 and let the adventure proper start at location 2. The following entries in the Status Table can then be used to do the introduction:-

```
INTRO *         AT              0
                KEY
                GOTO            1
                DESC

INTRO *         AT              1
                KEY
                GOTO            2
                DESC
```

More about Event and Status

When the Interpreter is processing these tables, it considers each entry until it reaches the end of the Table or it performs one of the actions INV, DESC, END, DONE, OK, SAVE or LOAD. The actions REMOVE, GET, DROP, WEAR, AUTOG, AUTOD, AUTOW or AUTOR can sometimes also stop processing of the Event/Status tables. Full details of all the actions can be found in part 2 of this manual.

Other Conditions

Our mini adventure made use of only a few of the conditions available in The Quill. Details of the other conditions can be found in Part 2 of this Manual but it is worth mentioning the condition CHANCE here as it can introduce a random element into an adventure. Condition CHANCE must be followed by a percentage, in the range 1-100, e.g. CHANCE 25 would have a 25% chance of being satisfied.

Designing your own Adventure

The suggested procedure of writing your own adventure is:-

a) Read Part 2 of this manual
b) Draw a map of your adventure and allocate locnos. (Make your first adventure a small adventure)
c) List the location texts you will use
d) List the Movement Table entries you will need
e) List the objects in the adventure. Allocate objnos giving objects that can be worn objnos > 199
f) List the messages needed in the adventure and allocate mesnos
g) List all the words and synonyms you will use and allocate word values
h) Decide which flags will be used and what they will be used for
i) Write out all the Event Table entries showing the conditions and actions to be used. If you have more than one entry with the same word values check that you have the entries in the correct order
j) Write out all the Status Table entries and plan the order of the entries. You might need to add further entries to the vocabulary which will be used as comments in the Status Table
k) Insert the entries into the database and save the database regularly (You might get a power

cut!)

l) Thoroughly test your adventure

As you can see, you need to do quite a lot of planning before you start to type your adventure into The Quill.

Selling Your Adventure

If you intend to sell adventures then they need to be very thoroughly tested by as many people as possible. In particular:-

a) Check the spelling of every word
b) Check it is impossible to get a score of over 100%
c) Try to move in every direction from every location
d) Try to GET, DROP, WEAR and REMOVE each object
e) It should be possible to solve the adventure each time it is played provided the correct commands are used. e.g. poison gas which has a 1% chance of appearing and killing the player should be avoided unless you also provide a gas mask.

A great deal of time has been spent testing The Quill but it is possible that a few well hidden bugs still remain within the 8K of machine code. If you do have any problems please let us know so that they can be corrected. We would also be pleased to hear of any commends or criticisms about The Quill or any suggestions for improvements.

If you intend to sell and adventure written with The Quill we would be grateful if you could mention that it was written with The Quill somewhere in it.

=== End Of Part 1 ===

=== Part 2 ===
Detailed description of the Interpreter

Main Interpreter loop

1 **Initialise**
All the flags are set to zero except for flag 0 (location lit) which is set to 255 so that adventure starts describing locations. The current location (flag 2), W1 and W2 are set to zero.

2 **Describe current location**
If flag 0 <> 0, or object 0 is present, then the location description is printed, otherwise message 2 is printed ("All is dark"). Any objects at the present location are listed.

3 **Execute Status Table**
Execute conditions and actions contained within Status Table.

4 **Print Diagnostics**
If diagnostics are required and adventure not saved then print the contents of the 64 flags at the top of the screen.

5 **Input Commands**
Input commands from user, decode word values into W1 and W2. If only one word is found, place in W1 and W2 = 255 (i.e. '*'). If no words are recognised then print message 14 ("I don't understand...") and go to step 3.

6 **Execute Event Table**
Execute entries in Event Table that match W1 and W2.

7 **Look up W1 in the Movement Table**
If W1 is in the entry for this location in the Movement table then go to the location indicated and go to step 2. If no match is found for this location then print "I can't do that" (or "I can't go in that direction" if word value < 13) and go to step 3.

Execute Event/Status Routine

A Initialise Event/Status Execute Routine
Set a pointer to the first entry in the Event/Status table.

B Words Satisfied?
If in the Status Table then go to step C.
If in the Event Table and W1 and W2 match then go to step C otherwise go to step D.

C Execute Entry
Fetch a command.
If a condition and it's false go to step D.
If an action then execute and set action done flag, point at next command (N.B. Some actions may need end processing) if not end of entry go to step C.

D Point at next Entry
If end of table and in Status then return to step 4.
If end of table and in Event then return to step 3 if any actions carried out.
Step 7 otherwise.

Conditions can be any of the following:-

`ATLT locno.`
Satisfied if current location is less than locno.

`ATGT locno.`
Satisfied if current location is greater than locno.

`NOTAT locno.`
Satisfied if current location is not the same as locno.

`AT locno.`
Satisfied if current location is the same as locno.

`PRESENT objno.`
Satisfied if objno. is carried, worn or at the present location.

`ABSENT objno.`
Satisfied if objno. is not carried, worn or at the present location.

`NOTWORN objno.`
Satisfied if objno. is not worn.

`WORN objno.`
Satisfied if objno. is worn.

`NOTCARR objno.`
Satisfied if objno. is not carried.

`CARRIED objno.`
Satisfied if objno. is carried.

`CREATED objno.`
Satisfied if objno. is created somewhere in the adventure.

`DESTROYED objno.`
Satisfied if objno. is not created somewhere in the adventure.

`CHANCE percent`
Satisfied if percent is less than or equal to a random number in the range 1-100 (A value of 0 is always false, and a value > 100 is always true).

`NOTZERO flagno.`
Satisfied if flagno. contents are not zero.

`ZERO flagno.`
Satisfied if flagno. contents are zero.

`NOTEQ flagno. no.`
Satisfied if flagno. contents are not equal to the no.

`EQ flagno. no.`
Satisfied if flagno. contents are equal to the no.

`GT flagno. no.`
Satisfied if flagno. contents are greater than the no.

`LT flagno. no.`
Satisfied if flagno. contents are less than the no.

Actions can be any of the following:-

`INV`

Message 0 is printed ("I have with me:-"). If no objects are carried or worn then message 2 is printed ("Nothing"). If an object is carried or worn then its description is printed. If an object is worn then message 1 ("Worn") is printed after the object description.

`DESC`

This action jumps out of the Event/Status table and describes the present location.

`DROPALL`

This action drops all objects that are carried or worn and they are created at the present location.

`DONE`

This action causes the execution of the Event/Status tables to be suspended and the program to leave the table. i.e. no entries after this command will be executed within this table.

`KEY`

This action prints message 6 ("Press any key") and waits for a key.

`AUTOG`

The object word relation table is searched for an entry which matches W2. If no match is found then message 5 ("I can't do that") is printed and DONE is performed. If a match is found then the object is passed to the GET routine.

`AUTOD`

This action behaves like the AUTOG command, but passes the object to the DROP routine if found.

`AUTOW`

If the value in W2 is < 200 then message 5 ("I can't do that") is printed and action DONE is performed, otherwise it behaves like the AUTOG command, but passes the object to the WEAR routine.

`AUTOR`

This action behaves like the AUTOW command, but passes the object to the DROP routine if found.

`OK`

This action prints message 8 ("OK") and performs action DONE.

`SAVE`

This action prints message 18 ("Filename?"), under which the game will be saved. It will then save the game as a file on tape or disc, and then perform action DESC.

`LOAD`

This action prints message 18 ("Filename?") and loads the saved game from tape or disc. It also performs action DESC once the game is loaded.

`QUIT`

This action prints message 17 ("Are you sure?") and waits for a keypress. If any key other than the first character of message 19 ("Y") is pressed then action DONE is performed.

`CLS`

The screen is cleared.

`END`

This action prints message 17 ("END OF GAME. Do you want to try again?") and waits for a keypress. If any key other than the second character of message 19 ("N") is pressed then a jump is made to initialise. Otherwise a jump is made to the Editor (if it is present).

`CREATE objno.`
This action creates the object objno. at the present location, flag 1 (no. of objects carried) is decremented if the object was carried or worn.

`DESTROY objno.`
This action destroys the object objno. Flag 1 is decremented if the object was carried or worn.

`MESSAGE mesno.`
This action prints the message mesno. and follows it by a newline.

`MES mesno.`
This action prints the message mesno. but does not follow it by a new line, so the next thing printed will be on the same line.

`GET objno.`
This action carried object objno. if it is at the present location. If the maximum no. of objects conveyable has been reached, then message 16 ("I can't carry any more") is printed and action DONE is performed. If the object is already being carried or worn then message 9 ("I already have it") is printed and action DONE is performed. If the object is not at the present location then message 11 is printed ("It's not here") action DONE is performed.

`DROP objno.`
This action tries to drop an object if it is being carried. If the object is not being carried then message 10 ("I don't have it") is printed and action DONE is performed.

`WEAR objno.`
This action tries to wear an object if it is being carried. If the object is being worn then message 12 ("I'm already wearing it") is printed and action DONE is performed. If the object is not being carried then message 10 ("I don't have it") is printed and action DONE is performed.

`REMOVE objno.`
This action tries to carry an object if it is being worn. If the object is not being worn then message 13 is printed ("I'm not wearing it") and action DONE is performed.

`PAUSE no.`
This action waits for a set length of time, depending on the value of the no. following it. (Pause lasts about no/50 secs.)

`GOTO locno.`
This command causes the present location to be changed to locno. but the new location is not described.

`SET flagno.`
This action sets the contents of flagno. to 255.

`CLEAR flagno.`
This action clears the contents of flagno. to 0.

`PRINT flagno.`
This action prints the contents of flagno. If the flagno. < 48 then just the flag contents are printed. If the flagno. > 47 then the flag is printed as a two byte no. i.e. the value that is printed is ?(flagno)+256*?(flagno+1), so that the next flag up is used as the MSB for the no. Note that the print command does not print a new line after the number, this is so that a message can follow the number, e.g. '%' or 'turns'. If nothing is to follow the number then one of the messages should be left empty and this should be printed after the PRINT command.

`STAR mesno.`
This action sends the commands contained within the message text to the command line interpreter. This allows the program access to all '*' commands, and its main use is to load sections of a multi-part adventure. If the message mesno. does not contain a '*' as the first character then the action DONE is performed.

`PLUS flagno. no.`
This action adds no. to the contents of the flagno. and stores the result in the flagno. If the flagno. is < 48 and the addition causes overflow then the flagno. is set to 255. If the flagno. is > 47 and the addition causes overflow then the overflow is stored in the flagno. and flagno. + 1 (MSB) is incremented. (If the MSB also overflows then it will be set to zero).

`MINUS flagno. no.`
This action subtracts no. from the contents of flagno. and stores the result in the flagno. If the flagno. is < 48 and the subtraction causes a borrow then the flagno. is cleared to 0. If the flagno. > 47 and the subtraction causes a borrow then the result is placed in the flagno, and flagno.+1 (MSB) is decremented. (If the MSB also overflows then it will become 255).

`LET flagno. no.`
This action sets a flagno. to the value in no.

`ADD flagno1. flagno2.`
This action behaves like PLUS, but the contents of flagno2 are added to flagno1, rather than a number.

`SUB flagno1. flagno2.`
This action behaves like MINUS, but the contents of flagno2 are subtracted from flagno1, rather than a number.

`SWAP objno1. objno2.`
This action swaps the positions of object 1 and object 2 around. i.e. if object 1 is 'worn' and object 2 is 'not created' then SWAP 1 2 will cause object 1 to be 'not created' and object 2 to be 'worn'.

`PLACE objno. locno.`
This action causes object objno. to be placed at location locno. if the object was worn or carried prior to the PLACE command, then the no. of objects carried (flag1) is decremented.

`JSR lsb. msb.`
This command allows the user to include his/her own machine code routines within The Quill, to perform specific functions or personalise the adventure, examples of this are routines that automatically display the exits available, or routines that change the typefont etc. (The former is given as an example in the appendix.) The two terms that follow the JSR command are the lsb and msb of the address, in decimal. This can be found in BASIC by;

```
PRINT "lsb ="; address MOD 256
PRINT "msb ="; address DIV 256
```

The Quill memory map is given in an appendix, but it should be noted that locations &70-&8F are available to the user, as are &B00-&CFF, memory above the end of the database, and in the case of adventures that don't run on disc &E00-&1900 (This can contain a BASIC loader, exploded user definable graphics or machine code routines).

`SOUND no. no.`
This command acts in the same manner as the SOUND command in BASIC, except that the only terms are pitch and duration. If the user requires more than single notes played on a single channel, then a machine code routine must be written.

Detailed description of the Database

The database consists of a number of inter-related and also miscellaneous information. e.g. no. of objects conveyable. The tables present are:-

A **The Vocabulary**

Each entry in the table consists of five bytes and contains a four letter word, and a number in the range 0-254. Words with the same word value are called synonyms. The entries are held in ascending order of word value and within each word value, words inserted first come first.

NOTE : Whenever the editor has to convert from a word value to a word it takes the first word with that word value.
Word values less than 13 should be reserved for movement words. Words relating to objects that can be worn should have word values greater than 199.

B **The Message Table**

This table contains the text of any messages which are needed for the adventure. The messages are numbered from 0 to 252. Each entry consists of a single byte which points to the next message, followed by the message text.

C **The Location Table**

This table contains the text for all locations used in the adventure. The text is stored in the same format as the messages, but a null entry is placed in the movement table when a location is inserted.

D **The Object Table**

This table contains the text for the object descriptions. It follows the same format as the Message and Location Tables, with the exception that each object reserves one byte in the Object Word Table and in the Object Start Table when it is inserted.

E **The Object Word Table**

This table contains one byte per object, which is the object word relation.

F **The Object Start Location Table**

This table contains one byte per object, which is the start location of the object. The address of the Object Start Location Table is calculated by adding the number of objects to the address in &2508-&2509 (&3508-&3509 in the cassette version).

G **The Movement Table**

This table consists of an entry for every location used in the adventure, and is of the format; one byte, which points to the next entry in the table, the entries and the termination character (255). Each entry is either empty, or consists of a group of 'movement pairs'. A movement pair is a word value from the vocabulary, followed by a location number, and means the word value specified when typed by the player will cause movement to that location. A typical entry could be: `SOUTH 6 EAST 7 ENTER 6 NORTH 5`

NOTE : The Movement pairs contain the word value, not the actual word. When the adventure is being played, it is only the first recognised word (W1) which will cause movement. If any movements are to be performed in the Event or Status tables using the action GOTO then those movements should be removed from the Movement table as they are redundant.

H The Event Table

This table, (together with the Status Table) is the main part of the database and each entry contains two word values followed by a string of commands, which are either conditions or actions. When the adventure is played, if there is an entry in the table which matched the word values in W1 and W2 then the commands within that entry are executed. The order of the entries is in ascending order of the first word value. Entries with the same first word value are stored in the order that they were inserted into the database (i.e. they must be inserted in the order required). An example of the order of the table with word values shown in brackets is as follows:-

```
LOOKUP  (30)(9)
LOOKDWN (30)(10)
LOOK*(30)(255)
GET KEY (100)(16)
GET LAMP(100)(26)
```

Each entry in the table has an overhead of three bytes and each command requires 1, 2 or 3 bytes, depending on the number of parameters.

I The Status Table

This table has exactly the same format as the Event Table. When the adventure is played, the Status Table is scanned between turns and the entries are executed in the order that they appear, irrespective of the word values associated with the entries, which are present to position entries at the required place and/or as a reminder of the entries' purpose.

Detailed Description of The Quill Editor

A The Vocabulary

Words can be inserted or deleted, the synonyms of a word may be displayed or the vocabulary listed.

Option 1: Insert word

Asks for a word, followed by word value in the range 0-254. If the word is not already in the Vocabulary then it is inserted.

Option 2: Delete word

Asks for a word. If the word is in the Vocabulary, but is not used in any of; the Movement, Object Word Relation, Event or Status tables, it will be deleted. (If the word is used in one of the tables, it may be deleted only if there is a synonym of the word in the Vocabulary).

Option 3: Show synonyms

Asks for a word, then lists all the synonyms of this word. (i.e. all the words with the same word value).

Option 4: List Vocabulary

Lists all the words present in the Vocabulary, in order of ascending word value.

Option 5: Copy Vocabulary

Sends a copy of the vocabulary to the printer, if the answer to the prompt "Do you have a printer fitted?" was 'Y', if not then this option just lists the Vocabulary, without sending output to it.

NOTE : Words with a word value of less than 13 are considered directions and those with values greater than 199 when linked to an object allow that object to be worn and removed by the AUTO actions.

B The Messages

Messages may be inserted, edited, viewed or listed.

Option 1: Insert message text

The next available message number is used and a null entry is made for it in the message table. An automatic call is made to the edit routine to allow the user to edit the null entry. There is a limit of 253 messages.

Option 2: Edit message text

The editor asks for a message number, which must be in the range of messages already inserted. If so then the existing text for that message is printed, ready to be edited using the < and > keys. If RETURN is pressed then the amended text is placed into the message table. If ESCAPE is pressed then the editing operation is aborted and the old text remains intact.

NOTE : COPY moves to the next line. However, it cannot follow a space and there is a limit of 18 lines.

Lower case letters and spaces are compressed. Spaces cannot be inserted before a TAB, or next to another space.

There is a limit of 248 characters.

The first 20 messages are the system messages, and although they can be changed, their meaning should not be altered. Message 19 has several purposes; the first character specifies the letter for a positive response ("Y" as default), the second the letter for a negative response ("N" as default) and from the third character on, the prompt for command input.

Option 3: View message text

This allows the user to view a message, but not have to edit it.

Option 4: List Message Table

This options lists all the messages in the Message Table. SHIFT should be pressed to move onto the next page of descriptions.

Option 5: Copy Message Table

As option 4 but if a printer is fitted then a copy is sent to the printer.

C The Locations

This table allows location descriptions to be inserted, edited, viewed and listed.

Option 1: Insert location text

This option inserts a null entry in the Location and Movement tables. From then on it behaves exactly like the Message Table. There is a limit of 253 locations.

Options 2, 3, 4 & 5:

As per Message Table, but acting on the Location Table.

NOTE: The adventure starts at location 0.

D **The Object Descriptions**

This table allows object descriptions to be inserted, edited, viewed and listed.

Option 1: Insert object text

This option inserts null entry in the Object Text Table, a null entry in the object word relation table, and sets the obj to not created. It then behaves exactly like the message table. There is a limit of 253 objects.

Options 2, 3, 4, 5:

as per message table, but acting on the object table.

NOTE: Object 0 is considered a source of light.

E **The Object Word Relations**

This option allows the editing, viewing and listing of the word which is related to an object (for AUTOG, AUTOD, AUTOW and AUTOR).

Option 2: Edit word relation

An objno. is asked for, the entry for that object is printed, and the word related to that object is printed ready to be edited using the cursor and delete keys.
If ESCAPE is pressed, then the entry remains unaltered.
If RETURN is pressed, the word is checked to see if it is in the Vocabulary. If it is, then it becomes the new word relation.

Option 3: View word relation

Behaves as option 2, but does not allow for the editing of the word relation.

Option 4: List word relation table

This option lists all the word relations.

Option 5: Copy word relation table

This option sends a copy of the Word Relation Table to the printer, if one is fitted.

NOTE: A null entry means that AUTOG, AUTOD, AUTOW and AUTOR will not work with that object.
Word values > 199 are followed by GDWR, i.e. all AUTO commands will work with that object.
Word values < 200 are followed by GD, i.e. only AUTOG and AUTOD will work with that object.

F **Object Start Location**

Object start locations (i.e. the location where an object starts the adventure) can be edited, viewed or listed.

Option 2: Edit object start locno.

This option asks for an object number, then displays its start locno. ready to be edited. ESCAPE leaves the entry unchanged, and RETURN checks that the number typed in is valid (i.e. between 0 and the number of locations inserted). Values above 252 have special meanings thus:-

253 - carried
254 - word
255 - not created (default value)

Option 3:

Displays a specific object start location without editing it.

Option 4:

This option lists all the object start locations.

Option 5:

This option sends a copy of the Object Start Location Table to the printer if fitted.

G Movement table

Movement entries can be edited, viewed or listed, and consist of a number of movement pairs (a word, followed by a location no.)

Option 2: Edit movement pairs

A location number is asked for, and if it is valid then the movement entries for that location are displayed ready to be edited using the cursor keys. If ESCAPE is pressed the edit is aborted, and the entry is left unchanged.

If RETURN is pressed the entry is checked to see if it is correct. If not then it will be for one of the following reasons:-

a) A word has been used which is not in the vocabulary,
b) A location number was used which does not exist,
c) The syntax was incorrect i.e. a word followed by a number, or
d) The spaces between the word and number, or movement pairs have been forgotten.

Option 3: View movement entry

This option behaves like option 2, but only displays the entry. It does not allow the entry to be edited.

Option 4: List Movement Table

This option lists all the movement entries present in the movement table.

Option 5: Copy Movement Table

This option behaves like option 4, but also sends a copy to the printer if fitted.

NOTE: A location description (even if it's null) must be present for a location before movements can be performed. Any words in the vocabulary can be used in the movement table, but only word values < 13 will give "I can't go in that direction". When an entry is decoded, the word value is converted into the first word with that word value. This may present a problem if a very long entry consisting 'N O N O.....N O' is inserted when the vocabulary has 'NORT' inserted before 'N' as when the entry is printed out for editing 'NORT' will be printed, causing the entry to over-flow (Delete the NORT before editing if this happens!). This does not present a problem if 'N' is before 'NORT' in the vocabulary.

H The Event Table

Entries can be inserted, edited, viewed or listed and consist of a two word identifier, followed by a string of commands (which can be conditions or actions).

Option 1: Insert Entry

This option inserts an entry into the Event Table, the user is asked for an identifier (Two words separated by a space, which are coded into W1 and W2 if the words are found in the vocabulary, (The second word may be '*' which has a value of 255, and tells the interpreter to carry out this command irrespective of the contents of W2. W1 cannot be '*'). The Event Table is searched and a null entry is placed in the database in the appropriate place (The Event Table entries are stored in ascending order of W1 and W2 contents), and an automatic call is made to the edit routine. If an entry already exists with the same identifier, then the new entry is placed after it.

Option 2: Edit entry

This option first of all asks for the entry identifier, as per the insertion routine, but then

searched for the first entry with the same identifier. This is then displayed for editing. If ESCAPE is pressed then the edit is aborted. If RETURN is pressed then the entry is checked for correct syntax. If the entry is empty, then the entry is deleted from the Event Table. Once the entry has been accepted then the database is amended and the next entry is checked to see if its identifier matches W1 and W2. If it does then the process is repeated until all entries with the required identifier have been edited, or ESCAPE is pressed. To delete an entry, change the entry so that it has no conditions or actions.

Option 3: View entry

This option behaves like option 2 but only displays the entries and does not allow them to be edited.

Option 4: List Event Table

This option lists the Event Table.

Option 5: Copy Event Table

This option sends a copy of the Event Table to the printer if fitted.

NOTE: Entries with the same identifier are stored in the order they were inserted. The Interpreter uses W1 and W2 to decide whether to execute an entry, and will continue with entries with the same identifier unless told to do otherwise. If ESCAPE is pressed when inserting an entry, a null entry remains in the database. This can be removed by editing it. The function keys produce several of the commands at a single keypress. A space must be placed between commands, and any terms that follow them.

I The Status Table

This behaves in exactly the same manner as the Event Table except that the identifier has no effect on the operation of the entries and is used as a comment or to position an entry at a particular position in the table (The position of an entry is very important as they are executed in the order they appear).

NOTE: If an entry is inserted at the beginning of the Status Table consisting only of the command DESC, then the Interpreter will loop continuously. This should be avoided at all costs!

J Load database

This option loads a previously saved database from disc or tape. If BREAK is pressed during this operation, the database needs to be reloaded as it will have become corrupt.

K Save database

This option saves the database to disc or tape, under a filename. If BREAK is pressed during this operation the file on disc or tape will have become corrupt and needs to be resaved.

L Test adventure

This option allows the user to test the adventure without having to first save the database. If diagnostics are requested, then the values of the flags will be displayed. Note that the use of diagnostics will cause the screen to scroll earlier than usual.

NOTE: The only way back to the Editor from the Interpreter is the END command in either the Event or Status tables.

M **Save adventure**

This option is very similar to the Save Database, except that the interpreter is saved along with the database in a form so that it will auto-run by typing '*RUN filename'

NOTE: The saved adventure is not designed to be reloaded into The Quill.

N **Bytes spare**

This option gives the amount of memory remaining for the database, followed by the address of the first free byte. (This address is also given in hex.)

NOTE: If the adventure is to run on the Electron, it is important that the database does not pass location 24576 (&6000).

O **Objects conveyable**

This option allows the user to alter the maximum number of objects that can be carried or worn at any one time. This number can be anything between 0 and 255.

P **Star commands (Disc version only)**

This command allows the user access to all the operating system commands (i.e. '*' commands). The Main Menu can be restarted by pressing RETURN without any text before it.

Editor error messages and their meanings

`DATABASE FULL`	There was insufficient room in the database for what you were attempting
`LIMIT REACHED`	The maximum number of locations/messages/objects has been reached (The maximum is 253. 0-252)
`WORD ALREADY PRESENT`	You are trying to insert a word that is already present in the vocabulary
`WORD NOT PRESENT`	You are trying to delete a word that is not in the vocabulary
`WORD USED ELSEWHERE`	You are trying to delete a word that is used elsewhere in the database and has no synonyms

=== End Of Part 2 ===

Appendix A: Customising The Quill to your needs

By using the STAR and JSR commands, the finished adventure can look very individual and sophisticated, and hide its Quilled origins. Features like user-defined typesets can greatly add to the outside appearance of the adventure. (Of course this does not mean that a bad adventure will be transformed into a good adventure, but the effort is worthwhile.)

By way of example, we will give a routine which eliminates the need to type 'There are exits north, east and west' into every location description. Thus the routine will reduce the amount of memory used by the location description (even with text compression), thus allowing more detailed descriptions and a more complex adventure.

The routine consists of a small section of Machine Code, which will occupy any page of memory (&B00-&BFF) is used in the example) and a small entry in the Status Table.

Type in the following BASIC program (Change line 50 depending on whether you have the cassette or disc version):-

```
 10FOR N%=0 TO 3 STEP 3
 20P%=&B00
 30[
 40 OPT N%
 50.START:LDA &250A:STA &70:LDA &250B:STA &71:LDX #0
 51\ &350A and &350B for Cassette version
 60.START1:CPX &502:BEQ START2:LDY #0:LDA (&70),Y
 70CLC:ADC &70:STA &70:LDA &71:ADC #0:STA &71:INX:JMP START1
 80.START2:BRK
 90EQUB 255:EQUB 13:EQUS "Visible Exits:":EQUB 13:EQUB 0
100LDY #1:STY &72
110.START3:LDY &72:LDA (&70),Y:CMP #255:BNE START0
120JSR &FFE7:JSR &FFE7:RTS
130.START0:CMP #0:BNE START5:BRK
140EQUB 255:EQUS "North,":EQUB 0
150.START4:INC &72:INC &72:JMP START3
160.START5:CMP #1:BNE START6:BRK
170EQUB 255:EQUS "South,":EQUB 0
180JMP START4
190.START6:CMP #2:BNE START7:BRK
200EQUB 255:EQUS "East,":EQUB 0
210JMP START4
220.START7:CMP #3:BNE START8:BRK
230EQUB 255:EQUS "West,":EQUB 0
240JMP START4
250.START8:CMP #4:BNE START9:BRK
260EQUB 255:EQUS "Up,":EQUB 0
270JMP START4
280.START9:CMP #5:BNE START4:BRK
290EQUB 255:EQUS "Down,":EQUB 0
300JMP START4
310]
320NEXT
330PRINT"LSB=";START MOD 256
340PRINT"MSB=";START DIV 256
```

Once this program has been typed in, save it and then run it. If there were no errors then note the values of LSB and MSB and save the resultant code using:

```
*SAVE filename B00 BFF
```

Now run The Quill and select option I (the Status Table) and insert the following entry.

```
N    N    NOTZERO      63
          JSR           0          11
          CLEAR        63
```

This entry makes use of the fact that flag 63 is set to 255 every time a location is described. First it checks if flag 63 is not zero, which it will be if the location has just been described, and then calls the routine at page &B00 (0 11 are the values of LSB and MSB given by the BASIC program if the code is paced at &B00). Once control has returned to The Quill Interpreter, it proceeds by clearing flag 63 and dropping through to the next entry. This is to stop the routine being called every turn.

In order for this routine to be used in a saved adventure a short BASIC routine should he saved with the adventure:-

```
10REM BASIC LOADER FOR SAVED ADVENTURES
20*LOAD filename B00
30*RUN adventure
```

If your saved adventure is to be used from disc then this BASIC loader will have to be placed elsewhere in memory to prevent it from overwriting the Interpreter. (e.g. position PAGE above the top of the database.) Of course the routine could have been loaded using the STAR command, e.g.

```
N    N    STAR         20
```

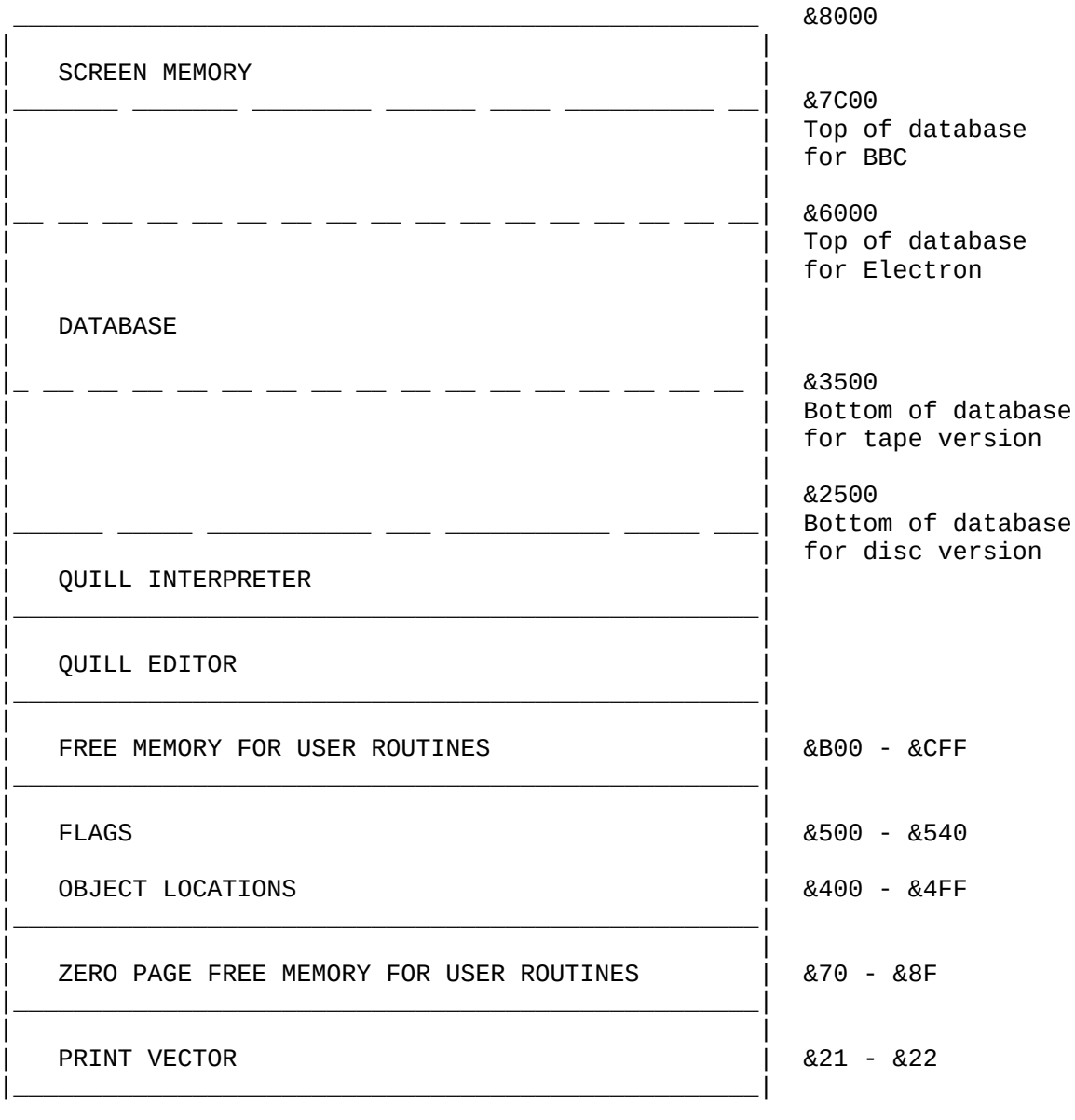
and make message 20 read: `*LOAD filename.`

The only disadvantage of this method is that there is a chance of the file being loaded in every turn and this must be prevented.

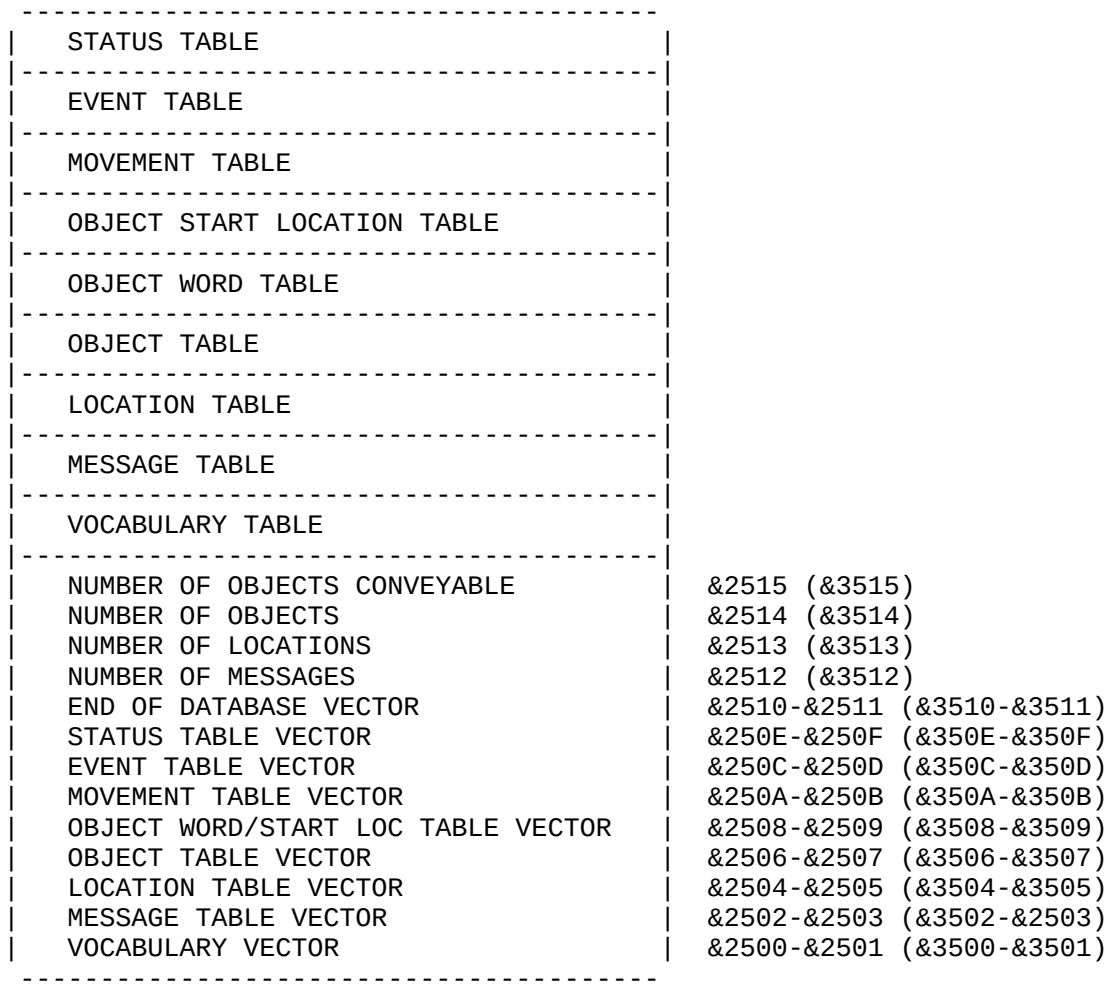
The program reveals several points that the user should note:

1) Locations &70-&8F are available to the user,
2) The Quill interpreter is restarted by RTS, and
3) Messages can be sent using BRK, followed by a 255, the message, and a 0.

Appendix B: The Quill Memory Map



Appendix C: The Quill Database Memory Map



Appendix D: Notes on the disc version of The Quill

The program is stored as several overlays, which occupy pages &1900 to &2500, and the database is constructed from page &2500 onwards, thus giving the user an extra 4K for the database cassette version starts at &3500). Because of the use of overlays, however, the master disc must be in the drive at all times, and a small delay may be noticed when selecting an option on the main menu.

When the database is to be saved or loaded, or the adventure saved on a single drive system, The Quill disc must be replaced by the spare database disc, but as soon as the database has been saved, The Quill disc should be put back into the drive.

Users with double drives should leave The Quill disc in drive 0 and spare database disc in drive 1.

If at any point a disc error is flagged, the user will be informed and the operation should be repeated after correcting the fault. (Note that the database may be corrupt if it occurred during a LOAD).

The disc version of The Quill includes an extra option on the Main Menu (Option P) which allows the user access to all the normal * commands. (The user can return to the Main Menu by pressing RETURN.)

NOTE: Certain commands related to the disc operating system should be avoided at all costs. Notably *COMPACT as it will wipe out the database!

Appendix E: Summary of conditions, Actions and Flags

A The Conditions

ATLT	locno.	ATGT	locno.
NOTAT	locno.	AT	locno.
PRESENT	objno.	ABSENT	objno.
NOTWORN	objno.	WORN	objno.
NOTCARR	objno.	CARRIED	objno.
CREATED	objno.	DESTROYED	objno.
NOTZERO	flagno.	ZERO	flagno.
CHANCE	1-100		
NOTEQ	flagno. no.	EQ	flagno. no.
LT	flagno. no.	GT	flagno. no.

B The Actions

INV		DESC	
DROPALL		DONE	
OK		KEY	
AUTOG		AUTOD	
AUTOW		AUTOR	
SAVE		LOAD	
QUIT		CLS	
END			
MESSAGE	mesno.	MES	mesno.
GET	objno.	DROP	objno.
WEAR	objno.	REMOVE	objno.
CREATE	objno.	DESTROY	objno.
PAUSE	1-100	GOTO	locno.
SET	flagno.	CLEAR	flagno.
PRINT	flagno.	STAR	mesno.
PLUS	flagno. no.	MINUS	flagno. no.
ADD	flag1 flag2	SUB	flag1 flag2
LET	flagno. no.	SWAP	objno1 objno2
PLACE	objno. locno.	JSR	lsb msb
SOUND	pitch duration		

C The Flags

```
flag 0     0   - location dark
           255 - location light          (initial value)
flag 1     no. of objects being carried  (initially 0)
flag 2     current location number       (initially 0)
flag 3  )
......  )
......  )  all flags single byte flags   (initially 0)
......  )
flag 47 )
flag 48 )
......  )  all flags double byte flags   (initially 0)
......  )  PLUS, MINUS, ADD, SUB and PRINT all act as 16 bit calcula-
......  )  tions i.e. all act on this flag, and the next highest flag
flag 62 )
flag 63 )
```


GILSOFT INTERNATIONAL LIMITED
2 Park Crescent, Barry
SOUTH GLAMORGAN CF6 8HD
Tel (01446) 732765