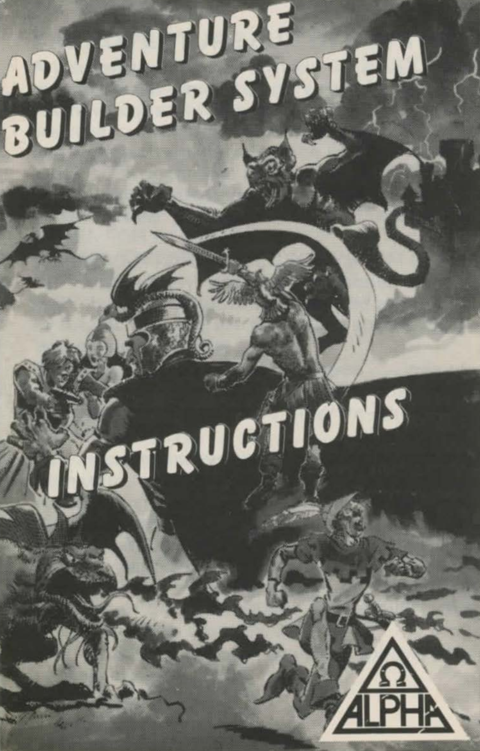
ADVENTURE BUILDER SYSTEM



INSTRUCTIONS

Ω ALPHA

THE ADVENTURE BUILDER SYSTEM

An adventure written entirely in BASIC has reaction times which compare very badly with machine coded programs. The slow sections are readily identifiable as the parsing routine, word recognition through long FOR ... NEXT loops or through large numbers of IF ... THEN lines, and identifying which few objects should appear at a location or in the player's INVENTORY.

The ADVENTURE BUILDER SYSTEM (ABS) can substantially decrease the above delays such that an adventure written predominantly in BASIC can operate at a speed approaching that of machine code.

The ABS consists of two programs . . . one to generate the required CODE with the other being a short BASIC core program which will utilise this CODE.

Before using the system it is advisable for the adventure to be outlined to a reasonably advanced stage. The map should be constructed with each location numbered and all objects well identified as to description and starting location. Details on how to use the system will make reference to the short demonstration adventure for which the DATA and map are provided.

The BASIC adventure program follows the convention of:

a) PRINTs location description (plus graphics, if included).
b) PRINTs any objects visible at this location.
c) Asks "What next?" and awaits input.
d) Accepts input from keyboard and PRINTs on screen.
e) Analyses input (usually into VERB and NOUN).
f) Sends control of program to a sub-routine specific to that VERB.

The CODE GENERATOR program includes the DATA for movement input (GO NORTH or N etc) and also for the commonly used VERBS:

GET (or TAKE)
DROP
LOOK (or L or R)
EXAMINE
QUIT (or STOP)
INVENTORY (or I or LIST)
SAVE
LOAD

METHOD OF OPERATION

1. Prepare a tape for copy of CODE GENERATOR with DATA for the adventure with loader program:

```
10 CLEAR 52000
20 LOAD ""CODE
25 LOAD ""
30 SAVE "loader" LINE 10
```

2. LOAD the CODE GENERATOR program, add your adventure DATA then SAVE on to the above tape by GOTO 9990 (Tape A).

3. Make a copy of the BASIC adventure core program. "VERIFY" as a direct command and do NOT rewind. (Tape B).

4. LOAD Tape A and supply the information required by the on screen prompts then SAVE the DATA on Tape B. Make a note of the CLEAR number.

5. Enter as a direct command: CLEAR number: LOAD "" and LOAD Tape B. Stop the tape when prompted, delete line 9949 then GOTO 9950 and re-start the tape.

6. "BREAK" and alter line 9991 to:

```
9991 SAVE "SYSTEM"CODE save, len
```

7. Add the remainder of the BASIC program (VERB sub-routines and screen display DATA etc).

8. SAVE at the start of a blank tape (by GOTO 30) the following loader:

```
10 CLEAR number
20 LOAD ""
30 SAVE "LOADER"LINE 10
```

9. SAVE the adventure after this loader (by GOTO 9990).

N.B. The enclosed tape contains two copies of each of the following programs:

- DEMONSTRATION ADVENTURE (with BLACK or WHITE background)
- Blank CODE GENERATOR
- Blank BASIC core program
- CODE GENERATOR with the DATA used for the two DEMONSTRATION ADVENTURES.

In order to use the ABS, the DATA for the adventure must be added to the CODE GENERATOR program and the on screen prompts satisfied in the following manner. Responses for the demonstration adventure are indicated after each section heading, and they refer to the version with BLACK background.

A) HOW MANY VERBS (17)

The DATA for these is entered in lines 1ØØ3 to 1999 in the format:

DATA "OPEN",9,"UNLOCK",1Ø

i.e. The VERB (in full), followed by the assigned number. As the verb numbers (vb) Ø to 8 are already assigned the first additional verb takes the value 9.

Synonymous verbs (e.g. SHUT and CLOSE) are allocated the same number but count as separate entries when responding to HOW MANY VERBS.

B) VERB LENGTH CHECK (5)

The ABS offers a choice (between 3 and 7) for the number of leading letters which will be used to determine a match between input and database. The selection of 4 or 5 is usual. If 3 is selected confusion could arise between such inputs as FEED and FEEL, both of which would be considered as FEE.

C) HOW MANY OBJECTS (9)

This refers to the objects which could be included in the INVENTORY of the adventurer.

(For DATA entry details, see section E).

D) MAXIMUM LENGTH OF OBJECT DESCRIPTION (22)

Enter here the number of characters in the longest object description.

E) HOW MANY OBJECTS WHICH CAN HAVE AN ALTERED DESCRIPTION (2)

This refers to items such as "A small TORCH" and "A lit TORCH". The DATA for these objects is entered in lines 2ØØØ 2Ø18 in the following format:

DATA "COAT",1,"A tartan COAT", "A tartan COAT (worn)".

DATA "TORCH",2,"A small TORCH","A lit TORCH".

i.e. The word which would be typed, the assigned number, the original description, the changed description.

The description of objects is held in the string array o (x,y), where x is the total number of objects and y is the character length of the longest description.

These objects which can alter during the course of the adventure must be allocated the first and last numbers of the string array. See list of DATA for demonstration program where "A tartan COAT" and "A small TORCH" are objects 1 and 2 while their corresponding changed versions are objects 8 and 9.

The DATA for the remaining objects (from section C) is entered in lines 2020 to 2188 in the format:

DATA "BOX"3,"A wooden BOX","SPADE", 4,"A short SPADE", . . . etc.

F) NOUN LENGTH CHECK (5)

This is similar to VERB LENGTH CHECK . . . select between 3 and 7. The value can be different from that chosen for VERBS.

G) OTHER WORDS (13)

The first assigned number for these words (DOOR, STAIRS, etc) is calculated as follows:

Total number objects + 7 (i.e. 9+7=16 for demo adventure)

The DATA is entered in lines 2190 to 2988 in the format:

DATA "DOOR",16,"CUPBOARD",17,. . . etc.

H) MAXIMUM INVENTORY (3)

Enter here the maximum number of items which the adventurer is allowed to carry at any one time.

I) NUMBER OF LOCATION (15)

Enter here the total number of locations in the adventure.

J) SCROLL ROUTINE (B)

The ABS contains a split-screen technique which can be used in two ways.

1) If it is required that the top x number of lines remain on screen, then an entry here of C, followed by x, will scroll "behind" these lines when the screen is full.
2) However if the number of lines required to remain on screen is variable (i.e. after HERE YOU CAN SEE section) then this can be achieved by entering B as this response.

If the normal SPECTRUM full screen scroll is required then select here.

K) STARTING LOCATION (7)

Enter here the number of the location at which the adventure commences.

L) MOVEMENT DATA

The DATA to control movement is entered in lines 3000 to 3998 as 7 numbers for each location as follows:

DATA 7,3,13,8,6,0,0

The first number is the location number, followed by the number of the six locations which would be reached by entering N,S,E,W, U,D respectively. An entry of Ø indicates no exit in that direction.
The DATA entered here should take no account of the need to remove obstacles or unlock doors, etc, as such restrictions can be accomodated by the BASIC program, in lines 1ØØ5 to 1Ø89.
See example in demonstration adventure at lines 1ØØ5 and 1Ø1Ø.
A different approach is detailed later.

M) OBJECT STATUS TABLE

Each object will at any one time have one of the following numbers allocated to it:

a) Ø	the object is 'invisible'
b) 99	the object is in the player's INVENTORY
c) Location number. . .	the object currently resides at this location and will appear on screen after HERE YOU CAN SEE when that location is visited.

The DATA for start-up positions is entered in lines 4ØØØ onwards in the format:
DATA 4,1Ø,Ø,1 etc
This indicates that object 1 is at location 4, objects 2 is at location 1Ø etc.

An entry must be made for each object.
These values are then altered as necessary by the BASIC program as the adventure proceeds.

N) SAVE AND LOAD ROUTINES

These routines can be used on a temporary basis effecting a SAVE into and a LOAD from RAM or as a permanent measure as usual, to tape.

O) LOCATION DESCRIPTIONS

The DATA is entered in line 5ØØ1 and onwards in the following format:

```
5ØØ1 DATA 6, "You are in a very t
idy BARN. The farmer who runs thi
s farm is       obviously very prou
d of his work          Or is he?
```

The number immediately after DATA refers to the INK colour for the text following. This number may be kept constant for all locations, but for special emphasis at some key locations a change of presentation can be effectively used as a highlight. The text within the quotes should be carefully constructed with regard to spaces and punctuation.

P) **HERE YOU CAN SEE and INVENTORY**

The phrases for these are entered in lines 6Ø4Ø and 6254 for the former and in 6Ø8Ø and 6254 for the latter, preceded by the PAPER, INK and BRIGHT values for that phrase. e.g.

```
  6Ø4Ø DATA 5, Ø, 1, " Here you can see:-  "
  6254 DATA 5, Ø, 1, " Nothing of interest "
```

The PAPER, INK and BRIGHT values for each OBJECT are entered (in order) in line 614Ø. A set of values MUST be entered for each object. If the choice of PAPER is not the same as the normal screen colour, then add spaces to the HERE YOU CAN SEE etc phrases to match the length of the longest OBJECT description.

Q) **CENTRALISATION OF PRINT**

The printing of HERE YOU CAN SEE and YOU HAVE WITH YOU followed by the relevant list of OBJECTS can be located in the centre of the screen or at the left of the screen... select either Ø or 1 when prompted.

R) **RESPONSES from within the M/C system (1) (6)**

The DATA for these (in numerical order) is entered in line 7Ø9Ø and onwards as follows:

```
  7Ø9Ø DATA 6,"You really need som
  e light in here!"
```

with the number again referring to the INK colour, and the text being arranged for tidy printing to screen.

Three types of responses are considered . . . .
STANDARD . . . in lines 7Ø9Ø to 7Ø98
TERMINAL . . . entered in line 71ØØ onwards
ROUTINE . . . . entered in line 715Ø onwards
but they are numbered in sequence without regard for type.

Hence, in the DATA for the DEMONSTRATION ADVENTURE as there are 9 STANDARD responses, 1 TERMINAL response and 6 ROUTINE responses the reference number for the first ROUTINE response is 11 and for the only TERMINAL response it is 1Ø.

Any response which is identified as a TERMINAL response will automatically be followed on screen by the "Do you want to try again?" phrase.

S) **BRIGHT VALUE (1)**

Enter here the BRIGHT value of the main screen.

T) **PERMANENT BACKGROUND COLOUR (Ø)**

Enter here the colour value of PAPER for main screen.

DEFINED VARIABLES

The following variables have been used in the CODE GENERATOR listing and may prove useful to replace numbers and so obtain the maximum benefit from the program.

Ø=n	1=j	2=a	3=w	4=aa
5=ab	6=ac	7=ad	8=b	9=ae
1Ø=af	11=ag	12=ah	13=ai	14=aj
15=ak	16=c	17=al	18=am	19=an
2Ø=ao	22=ap	23=aq	24=ar	31=as
32=d	33=e	35=f	4Ø=g	42=at
48=au	5Ø=av	54=h	58=i	6Ø=k
61=l	62=m	7Ø=ax	79=ay	92=az
99=bj				
1ØØ=ba	115=bb	126=bc	137=bd	184=be
19Ø=bf	2ØØ=bg	2Ø1=o	2Ø2=p	2Ø5=q
215=u	25Ø=v	251=bi	254=r	255=s
4ØØ=t				

The following sections explain how the system works and how it may be used to write your own adventure.

ANALYSER ROUTINE

The ABS allows a maximum input of 3Ø characters as upper case letters or numbers.

The analyser routine examines the input and returns to BASIC with an allocated number for VERB (vb) and NOUN (no) which the BASIC program can utilise. Any unrecognised word returns a value of 2ØØ for VERB and 2Ø1 for NOUN. This information can be used in various ways by the BASIC program but in the demonstration adventure, either situation produces the all-purpose cop-out of "You can't do that"!!

Possible single word inputs require some explanation. The parser routine analyses the input by considering the first word as the VERB and the last word as the NOUN so that "DROP THE SPADE" or "DROP THE SHORT SPADE" produces the same response as "DROP SPADE".

However, the one word input of, for example, LOOK gives both a VERB and NOUN of "LOOK" with values of vb=4 and no=2Ø1. This would produce a response of "You can't do that" due to the value of no=2Ø1. Such an outcome is prevented by including LOOK in the list of NOUNS and assigning it a default value of 99.

Any additional VERBS which could be used as single word inputs should be included in both VERB and NOUN lists in a similar manner.

Alternatively, the NOUN number could be a true assigned number for use by the BASIC program. In the demonstration program such an approach has been used for DIG where it as been assigned the same NOUN number as HOLE as an input of "DIG" implies "DIG A HOLE".

NUMBER OF LOCATIONS

If your adventure contains more than 65 locations then line 7665 should be altered to 7005+(10xnumber of locations).

CONTROL OF THE ADVENTURE

The adventure is controlled in the BASIC program by the following:

Variable vb (the VERB number)
Variable no (the NOUN number)
The OBJECT STATUS TABLE
The FLAG STATUS TABLE

All other controls are operated by the generated CODE.

a) **OBJECT STATUS TABLE (OST)**

Fifty entries are available in this TABLE and each object has at all times an entry here in the position identified by the number of the object.

e.g. Status of object number x is found by PEEK (o+X) and a change in status is effected by POKE (o+X), Y where Y indicates the new status.

b) **FLAG STATUS TABLE (FST)**

1ØØ entries are provided by this TABLE with a few being reserved for specific uses.

FLAG (f+99) is used to control size of INVENTORY.

FLAG (f+98) is used by the GRAPHICS AID program.

FLAG (f+Ø) contains number of current location.

The first X entries are reserved for use by X objects which change description during the course of the adventure.

All other FLAGS are available for use by the BASIC program.

A FLAG may also be used as a counter to a maximum count value of 255.

A FLAG status is determined by PEEK (f+X) and a change in status is effected by POKE (f+X), Z where Z indicates the new status and X indicates the FLAG number.

As the adventure progresses the BASIC program will be required to update the values in OST and FST.

At start up all FLAGS (except f) have the value of Ø.

e.g. In the demonstration adventure, the FLAGS (f+1) and (f+2) are altered to the value 1 when the COAT is worn and the TORCH is lit respectively, also (f+3) is altered to 1 when the CUPBOARD is open.

COMPLETION OF THE BASIC PROGRAM

1) VERB SUBROUTINES

1ØØ lines are available for each subroutine commencing at lines 1ØØØ + (vb x 1ØØ). If guidance on how to construct these is required then consult paragraph 5 of this section.

2) PRINTING OF RESPONSES

As it is most likely that a split-screen presentation will be selected, it is necessary to check for the need to scroll before each response is printed on screen. To remove the need to include such a check before each PRINT statement the responses MAY be identified as r$(1), r$(2), r$(3), and r$(4) which are then printed by the PRINT routine at line 5Ø.

e.g. . . . . . LET r$(1) = "OK":GOTO 5Ø

It is useful to employ this PRINT routine while testing and de-bugging sections of the adventure. However, the printing of responses can be incorporated into the machine code system by replacing the statement . . . LET r$( ) = "   ":GOTO 5Ø . . . with . . . LET r=x:GOTO 3Ø . . . where x=the response number held in the machine code system (see section R).

Seven responses in the DEMONSTRATION ADVENTURE have been incorporated into the machine code system . . . those in lines 2Ø8, 1Ø92, 11ØØ, 11Ø5, 111Ø, 1915, 7Ø7Ø and 715Ø. The difference in effect can be observed by comparing the response to OPEN CUPBOARD (in FRONT HALL) with DIG HOLE (in the GARDEN).

3) VARIABLES USED BY THE BASIC PROGRAM

Variables assigned in the core BASIC program are:

vb, no, f, o, MAX, SAVE, LEN, DMOV, TRNO, PRIN, PRCAR, PRMS, PRHY, SYS, SA, LO and QUIT.
r$(4,32)

4) LINE NUMBERS

It is important that lines 1ØØ and 195 are retained in the BASIC program as is (except for alteration of the detail in line 1ØØ). These lines are on occasions pointed to by the machine code system and any change could be disasterous!

5) The method of construction of the VERB sub-routines can be observed by studying lines 1ØØØ to 33ØØ of the DEMONSTRATION ADVENTURE. However, for a detailed examination (by way of an example) consider the input of LIGHT TORCH.

This will return a vb value of 12 and no. value of either 2 or 9 depending on whether the TORCH is lit or not (this check will have been automatically made by the code).

For a vb value of 12 control of the program will be directed to line 22ØØ (i.e. 1ØØØ +(12 x 1ØØ) as calculated in line 235).

Translation into ENGLISH of this section is as follows:

Line 22ØØ

IF PEEK (o+9)=99 . . . If lit TORCH is being carried

AND no=9 . . . and NOUN = lit TORCH

THEN LET r$(1) = "It's already lit!" . . . . formulate response and PRINT it.

Line 22Ø5

IF no=2 . . . If NOUN = a small TORCH

AND PEEK f⟨⟩11 . . . location is NOT 11 (Cellar)

AND PEEK (o+2)=99 . . small TORCH is in INVENTORY

THEN POKE (o+2),0 . . . remove TORCH from INVENTORY

POKE (o+9),99 . . . place a lit TORCH in INVENTORY

POKE (f+2),1 . . . set flag for TORCH is lit

LET r$(1)="OK . . . TORCH now lit" . . . formulate response and PRINT it.

Line 2210

Similar to line 2205 but player must be in CELLAR this time, so send program to line 100 to print full description as the player is now carrying a lit TORCH.

Line 2215

Any other situation produces the cop-out response!

An improvement would be to include a response for the situation where the player did not have the TORCH in his/her possession.

VERBS AND NOUNS INCLUDED

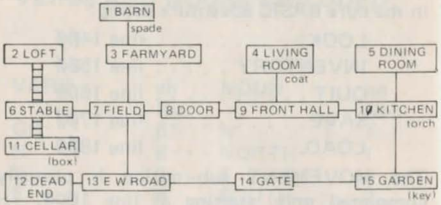
VERB	vb No.	NOUN	no No.
GO	Ø	N	1
N	Ø	NORTH	1
S	Ø	S	2
E	Ø	SOUTH	2
W	Ø	E	3
U	Ø	EAST	3
D	Ø	W	4
GET	1	WEST	4
TAKE	1	U	5
DROP	2	UP	5
EXAMINE	3	D	6
LOOK	4	DOWN	6
L	4	LOOK	99
R	4	L	99
INVENTORY	5	R	99
I	5	INVENTORY	99
LIST	5	I	99
QUIT	6	LIST	99
STOP	6	QUIT	99
SAVE	7	STOP	99
LOAD	8	SAVE	99
		LOAD	99

The following verb sub-routines are included in the core BASIC adventure listing:

LOOK.	line 14ØØ
INVENTORY	line 15ØØ
QUIT	line 16ØØ
SAVE.	line 17ØØ
LOAD.	line 18ØØ

The MOVEMENT sub-routine is partially completed only, starting at line 1ØØØ. It remains to insert any lines to account for temporary blocks such as closed doors or removal of guarding monsters!!

DEMONSTRATION ADVENTURE



NUMBER	OBJECT (o$)	DATA	STARTING LOCATION
1	A tartan Coat	COAT	4
2	A small TORCH	TORCH	1Ø
3	A wooden BOX	BOX	Ø
4	A short SPADE	SPADE	1
5	A bronze KEY	KEY	Ø
6	Splinters of WOOD	WOOD	Ø
7	A large DIAMOND	DIAMOND	Ø
8	A tartan COAT (worn)	COAT	Ø
9	A lit TORCH	TORCH	Ø

NUMBER	OTHER NOUNS	NUMBER	OTHER NOUNS
16	DOOR	23	GARDEN
17	CUPBOARD	24	FIELD
18	LAWN	25	SAFE
19	STAIRS	26	PATCH
2Ø	LADDER	27	JUMP
21	GATE	22	DIG
22	HOLE		

DEMONSTRATION ADVENTURE

vb number	VERB
9	OPEN
1Ø	UNLOCK
11	DIG
12	LIGHT
13	EXTINGUISH
14	CLIMB
15	DESCEND
16	CLOSE
17	TOSS
18	FILL
19	LOCK
2Ø	WEAR
21	REMOVE
14	ASCEND
16	SHUT
22	JUMP
23	USE

MOVEMENT DATA

location	N	S	E	W	U	D
1	Ø	3	Ø	Ø	Ø	Ø
2	Ø	Ø	Ø	Ø	Ø	6
3	1	7	Ø	Ø	Ø	Ø
4	Ø	9	5	Ø	Ø	Ø
5	Ø	1Ø	Ø	4	Ø	Ø
6	Ø	Ø	7	Ø	2	11
7	3	13	8	6	Ø	Ø
8	Ø	Ø	9	7	Ø	Ø
9	4	Ø	1Ø	7	Ø	Ø
1Ø	5	15	Ø	9	Ø	Ø
11	Ø	Ø	Ø	Ø	6	Ø
12	Ø	Ø	13	Ø	Ø	Ø
13	7	Ø	14	12	Ø	Ø
14	Ø	Ø	15	13	Ø	Ø
15	1Ø	Ø	Ø	14	Ø	Ø

FLAG STATUS TABLE . . . . ASSIGNED VALUES

FLAG NUMBER	CONDITION FOR VALE=1
1	COAT worn (fixed allocation)
2	TORCH lit (fixed allocation)
3	CUPBOARD open
4	DOOR open
5	HOLE in field
6	HOLE in garden
7	GATE open
8	SAFE unlocked
9	Found BOX
1Ø	Found KEY
11	First CUPBOARD open
.	
.	
.	
99	INVENTORY (fixed allocation)

OTHER CONSIDERATIONS . . . . RESPONSE TIME AND MEMORY

1) **VARIABLES**

When memory becomes a problem, the first consideration is usually to replace frequently used numbers with a variable. For example Ø and 1 abound in many BASIC adventure listings and the use of:

```
        LET n=Ø
        LET j=1
```

followed by the replacement in the listing of each of these numbers by n or j can effect a considerable saving of memory. Such an approach, when applied to an extreme extent can have an adverse effect on response time. It is therefore necessary to strike a considered balance between speed and memory-saving. When the variables have been finally defined then all lines such as 996Ø can be removed to increase the available memory.

2) **SUB-ROUTINE STRUCTURE**

The length of each sub-routine will affect the response time if the program is required to plough through a vast number of lines before reaching a conclusion. Careful construction can minimise this effect.

a) The "EXAMINE" sub-routine.
This is likely to be one of the longest in terms of number of lines but acceptable response times can be achieved by dividing the sub-routine into several sections.

The EXAMINE command in the first instance can be considered with two groups of objects . . . . those which can be carried (e.g. KEY) and those which are included in the location text (e.g. DOOR). A line such as 13ØØ in the demonstration adventure effectively produces two shorter "EXAMINE" sub-routines, each of which could be further divided if required.

b) The "MOVEMENT" sub-routine
This routine requires to be of almost instant response but can be slowed down if there are many considerations of locked doors etc to be evaluated by the BASIC program. This does not apply to the demonstration adventure as there are only two such lines. Removal of such lines, however, can be achieved by the following modifications.

On return from RANDOMIZE USR 65ØØ5 the maximum value of PEEK 64115 will be the number of locations in the adventure (in this case, 15). Also, if the DOOR IS CLOSED then **that** response is required for an EAST move from location 8 and a WEST move from location 9. Currently in the MOVEMENT DATA, EAST from 8 has a value of 9 and WEST from 9 has a value of 7. Change both of these values to 16 in the DATA lines of the CODE GENERATOR program, or as a short cut the changes can be effected by the direct commands: POKE ((DMOV–7) +59), 16 and POKE ((DMOV–7) +67), 16 . Now make the following alterations to the BASIC listings:

1) Remove lines 1ØØ5, 1816, 7Ø8Ø, 7Ø81

2) Add the following lines:

```
1820  IF  PEEK  (f+4)=1 THEN  POKE
((DMOV–7)+59), 9:POKE ((DMOV–7)+67),7
1825 GOTO 1ØØ
16ØØ  POKE  ((DMOV–7)+59),16:POKE
((DMOV–7)+67,16
```

Renumber 1Ø9Ø as 1Ø5Ø then delete 1Ø9Ø
Renumber 1Ø92 as 1Ø52 then delete 1Ø92

```
1Ø55 IF PEEK 64115<16 THEN GOTO 1Ø6Ø
1Ø58 LET r$(1)=n$(PEEK64115-15):GOTO5Ø
9972 DIM n$(2,9)
9973 LET n$(1)="The DOOR is closed!"
9974 LET n$(2)="The GATE is closed!"
```

3) Alter the following lines:

19ØØ after POKE(f+4),1: add–
```
  POKE ((DMOV–7)+59),9:POKE
  ((DMOV–7)+67),7
```
26ØØ after POKE(f+4),Ø: add–
```
  POKE ((DMOV–7)+59),16:POKE
  ((DMOV–7)+67,16
```

N.B. The calculation ((DMOV–7)+59) locates the position in the MOVEMENT DATA for the entry for an EAST movement from location 8.
The value 59 is found by (location number (8) x 7) + direction value, where N=1, S=2, E=3, W=4, U=5, D=6.
i.e. (8x7)+3=59

As an exercise, similar changes should be made to permit the removal of line 1Ø1Ø.
When the program is complete then all expressions such as ((DMOV–7)+59) should be replaced by their numerical value.

PERSONALISATION

Certain features incorporated into the ADVENTURE BUILDER SYSTEM can be readily altered to provide characteristics of your own choice.

A) STANDARD PHRASES

The STANDARD phrase (e.g."HERE YOU CAN SEE") can simply be altered by changing the DATA in the CODE GENERATOR. The only restriction would be to use a phrase no longer than the longest OBJECT description, for "HERE YOU CAN SEE.. etc, particularly if printing was centralised and/or a change in PAPER colour had been selected.

B) INPUT PROMPT AND BEEPS

By pokeing different values into the following locations it is possible to "personalise" your adventure.

LOCATION	CURRENT	RESULT
6451Ø	62	Prints 〉
64528	42	Prints *
64633	5Ø)	
64634	Ø)	
64636	2ØØ)	
64637	1)	. . Input keyboard BEEP
64Ø49	5)	
64Ø5Ø	1)	
64Ø52	125)	
64Ø53	1)	BEEP for wrong keypress

C) SCREEN PRESENTATION

An adventure with good puzzles and story line can be effectively ruined by poor screen presentation so considerable care should be taken to provide a good impression.

It appears that it has become almost universally acceptable that BORDER and PAPER are set to the same colour. This in turn affects the BRIGHT feature as the use of this provides a poor effect with background colours other than BLACK and to a lesser extent, BLUE. The value of BRIGHT is required by the ABS in order that the printing effected by the machine code system can provide an acceptable presentation.