EAMON DELUXE DUNGEON DESIGNER MANUAL

```
***************** THE WONDERFUL WORLD OF EAMON DELUXE *********************
*                      DUNGEON DESIGNER MANUAL                            *
*                                                                         *
*                        By Frank Kunze, 2002                             *
*                                                                         *
*                       (Update:  27 NOV 2002)                            *
************ ALL NON-COMMERCIAL DISTRIBUTION IS ENCOURAGED *****************
```

This manual has been written for those stalwart people who enjoy role playing games and want to design their own under the Eamon Deluxe system. It goes without saying that it is great to play Eamon adventures, but the real fun lies in designing your own adventures!

It is assumed that you are already familiar with the gaming system and the information included in the Player's Manual. If you want to program special events, you'll need to know the basics of programming in BASIC. Without further ado, lets get on with (what Don Brown called): HOOKING UP WITH EAMON or SENDING ADVENTURERS TO THEIR DEATH FOR FUN AND PROFIT

Contents:

---

1. CHARACTER RECORD DATA

Between adventures, all of the characters that the Eamon Deluxe system knows are stored in the following files in the EAMONDX\MAIN directory:

PLRS.DAT: The number of characters on record PLAYERS.DAT: The characters data. This is a random access file with a length of 200.

The data held in each record is: first a string (LEN=40) containing the character's name. Next is a series of INTEGER values: Character's status (1=OK, 2=Dead, 3=On adventure); hardiness; agility; charisma; the four spell abilities (blast, heal, speed, and power); the five weapon abilities (axe, bow, club, spear, and sword); armor expertise; and sex (0=male, 1=female). Then follow two LONG INTEGER values: Gold pieces carried, and gold in the bank. Then an INTEGER value: Armor class (leather=2, chain=4 plate=6, with one added for a shield). Then four strings (LEN=20) with the character's weapons names. Finally (in INTEGER values) come the four weapons' complexity, type, dice, and sides per die. If a player does not have four weapons, his weapons will be first, and all other weapon names will either be "NONE" or a null string ("").

1.1 WHAT HAPPENS WHEN A PLAYER GOES ON AN ADVENTURE

When a player leaves the Main Hall for an adventure, his character's status is changed to 3 in the PLAYERS.DAT file, then the character's information is transferred to a 1 record random access file called "NEW-MEAT" in the adventure's sub directory (in the same format as the PLAYERS.DAT file of course). The character's status in the NEW-MEAT file is switched to the character's record number in the PLAYERS.DAT file for re-entry when the adventure is finished. Finally, the adventure's INTRO.BAS file is run, which will display the adventure's story line and then run the MAINPGM.BAS file which is the actual adventure.

Once the adventure is over, control is returned to the Main Hall program. If the adventure ended in the character's death, the character's status is changed to dead and the main menu is run again. If the character survives the adventure, his character record is updated with his new items, abilities, etc. and he is sent back to the Main Hall. If you are playing with a character from the Test Bench, no changes are made and you are returned to the Dungeon Designer menu.

1.2 CONTENTS OF THE DUNGEON'S NAME.DAT FILE

The NAME.DAT file is sequential text file that holds all the information about the dungeon and its database. It contains the following values: # rooms, # artifacts, # effects, # monsters, the adventure's name, the number of directions (6 or 10) the player can move in, the Eamon Deluxe system revision number, and the number of dungeons stored in the database (always 1 for new dungeons).

If the number of dungeons is greater than 1, then this data will follow: Each adventure's name, followed by NR, NA, NE, NM, followed by each data type's location (of it's first array) in the database, and finally the number of directions for that particular adventure.

2. THE DUNGEON DESIGN PROGRAMS

Start a design: This initializes all the proper files for a new adventure into one of two special design directories and copies the base dungeon program, intro program, and install programs.

Edit an adventure: This is how you enter/edit all of your dungeon's data.

Test an adventure: You can select from three levels of M/F characters (weak to ridiculously tough) for testing your designs. The Super tough guy is useful for running through the design without worrying about getting killed. The weak, or 'average', character helps you test the scale of your dungeon. Generally speaking, the medium character should be able to survive your dungeon most of the time.

Edit/List hints: Eamon Deluxe allows you to create your own on-line help files that can be called up during the game.

List a design: A nice program which will display all the data of an adventure in a neat order and/or print out a hard copy. Note that Dungeon List supports adventures with multiple adventures in one database (see that section).

2.1 STARTING A NEW ADVENTURE

First, select "Start a new design" from the design menu. You'll have to enter two things; The adventure name, and the number of directions. Most people prefer 6-direction dungeons because they are easier to map. As far as the database goes, ALL Eamon Deluxe adventures are 10-directions. In a 6-direction dungeon, all the design programs will simply ignore the connections for the last 4 directions. The base dungeon (MAINPGM.BAS) will also ignore them, and remove their respective commands during initialization.

Once your adventure has been initialized, take note of which design it is listed as (1 or 2) for future reference. Your adventure will either be in EAMONDX\DGN-1 or EAMONDX\DGN-2 depending upon which design slot you chose. Note that adventures in the design slots can only be played with Test Bench characters.

If you copy all the files from that directory to a floppy disk and then run INSTALL.BAT your adventure will be copied to the master list and can (and definitely should) be removed from the design slot. If any of the main files (INSTALL.BAT, INSTALL.BAS, MAINPGM.BAS and INTRO.BAS) are missing from your design for some reason, they can be found in the EAMONDX\DD\DGNFILES directory.

So, to install your adventure, copy all the files to a floppy disk, change to that disk and type INSTALL. If the installation was successful, go to the test bench and test your adventure in its new location to make sure ALL the files were copied without error. When everything checks out, select 'Remove a design' from the design menu and remove your adventure from the design slot. (You don't want two copies of the same adventure.)

2.2 USING THE DUNGEON EDITOR

You should have your dungeon 90% designed before beginning data entry. Decide what rooms you have, how they connect and what monsters, treasures, and effects are in each room. For each of the four things you can enter (rooms, artifacts, effects and monsters), you can either add a new one to the list, edit one already there, or copy one.

Eamon Deluxe puts no limits on the amount of any type of data. However, because of the special room codes and such, you should never exceed 1000 of any type. Depending on your computer, you will most likely run into memory and/or speed problems if your designs get too large. The more monsters and artifacts you have, (and the older your PC is) the slower the program will run. If you have an amount over 200, you should warn people with "slow" computers (386s and slower) before they begin. As a rule of good measure, adventures requiring more should be broken down into a 'multi-database' type of adventure (see the section on those) so that people with older systems can enjoy them.

2.2.1 ENTERING TEXT

Adding and editing is very similar for all four data types. Every time you enter more than a single key, the entry is done through a special input routine. The following keys perform special functions:

Escape: Accepts all of the text, both before and after the cursor. Enter: Accepts all of the text from the beginning to the cursor. Backspace/Delete: Deletes a character. Tab: Inserts a space. Left, right, up, and down arrows: Move cursor through the text.

The editor has default values set up for all data fields of for rooms, artifacts, effects and monsters. This makes it faster to key in data when you want most of the fields to be default. It also displays helpful messages and charts relating to the type of data at hand.

NOTES: Do not begin artifact names with a number; use 'eight-inch knife' instead of '8-inch knife'. Also, as opposed to the classic Apple version of Eamon, any text character (including quotation marks) may be used in descriptions and names.

When entering ALL descriptions, let text wrap around the screen, DO NOT, I repeat DO NOT add EXTRA spaces -- Text will be separated properly by the base dungeon program.

2.2.2 CHAINING OR LINKING TEXT DESCRIPTIONS

Special programming has been installed for cases where you are entering a description and can't fit it all in the 255 characters allowed. Simply enter as much as will fit (and neatly wrap around the screen), then add an asterisk (*) and a three digit number that points to an Effect (see the section on Effects) where the text is to continue. For example, say you create a monster with the description "You see a rat.*001". MAINPGM.BAS will print "You see a rat.", print a blank line, then print Effect #1 underneath it (making paragraph breaks and such).

If you use two asterisks ("You see a rat.**001"), a line WONT be printed in between the text and the effect, making one large paragraph. You can chain an unlimited amount of text in this manner.

NOTE: For the above process to work, You MUST use three digits. "*1" will be ignored (MAINPGM.BAS checks the 4th to last character for an asterisk, then checks for a second preceding it), Effect #1 is "*001"!

NOTE: When entering ALL descriptions, let text wrap around the screen, DO NOT, I repeat DO NOT add EXTRA spaces -- Text will be separated properly by the base dungeon program.

3. ENTERING ROOMS

All 'rooms' in a dungeon are comprised of 14 pieces of data and stored in the following random access files: ROOMS.DSC (LEN=255), long description of room; ROOMS.DAT (LEN=101), room name, 10 connections for directions (N/S/E/W/U/D/NE/NW/SE/SW), and the level of light in the room (see below).

NOTE: Room names cannot exceed 79 characters.

NOTE: See section 2.2.2 for important notes on entering descriptions.

After the name and description you will have to give the room numbers that you can get to from that room in each direction. A few special codes are reserved by MAINPGM.BAS: A room connection of 0 means you cannot move in that direction, a positive direction (that doesn't exceed the number of rooms) means there is an open connection, -999 is the code to take the adventurer home, and an artifact number + 1000 is a code for a door or gate (1001 being artifact 1, etc.). Negative numbers (except -999) are your codes for whatever special meanings you want to program into MAINPGM.BAS. (See the section on special programming.)

4. ENTERING ARTIFACTS

An 'artifact', with the exception of captives and disguised monsters, is any non-living thing that is in the dungeon.

All 'artifacts' in a dungeon are comprised of 10 pieces of data and stored in the following random access files: ARTIFACT.DSC (LEN=255), long description of artifact; ARTIFACT.DAT (LEN=51), artifact name and its 8 pieces of data (see below)

NOTE: You must have as artifacts, all weapons used by your monsters.

NOTE: See sections 2.2.1-2.2.2 for notes on entering descriptions.

There are twelve pre-programmed types of common artifacts already recognized by the base dungeon program, which can save you a lot of work if you use them properly.

NOTE: Artifact names should be entered in lower-case only. Only use capitals for proper nouns (e.g. "bound troll", "chained Princess Jane").

4.1 ARTIFACT VALUE

When converting all the old adventures I developed a standard of values that is found throughout the Eamon Deluxe system. In the old days, when anarchy reigned in Eamon the same sword could be worth 5 gold pieces in one adventure and 500 in the next! This is because none of the documentation ever really mentioned what an artifact's value should be based around. You can use the following chart to go by, or feel free to disregard it if you wish (note that ALL of the present adventures are more or less based around this system).

Weapons (average)	10-25	Leather armor	75-90
Weapons (magic)*1*3	30-50	Chain armor	100-130
Weapons (exotic)*2	30-100	Plate armor	130-200
Gems/jewels	80-300+	Shields	25-40
Healing potions	50	Clothing*3	1-15
Doors/etc.	0	Lanterns*3	20-30
Messages, notes	0	Food	0
Bodies, captives	0	Wine/liquor/ale	5-100

*1 -- Including magic swords, etc.
*2 -- Stuff like guns and laser weapons from sci-fi scenarios.
*3 -- This assumes its in top condition, otherwise value decreases.

4.2 ARTIFACT WEIGHT

The Eamon Deluxe unit of weight is known as the grond/dos system (why? Because that's what Don Brown named it 20 years ago). I've decided to interpret this as 1 grond = the weight of 10 gold pieces. So an artifact's weight is determined by its weight in gold pieces divided by 10. 100 gold coins weighs 10 gronds, a sword about 5-8, an axe 8-10, a piece of paper 0, and rings and such 0-1. I'm not sure just how realistic this is, but it at least sets a standard. Besides, I've played a considerable amount of adventure games and all of them allow you to lug around way more than you could ever realistically carry.

Use negative numbers for things that allow you to carry more weight (like bags, sacks, etc.) An artifact with a weight of -999 can never be picked up or moved no matter how strong the player. A weight greater than 900 also may never be picked up. The only difference really between -999 and 900+ is -999 produces the "You can't get that" message and 900+ produces a "Don't be absurd." message. Doors & such should have a weight of -999, furniture should be 999.

4.3 LOCATION OF ARTIFACTS

Artifacts normally have room numbers ranging from 1 to the number of rooms in the dungeon, but there are additional codes recognized by the base dungeon program.

-1	Carried by player
-(Monster# - 1)	Carried by Monster #
-999	Worn by player
-(Monster# -1000)	Worn by Monster #
Container Artifact # + 1000	Inside a container artifact
Room # + 2000	'Embedded' in a room description

(If you want monster #1 to have it, it is -2. Monster #2 is -3, etc. If you want monster #1 to wear it, it is -1001.)

4.3.1 EMBEDDED ARTIFACTS

'Embedded' artifacts are those that are not listed in "You see [artifact]" loop initially, but can be acted upon by the player. For example, you could describe a room as containing a statue, and embed a statue artifact. The statue will not be listed as being in the room until it is examined or acted upon by the player. Thus you can hide things from the adventurer.

NOTE: An Embedded artifact MUST be mentioned in the room description or the description of another artifact which is in the room or it will never be found by the player who must use an artifact manipulation command to cause it to actually appear in the room. If the name of the artifact is a little too revealing (such as 'a secret door to the north') then you must use the Synonym routine to let the player look for something else (see section 7.1 for info on using synonyms).

4.4 ARTIFACT TYPES

There are 13 types of recognized artifacts. You are also allowed to enter a value of 14, 15, or 16 for your own special types. All USER types are treated as type 1 by the base dungeon program unless you add your own code. However, I think you'll find most any type you'll need here and it is strongly suggested that you stay within the usual types. The types are as follows:

```
 Type            Format        Type                 Format
 ----------------------        ----------------------------
  0. Gold...............0       9. Edible...............4
  1. Treasure...........0      10. Bound Monster........7
  2. Weapon.............1      11. Wearable.............8
  3. Magic Weapon.......1      12. Disguised Monster....9
  4. Container..........2      13. Dead body............10
  5. Lightable..........3      14. Undefined, User......0
  6. Drinkable..........4      15. Undefined, User......0
  7. Readable...........5      16. Undefined, User......0
  8. Door/Gate..........6
```

All of the formats have the same information in fields 1-4: Value, Type, Weight, and Room #. Format #0 is all USER fields, the rest of the format fields 5-8 contain the following.

```
  Format 1             Format 2               Format 3
5-Weapon Odds        5-Key No.              5-Light Counter
6-Weapon Type        6-Open/Closed(1/0)     6-(USER #6)
7-# of Dice          7-# of items inside    7-(USER #7)
8-# of Dice Sides    8-# items it can hold  8-(USER #8)

  Format 4             Format 5             Format 6
5-# of Heal Points   5-1st Effect         5-Room Beyond
6-# of Drinks/Bites  6-# of Effects       6-Key #
7-Open/Closed(1/0)   7-Open/Closed(1/0)   7-Open/Closed(0/1)
8-(USER #8)          8-(USER #8)          8-Hidden?

  Format 7             Format 8             Format 9
5-Monster #          5-Armor Class        5-Monster #
6-Key #              6-Armor Type         6-1st Effect
7-Guard #            7-(USER #7)          7-# of Effects
8-(USER #8)          8-(USER #8)          8-(USER #8)

  Format 10
5-'Takeable' status*
6-(USER #6)
7-(USER #7)
8-(USER #8)
```

* -- 0=Can't be taken, 1=Can be. Should ALWAYS be 0 unless the body serves some sort of special function (e.g. to collect rewards and such.)

4.4.1 ARTIFACT TYPES: GOLD, TREASURE, AND USER

Gold: Has a set value, like gold pieces.

Treasure: Has a value which will vary depending upon the character's charisma. Base dungeon programs version 4.5 and later won't let the character weasel by with more then 5% greater than the set value, no matter what their charisma.

4.4.2 ARTIFACT TYPES: WEAPONS AND MAGIC WEAPONS

Weapon's type values are 1=Axe, 2=Bow, 3=Club, 4=Spear, 5=Sword. Make all guns and such type 5 (bows), all weapons that the player must lunge and jab with (such as knives and daggers) should be type 4 (spears).

Number of dice and dice sides are used to compute the amount of damage the weapon does in combat. Example: If number of dice = 1 and number of sides = 8 then it will do 1D8 of damage, or from 1 to 8 points of damage. If the two numbers are 12 and 1 (12D1) it'll always do 12 points of damage.

Magic Weapons are magic within the current adventure only. If taken back to the Main Hall, it reverts to an ordinary weapon (type 2). These cannot break in combat which is the extent of their magical power. If you want other magic, you'll have to program it (such as "TrollsFire" in the Beginner's Cave). MAINPGM.BAS classes any weapon brought by the player that has a maximum hit potential (#dice x #sides) of 25 or greater as being magical.

4.4.3 ARTIFACT TYPES: CHESTS AND OTHER CONTAINERS

These can contain other artifacts. Artifacts inside are any with a room # of 1000 + the # of the container. The player can also put items inside and remove items from the container if you allow it. If it has a key number than it is locked and the artifact specified must be used to open it. If the key # is 0 than it is unlocked or doesn't have a key. If a container is locked and has no key (the player must smash it open) code the Open/Closed field with the number of damage hits it must take + 1000. Closed=0, Open=1, 1000+=Smash it to pieces. If it is locked, has no key, and can't be forced open, code the open/closed field as -1.

An artifact can't be PUT inside a container if its weight is greater than the weight of the container. If the weight of the artifact being PUT plus the weight of the other artifacts inside the container is greater than the weight of the container than the player gets the message, "It's full." If you don't want the player to PUT anything in it at all, code field 8 as 0.

NOTE: MAINPGM.BAS versions pre-4.5 used 100+ for the 'amount of hits to open' code.

4.4.4 ARTIFACT TYPES: LANTERNS, LAMPS, AND OTHER SOURCES OF LIGHT

This can be any light source used to illuminate dark rooms. The counter is decremented each turn the player takes and when it reaches zero is extinguished and may not be relit. A counter value of -1 means that it never runs out.

4.4.5 ARTIFACT TYPES: DRINKABLE AND EATABLE

These can do 1 of 3 things: Heal the user by the 'heal amount' specified, hurt or poison the player by having a negative heal value, or do nothing at all. Drinkable and eatable artifacts may only be used the amount of times specified before they are exhausted.

These 2 types are treated as the same, just worded differently. The only exception is that edible artifacts disappear once the player uses up the number of bites. Note: Don't forget to code food as being 'open'.

4.4.6 ARTIFACT TYPES: READABLE

Readable can be read by the player. The base program will automatically read the text from a record in the EFFECT.DSC file that is specified by field 5. You MUST of course add these Effects. Field 6 is included for compatibility with old Eamon systems, but isn't necessary in Eamon Deluxe because you can chain all text with an asterisk (*). See section 2.2.2 on linking text.

NOTE: If you do use field 6 for sequential text, you must make sure the Effects are in the proper order.

NOTE: Even if you link text via the asterisk, field 6 should still be 1.

4.4.7 ARTIFACT TYPES: DOORS

The number of these artifacts + 1000 is put in the room data as the connecting room (see section 3 on ROOMS). Field five tells the room where the player goes when passing through. If field 6 is anything but 0, then the door is locked and the player must have the specified key to open it. If field 7 is zero then the door is unlocked and may be used. If field 7 is greater than 100 then the door can be broken open. If field 6 is -1 then the door can never be opened via the OPEN command (you must add your own special code.

If you want the door to be hidden or secret, make it an embedded artifact (see section 4.3.1 on EMBEDDED ARTIFACTS) and code field 8 as 1. If a door is hidden then the player will get the message, "You can't go that way!" until the door is discovered. The door will also be ignored if it isn't in the room (visibly or embedded) yet.

4.4.8 ARTIFACT TYPES: BOUND/CAPTIVE MONSTERS

These are used to simulate a captive. Field 5 is the actual monster to be freed. If field 6 contains anything but 0, a key is needed. If field 7 is anything but 0, then it is the monster # of a guard (who should be in the room also). Freeing the bound monster causes the artifact to disappear from the room and the actual monster to take its place.

4.4.9 ARTIFACT TYPES: WEARABLE ARTIFACTS

This includes all artifacts that can be worn by the player. "Clothing Type" is not presently implemented by the base program, but the following protocol is used in Eamon Deluxe and should be adhered to.

```
 Armor Class                 Clothing or Armor Type
0. Clothing                 0. Armor, Shields, Plain Clothes
1. Shield                   1. Overclothes (Coats, Capes, etc.)
2. Leather Armor            2. Shoes, Boots
4. Chain Mail Armor         3. Gloves
6. Plate Mail Armor         4. Hats, Headwear
                            5. Jewelry
                            6. Undergarments
```

4.4.10 ARTIFACT TYPES: DISGUISED MONSTERS

These are artifacts that appear to be normal, but are really a monster in disguise. When a player attempts to get or manipulate the object, it disappears from the room and is replaced with the monster specified in field 5. If it's an enemy, it gets one shot to attack the player. If you want some special message to be printed then add the effects and code field 6 with the number of the first effect. Field 7 is included for compatibility with old versions of Eamon. See section 4.4.6 for more information on this obsolete field.

4.4.11 ARTIFACT TYPES: DEAD BODIES

As a general rule of thumb, dead bodies are nothing but trouble and really have no use at all. They waste memory space, slow down the program and make late data entry tricky. The option to use them, however, is open to the designer and MAINPGM.BAS has code already in it to handle dead bodies which can be activated by setting the DB variable. Here's what to do if you want dead bodies:

Once you have your adventure COMPLETELY designed and ALL of your artifacts and monsters entered, select the "Special functions" option on the dungeon editor menu, and then the "Generate bodies" option. Enter "1" for starting monster. A type 13 artifact will then be generated for each monster in order and given the name "(monster name)'s body". Now, at the "Init:" routine in MAINPGM.BAS enter the line "DB = x", where 'x' is the artifact number of the first dead body.

By setting DB (which normally has a value of zero), you activate the dead body routines in MAINPGM.BAS. When a monster dies, their body will be placed in the room. When a player tries to get a body, they'll get the message, "Dead bodies are best if left alone.". And if they attack the body it will disappear and they'll get the message, "You hack the body to bits!".

If you want the player to be able to pick up the dead body, code field 5 of the dead body artifact as 1.

The majority of the old 'classic' Eamon adventures used dead bodies.

5. ENTERING MONSTERS

Monsters are any living (or animate) things in the dungeon. Monsters are treated very similar to characters, however they are assumed to have their full armor expertise and know all weapons equally well.

All 'monsters' in a dungeon are comprised of 15 pieces of data and stored in the following random access files: MONSTERS.DSC (LEN=255), long description of monster; MONSTERS.DAT (LEN=71), monster's name and its 13 pieces of data (see below)

NOTE: See sections 2.2.1-2.2.2 on entering text.

5.1 ENTERING MONSTERS: HARDINESS AND AGILITY

Hardiness is the amount of damage points a monster can take before it dies.

Agility is it ability in combat. A monsters ability to hit is 50% + 2 times the difference in monster agility and armor between the attacker and the defender. For example, assume that the attacker has an agility of 20 and armor of 2, and the defender has an agility of 15 and an armor of 6. The more agile the monster, the greater his ability in battle, and the heavier his armor, the more his agility is compromised. In the above example, the attacker has an effective agility of (20 - 2) = 18 and the defender has an effective agility of (15 - 6) = 9, and the overall odds to hit is 50 + 18 or 68% (18 minus 9 times 2 equals 18). If their roles are reversed the odds to hit are 32% (9 minus 18 times 2 equals -18).

5.2 ENTERING MONSTERS: NUMBER OF MEMBERS AND COURAGE

All monsters in Eamon Deluxe are group monsters, single monsters are just a "group" of 1. This field is very useful. You can have "4 Rats", "12 Orcs", or whatever without having to code more than one of them.

NOTE: You should avoid giving weapons to group monsters, or at least not allow them to drop or break them. It doesn't hurt anything, but can read kind of funny sometimes.

Courage is a monster's tendency to flee from a fight when injured and/or to pursue the player when he flees. Courage of less than 100 is the % chance that a monster will stay and fight. Courage of 200% will always pursue the player and fight to the death. If the monster is injured then courage is reduced by 5%, if the monster is nearly dead then courage is reduced by 10%. If a monster is in perfect health but the 'GROUP' number is less than the original size of the group (some of them died or fled) then the monster is considered injured.

5.3 ENTERING MONSTERS: MONSTER ROOM NUMBERS

There are no special room codes for monsters. Either it's the number of the room in which it will be found or zero if you don't want it around until activated by special programming (such as a disguised monster).

5.4 ENTERING MONSTERS: ARMOR AND WEAPONS

Monster armor is the amount of hits absorbed or stopped per blow and is equivalent to the player armor system. Armor class is deducted from agility during battle (the weight of its armor slows it down).

Monster weapons are the number of the artifact that it is using for a weapon. A value of 0 means that it has natural weapons such as claws. A negative number means that it is unarmed.

Number of dice and number of dice sides is used to compute battle damage by the monster ONLY if it is using natural weapons. If it uses a weapon than the weapon will determine the battle damage. Example: If number of dice = 1 and number of sides = 8 then it will do 1D8 of damage, or from 1 to 8 points of damage. If the two numbers are 2 and 6 (2D6) it will do between 2 and 12.

5.5 ENTERING MONSTERS: FRIENDLINESS AND COMBAT CODE

Monsters in Eamon Deluxe are rather two dimensional. That is, they have little in the way of personality unless you program them to. The following fields give them what personality they do have.

Friendliness ratings are 1=Enemy, 2=Neutral, 3=Friend. Random chance of friendliness is assigned by using the percent chance + 100. Example: 140 indicates a 40% chance for friendliness.

Friendly monsters will smile at the player, follow them around, fight with them, and give them any possessions they carry. Neutral monsters will simply ignore the player.

The other attribute of the monster's personality is a new field that wasn't in old versions of Eamon and can be a very powerful tool. 'Combat Code' values are as follows:

1.......The monster will be expressed as "ATTACKING" rather than using the random combat verbs. (This should be used with monsters such as snakes, green slime, birds, etc. which attack in an unusual way)
0.......The monster will only fight with a weapon (or its natural weapons) This is the standard value for normal monsters.
-1......If there's no weapon around, the monster will use natural weapons
-2......The monster will never fight

5.6 ENTERING MONSTERS: USER FIELDS

USER fields are two reserved data fields included to make your own special programming a bit easier. They don't do anything at all unless you program routines which implement them.

6 ENTERING EFFECTS

'Effects' are stored in a random access file called EFFECT.DSC (LEN=255) and are there for you to use to store extra text, rather the cluttering up the program with it. To read and print your effects during the game use the code: "r = (effect no.) : GOSUB Effect". Where "r" is the number of the effect to be displayed. For examples of this, check out the TrollsFire routines in the Beginner's Cave.

NOTE: See section 2.2.2 on linking more than one effect together.

7 MODIFYING THE BASE DUNGEON PROGRAM

All of the work above was to put your dungeon into a format that the Eamon Deluxe system can use. You can simply play the adventure straight after data entry is over. Select the test option and see how everything works out.

However, if you want special effects such as a magic sword (like TrollsFire in the Beginner's Cave) then you'll have to program them in. Start your adventure using the "Test" option from the design menu, then when the program starts up, hit control-break and add your code. The base dungeon program is called "MAINPGM.BAS" and the intro program is called "INTRO.BAS". Because of the large amount of pre-programmed special artifacts & locations, you can actually write a decent dungeon without programming a line. However, a good dungeon needs a few special events; a GREAT dungeon is nothing BUT specials. The Eamon Deluxe Demo adventure is an example of an adventure with no special programming.

NOTE: If you are adding new code, you should disable the ON ERROR code at the beginning of MAINPGM.BAS. Simply change ON ERROR GOTO... to 'ON ERROR GOTO... When you are done testing make sure to enable it again!

7.1 THE SYNONYM ROUTINE

This is a very simple, yet effective and powerful routine which helps you to hide things. For example, if a "secret door to the east" is embedded in a room, you don't want to give away the name to the player. This is where the synonym checker comes in. You can have your room description say something like "The north wall looks rather smooth compared to the rest of the room." Then, add this line to the "Synonym:" subroutine in MAINPGM.BAS:

```
IF RO = 12 then SY$ = "wall" : SY = 7
```

Where artifact 7 would be the secret door and room #12 would be the location. The secret passage in the Beginner's Cave makes a good example of using the synonym checker.

7.2 ADDING A NEW COMMAND

To add a new command, you must make three changes to MAINPGM.BAS. First, find the DATA statements at the end of the Init routine which hold the command set. Increment the number of commands (it's the number before "north"), then add your new command to the list. Next find the "Com3:" label and add your command label to the end of the "ON C GOTO" stuff. Finally, find the "Err0" label and change "New commands: (None)" to list your new command(s).

If you don't want your new command to be at the end of the list, it will be a bit more tricky. You'll have to modify the "Err0:" routine and the last line of "Com2:" also. See "Err0:" and "Com2:" in appendix 2 for more details.

7.3 USING NEW VARIABLES

When adding your own special routines, you'll probably be needing to use special variables. Let me strongly suggest that you only use the D(x) and D%(x) variables and add no new ones to the MAIN.PGM. D(x) & D%(x) are sets of arrays each from 0-50 which were put in there just for you to use. D%(x) is for INTEGER numbers (-32768 to 32767) and D(x) is for longer, floating point numbers.

My reason for saying this is well-founded: The SAVE and RESTORE commands work by making a 'snapshot' of the dungeons variables in a data file. The 'D' arrays are automatically being saved, but any new variables will have to be added to these routines and could be a hassle for you in the long run. I also have gotten into the practice of keeping track of what each arrays means by adding comments to the beginning of MAINPGM.BAS. That way you'll never forget, and others can better understand your program as well.

NOTE: None of the string variables (room, artifact and monster names, etc.) are saved by the save and restore routines. If you change the name of something during the game, you MUST make sure it gets SAVEd & RESTOREd.

NOTE: Use LINE INPUT rather than INPUT when restoring strings, in case they have any commas or other 'illegal' characters in them.

NOTE: The D%() arrays are INTEGER variables and can't hold a number higher than 32767 or less than -32768.

7.4 BASE PROGRAM SUBROUTINES

You can save yourself a lot of time by utilizing the subroutines already in the base program. You're likely to find most everything you'll need. See Appendix 2 for a complete list.

7.5 TELEPORTING MONSTERS IN/OUT OR CHANGING FRIENDLINESS

Whenever a monster comes, goes, or suddenly becomes a friend/enemy, you must make sure the t() variables are set or things could get screwed up. To avoid problems, simply GOSUB EnemyCheck after all such events. (e.g. m%(12,11) = 3 : GOSUB EnemyCheck)

7.6 DAMAGING A PLAYER/MONSTER

It is not uncommon to have events that cause harm to either the player, a monster, or both. Set the following variables: a (1=armor can absorb some of the damage, 0=armor ignored); d2 (amount of damage); df (monster number of victim, 0=player) and GOSUB MStatus. Example: "df = 0: d2 = 5: a = 0: GOSUB Mstatus" will do five points of damage to the player, ignoring his/her armor.

7.7 SPECIAL EVENT DELAYS

Because all PCs run at different speeds these days, all pauses and timed effects require different delay values to work properly. Eamon Deluxe provides its own variables to make it simple for you. The SETTINGS.DAT file is written to all adventures when being played or tested. This small file holds 13 variables (DLY#, CC1#, CC2#, CC3# and 10 numbers that describe the current system settings of your computer). 'DLY#' is the number of times to repeat a FOR/NEXT loop to delay for about one second.

Examples: FOR second# = 1 to dly# : NEXT FOR halfsecond# = 1 to dly# / 2 : NEXT

I anticipate PCs in the future being even more incredibly fast than they already are these days, that's why I made the values double-precision (#) variables. Always use double-precision variables in your delay loops (e.g. "FOR X# = 1 TO..." instead of "FOR X = 1 TO...").

CC1#-CC3# are delay values for the standard twirly-cursor input routines. The values are set to your computer's speed when you first install Eamon Deluxe and can be reset via the main menu's Control Panel option. You MUST base any delays in your programs on this variable or they will be too fast/slow on different computers. For an example, check out the USE SHOVEL code in The Lair of the Minotaur (adventure #2, The Don Brown Adventures) or the Delay SUBS in the EDXDEMO.BAS file (found in the EAMONDX\MAIN directory).

8. USING DUNGEON LIST

Once you have put all of your data into your files, you will probably want to see what you entered, to catch errors and get a good overview. Select "List a dungeon" from the design menu, it is a program that will list all of your data in a simple, organized fashion. It can also print this information on your printer. Dungeon list is also useful for tearing apart other people's designs. Note that most official Eamon Deluxe adventures have several adventures compressed into one databases. Dungeon list will let you select which one to list.

9. USING THE INTRO PROGRAM

"INTRO.BAS" is the first program which is run by the Main Hall. All you have to do is add your intro story after the "Intro:" label. INTRO.BAS automatically puts the name of the character in the "name$" variable, the character's sex (0=male,1=female) into the SEX variable, and a quotation mark in the q$ variable for you to use. You can refer to the intro to "Lair of the Minotaur" on The Donald Brown Adventures which makes good use of the SEX variable.

You need to change the variables "adventure$" and "author$" at the beginning to your name and the adventure name. Additionally, INTRO.BAS has some useful subroutines for you to use. "Title:" clears the screen and displays the adventure and author names, "KeyWait:" prompts the user for a keypress then goes to title, "Center:" centers and prints the variable "a$" on the 80-column screen.

10. HAVING MULTIPLE ADVENTURES IN ONE DATABASE

You'll notice that the first 20 or so Eamon Deluxe 'adventures' are really collections of up to 20+ 'classic' adventures. MAINPGM.BAS contains code to handle these 'multiple adventures', it just isn't implemented until you activate it. Actually, all Eamon Deluxe adventures are multiples, new adventures are just 'groups' of one. Making a multiple is fairly easy, but is DEFINITELY for experienced designers.

Here's how. First, create each adventure as a separate adventure. Then when each is completely done, use the append program from the Eamon Deluxe utilities menu. This will put all the rooms, artifacts, effects, and monsters at the end of the files of the first adventure and tell you the proper adjustments. IMPORTANT: Write down the numbers that the append program gives you when it is done.

Now add a line under the line in MAINPGM.BAS that loads nr, naf, ne, and nm from NAME.DAT that re-sets them to what they should be (they'll be loaded as the sum of all the adventures otherwise). Then change "adj1=0: adj2=0: adj3=0: adj4=0" to the numbers that the append program gave you.

If you are using separate sets of on-line hints, you can further modify MAINPGM.BAS to list only the ones you want. Simply find the "Hints:" label and change "IF nh > 1 THEN a = 2: m = nh" to "a = [first hint]: m = [last hint]". e.g. "a = 3: m = 6" will only list hints 3 thru 6. Note that hint #1 ("General Help") is always listed.

That's it!

NOTE: You want to make sure that the adventures are completely finished before you merge them because you'll never be able to add new data to the previous adventure again (although you can edit existing data).

14. KNOWN FLAWS IN THE EAMON DELUXE SYSTEM

The on-line hint editor (HINTEDIT.BAS) has goofed up on me a couple of times when adding the 6th or 7th hint. I haven't been able to locate the problem yet. Any help would be appreciated.

NEVER, EVER USE DIRECTORIES WITHIN YOUR ADVENTURES! Eamon Deluxe will crash if any adventure directory contains a sub-directory and you try to install it, remove it, etc.

15. FINAL NOTES ON DESIGNS

You can learn a lot by tearing apart other people's dungeons and seeing how they did things. More often than not you can save yourself a lot of time writing and debugging by using a routine from a different dungeon. I can't think of a greater honor than somebody thinking a routine of mine good enough to 'steal'. And with well over 200 Eamon adventures in existence, it is safe to assume that someone may have already had the same ideas as you.

People really hate traps of the "Zap! You're dead!" type. Try to avoid loading your adventures with instant death traps. A better idea is a trap which simply does damage to the player. Killing the player off without warning generally takes some of his interest out of the game. Note that this was a large part of early Eamons (and most commercial text adventures) and many of their Eamon Deluxe counterparts still have them.

Try to have some point to your adventure. Free a princess, steal a magic sword, slay an arch-fiend, anything. Themes like "You saw a cave and decided to explore it" get boring fast.

Many people have griped over the years about technological barriers being completely ignored by designers. In more simple terms, they hate "walking down the corridor of a medieval castle in full plate armor and being suddenly mowed down by a Nazi with an uzi." I'm not bothered by this myself, Eamon supposedly being a strange magical world where anything goes, but a lot of people feel otherwise. Just a point to consider.

Don't feel at all limited to what's been done already. The beauty of a computer-based role playing system is that is completely open-ended. If you work hard enough, most anything can be done. Monsters that can hold complete and intelligent conversations? Not impossible, you'd just need a good parser and a large database.

Try and use known characters in the right context. If Hokas Tokas from the Main Hall is in your adventure, he should act like he normally does. Feel free to expand his character of course, just be reasonable. The player will see these guys as soon as he returns to the Main Hall so they can't become arch-villains or anything. The player should also not be allowed to attack them and they should never "die" (have them "vanish in a puff of smoke!" instead). If you use characters from other adventures try and make them act and look like the original author did.

Last but not least, don't be afraid to break any of these suggestions. If you truly believe that your dungeon will be better, do anything you please. The worst that can happen is that other people may not like it. The only thing that truly matters in YOUR dungeon is that it is just the way you want it.

16. DEDICATION & CREDITS

Eamon Deluxe hereby officially belongs to the Public Domain, and anybody who is interested may have it for any non-commercial use they wish free of charge and copyright. All I really ask is that you enjoy it.

Frank Kunze December, 2000

The original Eamon system was designed by Donald Brown for the Apple II. It was supported and revised by John Nelson, then redesigned and made more efficient by Thomas Zuchowski. Eamon Deluxe is a completely new version for the Windows/MS-DOS environment written by Frank Kunze and is significantly different from the previous versions. Most of the data structures and standards were written by Don Brown and Tom Zuchowski.

SPECIAL NOTE: Don Brown, the "Father of Eamon" can be found on-line, but is not interested in any sort of Eamon-related correspondence. He hasn't answered anybody's mail (electric or snail) in over fifteen years! Please don't try to contact him as he wishes to be left alone and won't answer you anyway.

APPENDIX 1: BASE PROGRAM VARIABLE LIST

This is a list of most of the variables used by the base program. I tried to keep the names short to save memory and reduce the program bulkiness. All of the variables in MAINPGM.BAS v4+ are of the "SHARED" type and will be passed to any SUB.

```
A%(x,y): Artifact data: x is the artifact number, y is:
     0='Seen' flag
     1=Value
     2=Type
     3=Weight
     4=Room
     5 to 8: See section 4.4 on ARTIFACT TYPES
     AC: Player armor class value
     AE: Player armor expertise
     AR: Artifact # of armor worn
 ANAME$: Name of adventure
  A$(x): Artifact names
   BANK: Players gold in bank
BV$(x,y): Battle verbs for different weapon types
      C: Number of command given by player
     CH: Player charisma
  C$(x): Valid commands
    CZ$: Last command given
   D(x): User variables for the designer.  1-50
  D%(x): User integer variables for the designer.  1-50
     D2: Damage to defender in battle
     DB: Number of first dead body artifact (if used)
     DF: Monster # of defender in battle (0=player)
```

```
      DIE: Death flag (1 or greater = player died)
     DLY#: # of times it takes your computer to delay 1 sec. in FOR/NEXT loop
       EA: Armor factor
        F: 'Found' flag used by search routines
       F1: 'Found more than 1 match' flag used by art. search routine
     GOLD: Players gold on hand
      HIT: Hit flag in combat
        L: Line counter for screen pause
       LS: Artifact # of current light source
       LT: Light level of room (includes artificial light)
  M%(x,y): monster data: x is monster #, y is:
       0='Seen' Flag
       1=Hardiness
       2=Agility
       3=No. of member in group
       4=Courage
       5=Room
       6=Combat Code (See section 5.5)
       7=Armor
       8=Weapon #
       9=# Dice
      10=# Dice sides
      11=Friendliness Rating
      12=Original size of group
      13=Damage taken
      14=USER #14
      15=USER #15
       M9: End of combat round flag for GetWep routine
       MC: Counter for group monsters in battle
      MR%: Monster morale
    M$(x): Monster names
       NA: No. of artifacts (includes player weapons & armor)
      NAF: No. of artifacts (not including player weapons & armor)
       NC: No. of commands
       NE: No. of effects
       NH: No. of hints in help file
       NL: Natural light level of room
       NM: No. of monsters
       NR: No. of rooms
       NW: No. of weapons
     ODDS: Odds for monster OF to hit monster DF in combat
       OF: Attacker (Offender) in combat
       Q$: Holds a quotation mark (CHR$(34)) for printing stuff in quotes
       R2: Room being moved to
       R3: Room just exited
  R%(x,y): Room connections, x is room #, y is connections. x,0 is 'seen'
           flag, x,1-10 are N/S/E/W/U/D/NE/NW/SE/SW, x,11 is light level
      REC: Player record # in character file
       RL: 'Dice roll', a random no. (usually) from 1-100
       RO: Room no. player is currently in
    R$(x): Room names
   RB$(x): Battle response verbs
       SH: Artifact # of shield being worn
       SL: String length (usually for S$ variable)
   SA%(x): Current spell abilities (1=Blast, 2=Heal, 3=Speed, 4=Power)
   SP%(x): Total spell abilities   (1=Blast, 2=Heal, 3=Speed, 4=Power)
      SEX: The player's sex (0=male, 1=female)
    SPEED: Counter for speed spell
      SUC: Spell successful
       V$: Verb of command
       S$: Object of player command
      S2$: Second object for some commands (GIVE, REQUEST, PUT, etc.)
   SM$(x): Smile verbs
       SY: Artifact # of synonym match
    SY$: Synonym
   T(x): Hardiness of each side in combat
     TA: How player is attacking an item (0=ATTACK, 1=BLAST)
     TP: Value of loot (at exit)
     UP: Logical flag if weapon ability increased
      W: Artifact # of attacker's weapon in combat (0=Natural weapons)
     W2: Type of weapon if OF is player
 WA%(x): Players weapon abilities
 WP%(x): Weapon pointers (exiting program)
     WT: Weight player is carrying
```

The variables adj1, adj2, adj3, and adj4 are used in adventures that have multiple adventures in their databases. (See section 10)

APPENDIX 2: BASE PROGRAM LABELS

The v4.0 base program (MAINPGM.BAS) has everything except the init routines and the final exit routine in the MAIN SUB to preserve space and allow larger additions of special programming. Earlier versions of MAINPGM.BAS have all the code in the main program. All of the variables in v4+ are of the "SHARED" type and will be passed to any SUB.

Main subroutines (access with GOSUB)

```
    Main subroutines (access with GOSUB)

Effect: Reads record # R from the EFFECT.DSC file, goes to DiskRead4
Effect1: Reads record # R from the EFFECT.DSC file, changes text color to
         green, GOSUBs to DiskRead4, changes text color to blue.
DiskRead1: Reads record # R from the ROOMS.DSC file, goes to DiskRead4
DiskRead2: Reads record # R from the ARTIFACT.DSC file, goes to DiskRead4
DiskRead3: Reads record # R from the MONSTERS.DSC file, goes to DiskRead4
DiskRead4: Sends a$ to Lp0 to be printed on the screen, if the end of a$
           contains a "*" and a 3-digit number, R is set to the number and
           it goes to Effect; if not it GOSUBs to Lp1 and returns
DE: 'DiskRead Error', one of the above was asked to read a record number
    that exceeds the database limits.
Lp0: Prints the variable a$ without wrap-around on the screen
Lp1: Does a PRINT and increments line counter
Lp2: Increments line counter
Lp3: Adds three to line counter, does a PRINT
Lp4: Adds two to line counter, does a PRINT
Lp5: Checks to see if line counter (L) is less than 20, if not falls into Lp6
Lp6: "[MORE]" line pause, resets L to zero, erases "[MORE]" text
Lp7: Clears screen, sets line counter to zero
Lp10: GOSUBs Lp0, goes to Lp1
Lp11: GOSUBs DiskRead, removes extra line printed afterwards

Smile1: Smile subroutine, prints smile response for monster in M variable

AF: Prints "Attack non-enemy?" and falls into AF1
AF1: Displays " (Y/N):" prompt, returns with "y" or "n" in S$ variable

Synonym: Sets synonym variables, calls Synonym1
Synonym1: Checks to see if player input matches SY$ variable, if so player
          input is changed to the name of artifact specified by SY variable

EnemyCheck: Sets up T(x) variables.  T(1) is 'enemies in room' flag
            If monster has a random friendliness chance then friendliness is
            determined
MoveMons: Moves monsters who follow from room R3 to room RO
```

```
SpellCast: Checks to see if player successfully cast spell specified in S, if
           not "Nothing happened." is printed
RollDice: Gets a roll from 1-100, returns with roll in RL variable
RollDice2: Gets a roll from 1-RL, returns with roll in RL variable

CheckSub: Prompts player for a subject if they didn't give one, returns with
          S$ in lower-case
CheckSub2: Prompts player for a second subject if they didn't give one,
           returns with S2$ in lower-case, displays a$ as prompt

ItemTake: Prints "(Taking it first)" and falls into ItemTake1
ItemTake1: GOSUBs ItemGet3

WTRem: Removes weight of artifact #a IF it is carried by player

Caps0: Sets variable a$ to a$(a), falls into Caps1
Caps1: Capitalizes leftmost letter of a$ and adds a space to the end

Pu2: Makes a$ monster #m's name and adds 's or ' depending upon last letter
     of name and adds space to the end (e.g. "Bob's ", "Brutis' ")

                  Main routines (access with GOTO)

Lp8: GOSUBS Lp10, goes to Main
Lp9: GOSUBS Lp10, goes to Pfoe
Lp12: GOSUBS Lp4, goes to Pfoe

Err0: Invalid command message, valid command list
Err1: "You aren't carrying it."
Err2: "You must open it."
Err3: "You can't (command) (subject)."
Err4: "Nobody here by that name."

                     Search routines (access with GOSUB)

Msearch: Checks for monster name in room that matches S$ variable, if no
         match is found F = 0, if match is found F = 1 and M = monster #
Msearch0: Alternate entry, you must set WH and HA to the desired location
          of monsters for search
Asearch: Search for artifacts carried by player only
Asearch0: Search for artifacts carried, in room or embedded in room
Asearch1: Search for artifact in room, carried and whatever you set HA to
Asearch2: Search for whatever you set HA, WH, and EM to

                    Combat subroutines

GetWep: Monsters who don't have a weapon look for one in room to pick up

Battle: Combat routine, monster OF attacks monster DF
Battle1: Checks if players weapon ability and AE increased
Battle2 and Battle3: A hit!
Damage0: Calculates damage: D=# of dice, S=#of sides, A=1 means monsters
         armor value is subtracted from damage.  A=0 ignores armor
Damage1 and Mstatus: Adds D2 worth of damage to monster
Ms1-6: Prints monster's health condition

Mdead: Monster dies routine, if monster is player then DIE is set to 1
Mdead0: Special effects of monsters death, adds body if bodies are used

                  Main labels

Main0: Does a GOSUB to Lp4 before falling into Main
Main: The main loop, prints room, artifact, monster names, and descriptions
```

```
Main3B: Prints the room name
Main4A: Lists monsters in room
Main5A: Lists artifacts in room
Com0 & Com1: Get player's command

(In between Com1 and Com2, the command is parsed into V$, S$, and S2$.  V$
is the 'verb' or actual command, S$ is the subject.  S2$ is the second half
of S$ if S$ contains " to ", " on ", " in ", "at ", " from ", or " with ".
e.g. "GIVE GUN TO COWBOY" becomes: V$="give", S$="gun", S2$="cowboy".
"READY BALL AND CHAIN" becomes: V$="ready", S$="ball and chain", S2$="".)

Com2: Searches for a match between V$ and the command set, branching to Err0
      if none is found
Com3: Branches to the command given

Pfoe: Commands return to here.  If there are enemies in the room then combat
      routines are run through.

Pfoe3: Special stuff you want to have happen during every round.  Right now
       all that happens is monsters without weapons will look for them.

Move: The directional commands (except "FLEE") branch here
MoveCheck: Checks for special moves (like -999). Add your special moves here
Move1: Handles doors/gates
Move2: Moves the player to the next room (specified in the R2 variable),
       GOSUBS to EnemyCheck & MoveMons, and sets the level of light in the
       new room.  NX variable is set to the number of normal exits in the
       new room (not counting doors or negative moves)

Flee: The flee command
Flee1: This subroutine picks a random valid direction for fleeing and puts it
       in the R2 variable

ItemClose: Close command
Drink: Drink/Eat commands
Drop: Drop command
Drop1: "Drop All" routine
Drop2: The drop command subroutine
Examine: Examine command
ItemGet: The "Get" command
ItemGet2: "Get All"
ItemGet3: The GET subroutine, tells you whether or not you can get the item,
          gets it if it's takeable
Light: Light command
ItemOpen: Open command, branches to DoorOpen for door/gates
DoorOpen: Opens doors
ItemPut: Put command
ItemRead: Read command
Ready: Ready command
Remove: Remove command, branches to remove1 if item is not worn
Remove1: Remove item from container
Use: Use command.  Currently does nothing
Wear: Wear command
Wear2: "You're already wearing one" message
Attack: Attack command, branches to AttackMon if attacking monster
AttackItem: Beat down doors, break open chests
AttackMon: Player attacks a monster
Free: Free command
Give: Give command, branches to GiveMoney if S$ is a number
GiveMoney: Pay gold to a monster
Give1: Gives an item to a monster, if its food/drink it branches to GiveDrink
GiveDrink: Monster takes a drink/bite and hands given item back
Request: Request command
Smile: Smile command
```

```
Smile1: Smile subroutine, prints smile response for monster in M variable
Inventory: Inventory command
Inv1: Displays what player is wearing
Inv3: Display container contents subroutine, prints "closed" if it's closed
Inv4: Alt. entry for Inv3, prints nothing if container is closed
Status: Status command
Look: Look command
Say: Say command
Blast: Blast command
Heal: Heal command
Speed: Speed command
Power: Power command
Hints: Hints command
GameSave: Save command
GameSave1: Actually save the game
GameRestore: Restore command
Quit: Quit command

EndGame: If player survived, asks them if they want to leave adventure
Endgame0: End of game routine, branches to Dead if DIE variable is 1 or more
SellWep: Sell extra weapons routine, S$ holds the name of the guy who says
         "you have too many weapons" for easy changing
EndGame1: Sell treasure routine, S$ holds the name of the buyer

ExitRtn: Kills the NEW-MEAT and saved game files, resets the dungeon's status
         to empty (no character in it), returns updated character data to
         PLAYERS.DAT file and sends the character to the Main Hall
Dead: Menu of options for dead characters
```

APPENDIX 3: EAMON DELUXE FILES AND DIRECTORIES

```
                      EAMONDX:\
ADVENTRS ..............Contains the names of installed adventures
EAMON.BAT .............Startup program, calls QBASIC and runs STARTUP.BAS
EAMONDX.BAS ...........The main menu.  Most stuff returns to here
EDXEXIT.BAS ...........Displays exit message and tries to return to
                       DOS/Windows.
SETTINGS.DAT ..........Holds the current system settings (see below)
STARTUP.BAS ...........Runs EAMONDX.BAS.  Included for compatibility with
                       very old versions of Eamon Deluxe.

                      EAMONDX\MAIN\
CHAREDIT.BAS ..........The character editor
EDXDEMO.BAS ...........The intro graphics routines
FRNTDESK.BAS ..........The front desk and new characters routines
HALL-CGA.SHP ..........Holds the shape set for the CGA Graphic Main Hall
HALL-EGA.SHP ..........Holds the shape set for the EGA Graphic Main Hall
MAINHALL.BAS ..........The Main Hall program (both text and graphic)
PLAYER ................The record # of the player entering the Main Hall
PLRS.DAT ..............The number of characters on file
PLAYERS.DAT ...........The file which holds all the character data (LEN=200)
SETTINGS.DAT ..........Holds the current system settings (see below)

                      EAMONDX\DD\
DDMENU.BAS ............The Dungeon Designer menu
DGNDIR.DAT ............The status of the 2 design slots (1=Used,0=Empty)
DGNMAP.BAS ............The room connection mapping program
EDIT.BAS ..............The dungeon editor program
HINTEDIT.BAS ..........The hint files editor
LIST.BAS ..............The dungeon list program
SETTINGS.DAT ..........Holds the current system settings (see below)
TEST.BAS ..............The test bench
```

```
                      EAMONDX\DD\DGNFILES\
INTRO.BAS .............The first adventure program run by the Main Hall
MAINPGM.BAS ...........The base dungeon program
INSTALL.BAT ...........Copies adventure files from floppy disk to
                       C:\EAMONDX\NEWADV
INSTALL.BAS ...........Renames EAMONDX\NEWADV to a proper name and adds
                       adventure name to list.  (ADVENTRS file)

                      EAMONDX\E001
ADV.DAT ...............Holds the number of the adventure being played
ALTCAVE.BAS ...........The Alternate Beginner's Cave base program
ARTIFACT.DAT ..........Contains artifact names and data
ARTIFACT.DSC ..........Contains artifact descriptions
BEGCAVE.BAS ...........Beginner's Cave/Enhanced Beginner's Cave base program
CAVEII.BAS ............The Beginner's Cave II base program
DEMOADV.BAS ...........Eamon Deluxe Demo Adventure base program
EFFECT.DSC ............Contains effect text
FOREST.BAS ............The Beginner's forest base program
GAMEDIR.DAT ...........Contains # of saved games and descriptions of saves
HINTDIR.DAT ...........Contains the # of hints, how many paragraphs, etc.
HINTS.DSC .............Contains the text for the hints
INTRO.BAS .............The intro
MONSTERS.DAT ..........Contains monsters names and data
MONSTERS.DSC ..........Contains monster descriptions
NAME.DAT ..............Contains adv. name, # of rooms, art, etc.
ROOMS.DAT .............Contains room names, connections and light level
ROOMS.DSC .............Contains room descriptions
SETTINGS.DAT ..........Holds the current system settings (see below)
```

All adventures are stored in the same files and format as E001, except the base dungeon program will be called 'MAINPGM.BAS' if it's a 'single' adventure. All installed adventures will be in a directory with 'E' and the number of the adventure in the ADVENTRS file (E004, E005, E126, etc.).

EAMONDX\DGN-1\ and EAMONDX\DGN-2\ are the two Dungeon Design adventure slots. Same format as E001

EAMONDX\MANUALS\ is the directory which holds all of the documentation in 'TXT' file format. I've also included my Microsoft Works database file (EAMONDX.WDB) for those who want it.

SETTINGS.DAT (A file found in almost all directories): This small file holds 13 variables (DLY#, CC1#, CC2#, CC3# and 10 numbers that describe the current system settings of your computer). 'DLY#' is the number of times to repeat a FOR/NEXT loop to delay for about one second.

CC1#-CC3# are delay values for the standard twirly-cursor input routines. The values are set to your computer's speed when you first install Eamon Deluxe and can be reset via the main menu's Control Panel option. You MUST base any delays in your programs on this variable or they will be too fast/slow on different computers. For an example, check out the USE SHOVEL code in The Lair of the Minotaur (adventure #2, The Don Brown Adventures) or the Delay SUBS in the EDXDEMO.BAS file (found in the EAMONDX\MAIN directory).

Presently only the first three system settings values are used. 1 is the preferred graphics mode (equivalent number for the SCREEN command), 2 is the preferred

Main Hall (1=Text, 2=Graphics), and 3 is the number of the preferred COLOR of the Text Main Hall's text (1-15).

SETTINGS.DAT is written to nearly every directory in the Eamon Deluxe system.

APPENDIX 4: AVERAGE MONSTER RATINGS

The following was compiled to give you something to work from when designing a dungeon for the Eamon Deluxe system and to try and set a standard. Please don't feel at all constrained by these ratings.

A monsters ability in combat is determined by his agility, with his armor slowing him down. His hardiness is the amount of damage points he can take before he dies.

Weak monsters: This includes small creatures like rats and kobolds. Medium monsters: Including thugs, orcs, goblins. Tough monsters: Including giants, skilled warriors, trolls. Super tough: Such as dragons and demons.

	Weak	Medium	Tough	Super tough
Hardiness:	2-8	9-15	16-30	31-90
Agility :	5-9	10-16	17-24	25-35
Courage :	80-90%	95-100%	200%	200%
Armor :	0	1	2-3	3+
Nat. Damg:	1D4	1D6-1D10	1D10-2D6	2D8+

If you want a monster to do a set number of damage points, set his (or his weapon's if he uses one) damage to Dice=Amount of damage, Sides=1. 10D1 will always do 10 points of damage. (Subtracting the defender's armor value, of course.)

If you want your monster to be especially fierce, code his Combat Code as -1, if you don't want your monster to fight ever, code his Combat Code as -2. As a general rule of thumb, most monsters should have a combat code of 0.

If you have an especially large group of monsters, such as "12 Orcs", you should probably scale them down to be a bit weaker than usual.