The ZZT File Format

Kevin Vance <kvance@zeux.org>

Version 1

Contents

1 Introduction **1**
- 1.1 About this document . . . 1
- 1.2 About the author . . . 1
- 1.3 The Story So Far . . . 1
- 1.4 The Files . . . 2

2 ZZT World Files **2**
- 2.1 Header . . . 2
- 2.2 The Rest . . . 3

3 ZZT Boards **3**
- 3.1 Individually Wrapped . . . 3
- 3.2 Board Header . . . 4
- 3.3 Run Length Encoding . . . 4
- 3.4 Board Information . . . 5
- 3.5 Objects . . . 6
  - 3.5.1 The Player . . . 6
  - 3.5.2 Scrolls . . . 6
  - 3.5.3 Pasageways . . . 7
  - 3.5.4 Duplicators . . . 7
  - 3.5.5 Bears . . . 7
  - 3.5.6 Ruffians . . . 7
  - 3.5.7 Objects . . . 8
  - 3.5.8 Slimes . . . 8
  - 3.5.9 Sharks . . . 8
  - 3.5.10 Spinning Guns . . . 9
  - 3.5.11 Pushers . . . 9
  - 3.5.12 Lions . . . 9
  - 3.5.13 Tigers . . . 9
  - 3.5.14 Centipede Heads . . . 10
  - 3.5.15 Centipede Segments . . . 10
  - 3.5.16 Blinking Walls . . . 10
  - 3.5.17 Transporters . . . 11

3.5.18 Bullets and Stars 11

4 Saved Games **12**
4.1 Just one bit 12

5 Appendix A: Code Table **12**

6 Appendix B: Colour scheme **13**

1 Introduction

1.1 About this document

The ZZT File Format exists to tell you everything you ever wanted to know about the ZZT file format. Hopefully, this will make everyones' life easier when they want to write a ZZT utility or clone. This document is jam-packed with information and diagrams for visualization of the file format. All numbers are given in hexadecimal unless otherwise noted. Good luck with your ZZT program!

Release history:

Number	Date	Note
Version 1	Jun 18, 1999	Initial release

1.2 About the author

The ZZT File Format is authored by the world famous Kev Vance ¡kvance@zeux.org¿ Any comments, suggestions, or corrections should be sent to that email address. Enjoy the document.

1.3 The Story So Far

ZZT is an early shareware game written by Epic MegaGames in the early 1990s. The difference between ZZT and other games of its time is that ZZT contains a built-in game editor. Besides the "true ZZT series," thousands of new games were written and distributed over Bulletin Boards, Online Services, and the Internet. The author of ZZT and president of Epic MegaGames, Tim Sweeney, lost the source code to ZZT as well as any documentation of its file format. The game was released into the public domain in 1998 still without any technical information... until now.

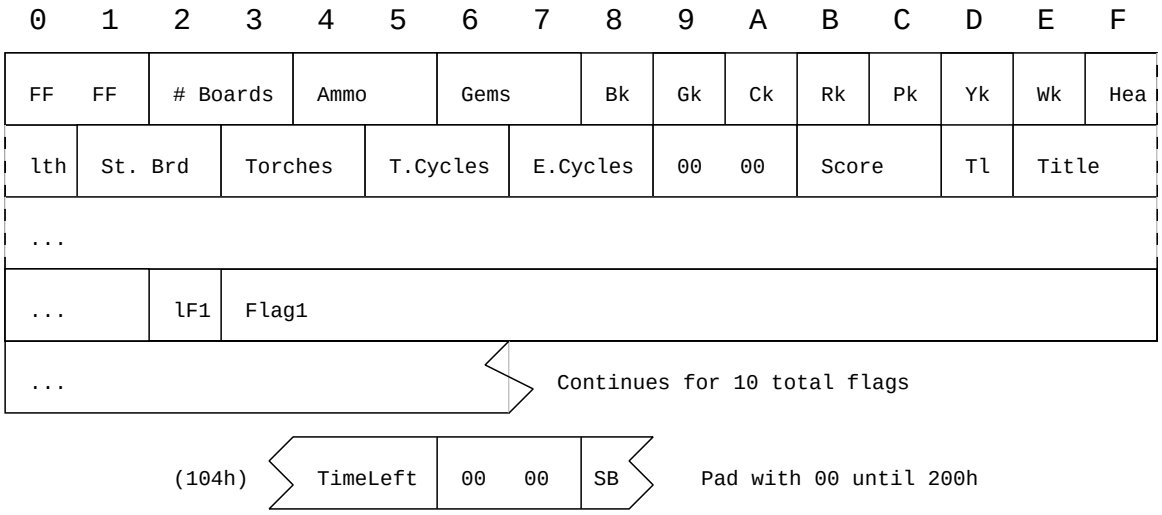
1.4 The Files

ZZT produces three different kinds of files, all similar. These files are:

Extension	Purpose	Description
*.ZZT	ZZT world file	Starting values, boards, and programming for an entire ZZT world
*.SAV	Saved game file	Same as a ZZT world file, with a saved game bit set
*.BRD	Board file	An excerpt of a ZZT world file containing a single board

2 ZZT World Files

2.1 Header



The ZZT header starts off with FF FF, this is the magic number to identify a ZZT world file. All of the 2 byte-long values in the header are Intel style and unsigned. The rest of the header is explained below.

Label	Location	Description
FF FF	00 - 01	Magic Number - always FF FF
# Boards	02 - 03	Board Count (zero based)
Ammo	04 - 05	Starting amount of ammo
Gems	06 - 07	Starting amount of gems
Bk	08	Blue Key: 0=no, 1=yes
Gk	09	Green Key: 0=no, 1=yes
Ck	0A	Cyan Key: 0=no, 1=yes
Rk	0B	Red Key: 0=no, 1=yes
Pk	0C	Purple Key: 0=no, 1=yes
Yk	0D	Yellow Key: 0=no, 1=yes

Wk	0E	White Key: 0=no, 1=yes
Health	0F - 10	Starting amount of health
St. Brd	11 - 12	Starting board number
Torches	13 - 14	Starting amount of torches
T.Cycles	15 - 16	Cycles remaining on lit torch
E.Cycles	17 - 18	Cycles remaining on used energizer
00 00	19 - 1A	Padding?
Score	1B - 1C	Starting amount of points
Tl	1D	Length of world title
Title	1E - 31	Title of world (usually filename)
lF1	32	Length of Flag 1
Flag1	33 - 46	Name of Flag 1 (lF and Flag repeat 9x more)
TimeLeft	104 - 105	Time left: 0=no limit, else=seconds left
SB	108	Saved game byte: 0=world, 1=saved game

2.2 The Rest

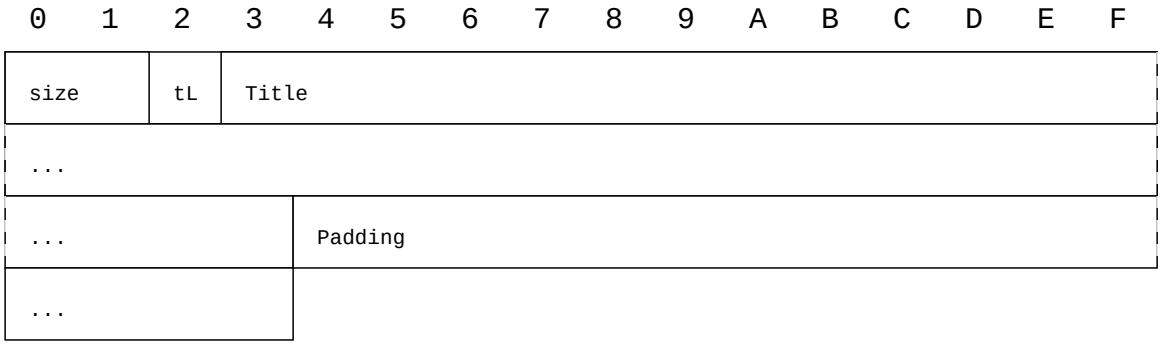
Starting at 200h, a ZZT file contains all of the boards written sequentially. Each board contains all of the information in a BRD file, explained in the next section.

3 ZZT Boards

3.1 Individually Wrapped

Normally, boards are written in order in a ZZT world file. However, to transfer boards between worlds ZZT produces a BRD file with all of the information from one board. These individually wrapped boards are manipulated with the Import and Export commands in the ZZT Editor.

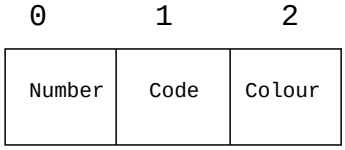
3.2 Board Header



Each board is preceded with a small header. In a ZZT world file, this would be written at 200h and subsequently before each new board. In a single board file, it is the first thing written.

Label	Location	Description
size	00 - 01	The size in bytes of the entire board
tL	02	The length of the board title
Title	03 - 23	The title of the Board
Padding	24 - 33	Just some padding (fill with 00s)

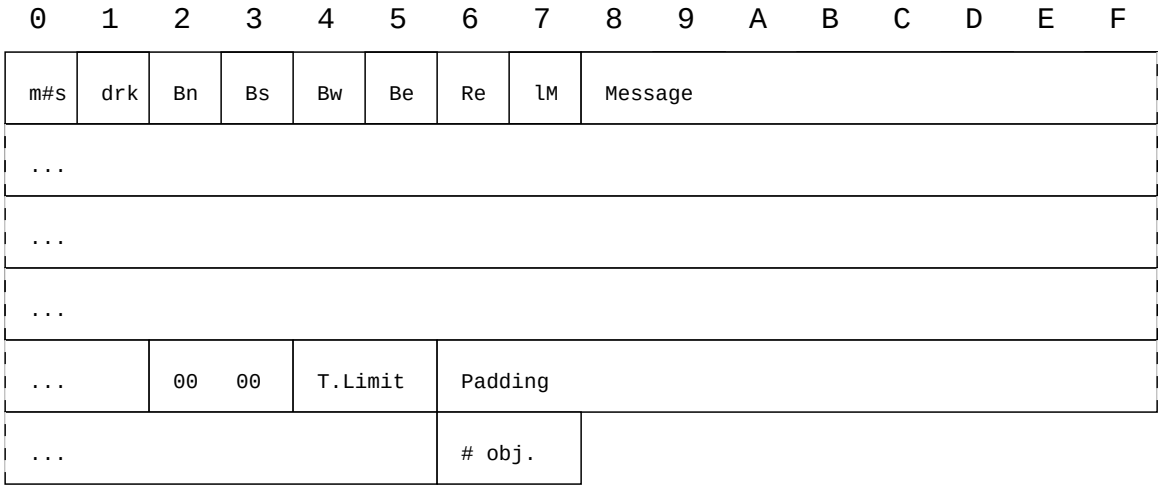
3.3 Run Length Encoding



ZZT uses a simple run length encoding scheme for storing board layout information. It encodes from the first tile on the first row to the second tile on the first row, wrapping around to the first tile on the second row as it goes. The first byte is the number of tiles to be expanded. The second byte is the code for that tile (see Appendix A). The third byte is the colour for that tile. The colour is represented in standard VGA video memory form (see Appendix B for more colour info).

The tiles are stored in this fashion for a whole screens worth of 60*25 (1500 tiles).

3.4 Board Information

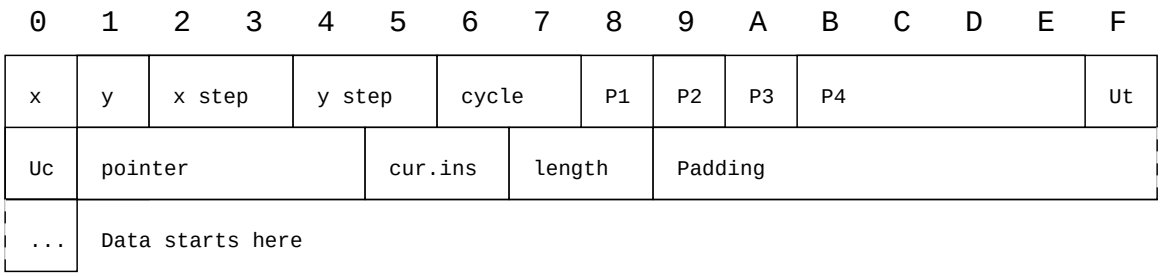


After the board tiles, the board information is written. This is the data that you modify in the "Board Information" dialog box in the ZZT Editor.

Label	Location	Description
m#s	00	Maximum number of shots fired
drk	01	Darkness: 0=no, 1=yes
Bn	02	Board # to north: 0=none, 1=second board
Bs	03	Board # to south: 0=none, 1=second board
Bw	04	Board # to west: 0=none, 1=second board
Be	05	Board # to east: 0=none, 1=second board
Re	06	Re-enter when zapped: 0=no, 1=yes
lM	07	Length of on-screen message
Message	08 - 41	On-screen message (internal use only)
00 00	42 - 43	Padding?
T.Limit	44 - 45	Time Limit for board
Padding	46 - 55	Fill with zeros
# obj.	56 - 57	Number of objects on board

Most of these are self-explanatory. It is not possible to link a board to the title screen. When a board link is set to zero, ZZT does not link that side to anything. When it is set to one, it links to board number one – the board after the title screen. The on-screen message is used by the game only. Modifying this will not cause ZZT to display a message. The object count includes everything with parameters: objects, scrolls, creatures, passageways, and more.

3.5 Objects



After the board information, the parameter records are written sequentially. The order is not important, except that the player (yes, the player needs parameters) must come first. The format for the object parameter records varies depending on what kind it is for. A scroll makes different use of the P values than a tiger. An important detail to note is that the x and y parameters are 1 based. The top left corner of the screen is (1, 1) not (0, 0). The "pointer" variable is only used internally by ZZT, writing to it is not necessary.

The sections below show how all of the parameter-needing objects in ZZT are used. Any unmentioned variables are unused by that object type and should have zeros written to them.

Every object also has a Ut and Uc variable. Ut is the code of the tile underneath the object and Uc is the colour of the tile underneath the object.

3.5.1 The Player

The player must always be the first object in the series of parameter records. The only known parameters it uses are the x and y values.

Label	Location	Description
x	00	Player's x position
y	01	Player's y position
cycle	06 - 07	Player's cycle (normally 1)

3.5.2 Scrolls

Scrolls contains ZZT-OOP that is executed when the player steps on the tile containing the scroll.

Label	Location	Description
x	00	Scroll's x position
y	01	Scroll's y position
cycle	06 - 07	Scroll's cycle (normally 1)
cur.ins	15 - 16	Offset of current instruction
length	17 - 18	Length of the scroll's program
data	21...	Scroll's program

3.5.3 Pasageways

Passageways transport the player from one board to another when the player steps on their tile. The destination board number starts with zero, which leads to the title screen. Although you cannot select this with the ZZT Editor, it is possible within the file format.

Label	Location	Description
x	00	Passage's x position
y	01	Passage's y position
P3	0A	Passage's destination

3.5.4 Duplicators

Duplicators copy the tile in one direction to the opposite direction. The source direction is given relative to the duplicator's position. To duplicate from one

square north to the square south, the coordinants would be (0, -1). No movement on the x axis, up one square on the y axis. The coordinants are written to the "x step" and "y step" variables and are 16 bit signed integers.

Label	Location	Description
x	01	Duplicator's x position
y	02	Duplicator's y position
x step	02 - 03	Copy source rel. x position
y step	04 - 05	Copy source rel. y position
P2	09	Rate

3.5.5 Bears

Bears are particularly daft creatures unless you crank up (down, really) their sensitivity. They charge at the player when they see him and they can bash through breakable walls at the cost of their own life.

Label	Location	Description
x	00	Bear's x position
y	01	Bear's y position
cycle	06 - 07	Bear's cycle (normally 3)
P1	08	Bear's sensitivity

3.5.6 Ruffians

Ruffians alternate between charging at the player and resting.

Label	Location	Description
x	00	Ruffian's x position
y	01	Ruffian's y position
cycle	06 - 07	Ruffian's cycle
P1	08	Ruffian's intelligence
P2	09	Ruffian's rest time

3.5.7 Objects

Objects, like scrolls, contain ZZT-OOP programming. Unlike scrolls, they are free to move and do not disappear until issued the "#die" command.

Label	Location	Description
x	00	Object's x position
y	01	Object's y position
cycle	06 - 07	Object's cycle (normally 3)
P1	08	Object's ASCII character
P2	09	This must equal 00
cur.ins	15 - 16	Offset of current instruction
length	17 - 18	Length of the object's program
data	21...	Object's program

3.5.8 Slimes

Slimes multiply and crawl across the board, leaving a mass of breakable terrain behind them.

Label	Location	Description
x	00	Slime's x position
y	01	Slime's y position
cycle	06 - 07	Slime's cycle (normally 3)
P2	09	Slime's speed

3.5.9 Sharks

Sharks are little fins that can only swim in water. If the player comes in contact with one, it hurts him.

Label	Location	Description
x	00	Shark's x position
y	01	Shark's y position
cycle	06 - 07	Shark's cycle (normally 3)
P1	08	Shark's intelligence

3.5.10 Spinning Guns

Spinning guns spin around and shoot at the player. They can not be destroyed and make for very annoying puzzles because they can shoot over water. They have a variable firing rate and can shoot bullets or stars. The default is to shoot bullets, but adding 128 to the firing rate will switch the firing mode to stars.

Label	Location	Description
x	00	Spinning gun's x position
y	01	Spinning gun's y position
cycle	06 - 07	Spinning gun's cycle (normally 2)
P1	08	Spinning gun's intelligence
P2	09	Spinning gun's firing rate / mode

3.5.11 Pushers

Pushers simply move everything they can in the direction they are going. Pushers can be blocked by sliders to make pusher/slider puzzles. Their direction is given relative to the pusher's position. To move north, the coordinants would be (0, -1). No movement on the x axis, up one square on the y axis. The coordinants are written to the "x step" and "y step" variables and are 16 bit signed integers.

Label	Location	Description
x	00	Pusher's x position
y	01	Pusher's y position
x step	02 - 03	Pusher's x step
y step	04 - 05	Pusher's y step
cycle	06 - 07	Pusher's cycle (normally 4)

3.5.12 Lions

Lions simply run around and try to kill the player.

Label	Location	Description
x	00	Lion's x position
y	01	Lion's y position
cycle	06 - 07	Lion's cycle (normally 2)
P1	08	Lion's intelligence

3.5.13 Tigers

Tigers are like a cross between lions and spinning guns. They run around trying to kill the player, but can also shoot him with bullets or stars. The default is to shoot bullets, but adding 128 to the firing rate will switch the firing mode to stars.

Label	Location	Description
x	00	Tiger's x position
y	01	Tiger's y position
cycle	06 - 07	Tiger's cycle (normally 2)
P1	08	Tiger's intelligence
P2	09	Tiger's firing rate / mode

3.5.14 Centipede Heads

Centipede heads lead the centipede segments around the board in a variably normal pattern.

Label	Location	Description
x	00	Centipede head's x position
y	01	Centipede head's y position
cycle	06 - 07	Centipede head's cycle (normally 2)
P1	08	Centipede head's intelligence
P2	09	Centipede head's deviance

3.5.15 Centipede Segments

Centipede segments follow around centipede heads until they get disconnected. They become heads after that.

Label	Location	Description
x	00	Centipede segment's x position
y	01	Centipede segment's y position
cycle	06 - 07	Centipede segment's cycle (normally 2)

3.5.16 Blinking Walls

Blinking walls flash a laser beam in a preset direction for a preset amount of time. The length of time of the shot and the length of time between shots can be different. The firing direction is given relative to the blinking wall's position. To shoot north, the coordinants would be (0, -1). No movement on the x axis, up one square on the y axis. The coordinants are written to the "x step" and "y step" variables and are 16 bit signed integers.

Label	Location	Description
x	00	Blinking wall's x position
y	01	Blinking wall's y position
x step	02 - 03	Blinking wall's x direction
y step	04 - 05	Blinking wall's y direction
cycle	06 - 07	Blinking wall's cycle (normally 1)
P1	08	Blinking wall's start time
P2	09	Blinking wall's firing period

3.5.17 Transporters

Transporters beam the player to another aligned transporter. They can face in any of the 4 directions. The transporting direction is given relative to the transporter's position. To transport north, the coordinants would be (0, -1). No movement on the x axis, up one square on the y axis. The coordinants are written to the "x step" and "y step" variables and are 16 bit signed integers.

Label	Location	Description
x	00	Transporter's x position
y	01	Transporter's y position
x step	02 - 03	Transporter's x direction
y step	04 - 05	Transporter's y direction
cycle	06 - 07	Transporter's cycle (normally 2)

3.5.18 Bullets and Stars

Bullets are the main form of attack in ZZT. The player fires them, creatures fire them, and objects fire them. Each bullet has its own parameter record to show who owns it and in which direction it's going. The direction is given relative to the bullet's position. To be moving north, the coordinants would be (0, -1). No movement on the x axis, up one square on the y axis. The coordinants are written to the "x step" and "y step" variables and are 16 bit signed integers. Stars work in the same way, but are only fired by enemies

Label	Location	Description
x	00	Bullet/Star's x position
y	01	Bullet/Star's y position
x step	02 - 03	Bullet/Star's x direction
y step	04 - 05	Bullet/Star's y direction
cycle	06 - 07	Bullet/Star's cycle (normally 1)
P1	08	Owner: 0=player, 1=enemy

4 Saved Games

4.1 Just one bit

The only difference between a ZZT world file and a saved game file is that the saved game bit is set. If 108h is set to 1, it is a saved game. If it is set to 0, it is a world file. You can't load saved games in the ZZT Editor and you can't restore world files as a saved game... until you toggle the saved game bit.

5 Appendix A: Code Table

Every type of tile in ZZT has a unique code which gets packed into boards. Many codes have a corresponding character that is displayed on the screen. This table is a list of all of the known codes with their characters given where available:

Code	Character	Description
00	00	Empty space
01	00	Special: acts like edge of board
04	02	Player
05	84	Ammo
06	9D	Torch
07	04	Gem
08	0C	Key
09	0A	Door
0A	E8	Scroll
0B	F0	Passage
0C	(growing O)	Duplicator
0D	0B	Bomb
0E	7F	Energizer
0F	(spins)	Star
10	(spins)	Clockwise conveyer
11	(spins)	Counterclockwise conveyer
12	F8	Bullet
13	B0	Water
14	B0	Forest
15	DB	Solid
16	B2	Normal
17	B1	Breakable
18	FE	Boulder
19	12	Slider: North-South
1A	1D	Slider: East-West
1B	B2	Fake
1C	(invisible)	Invisible wall

1D	(varies)	Blink wall
1E	(varies)	Transporter
1F	(varies)	Line
20	2A	Ricochet
21	CD	Horizontal blink wall ray
22	99	Bear
23	05	Ruffian
24	(varies)	Object
25	2A	Slime
26	5E	Shark
27	(spins)	Spinning gun
28	(varies)	Pusher
29	EA	Lion
2A	E3	Tiger
2B	BA	Vertical blink wall ray
2C	E9	Centipede head
2D	4F	Centipede segment
2F	(set in colour byte)	Blue text
30	(set in colour byte)	Green text
31	(set in colour byte)	Cyan text
32	(set in colour byte)	Red text
33	(set in colour byte)	Purple text
34	(set in colour byte)	Yellow text
35	(set in colour byte)	White text
36	(set in colour byte)	White blinking text
37	(set in colour byte)	Blue blinking text
38	(set in colour byte)	Green blinking text
39	(set in colour byte)	Cyan blinking text
3A	(set in colour byte)	Red blinking text
3B	(set in colour byte)	Purple blinking text
3C	(set in colour byte)	Yellow blinking text
3D	(set in colour byte)	Grey blinking text

6 Appendix B: Colour scheme

ZZT uses the same scheme for storing colours that is used in coloured text mode. There is a foreground and background colour for each cell. The foreground colour ranges from 0-F. The background colour ranges from 0-7 unblinking and 8-F blinking. Each number has a separate colour:

Foreground Colours:

Number	Colour
00	Black
01	Dark blue
02	Dark green
03	Dark cyan
04	Dark red
05	Dark purple
06	Dark yellow (brown)
07	Light grey
08	Dark grey
09	Light blue
0A	Light green
0B	Light cyan
0C	Light red
0D	Light purple
0E	Light yellow
0F	White

Background colours follow the same scheme, only there are no light colours. They loop around at 08, causing the foreground colour to blink.

The foreground and background colours get stored together in one byte. The left nybble holds the background colour and the right nybble holds the foreground colour. For instance, 1F would be white on dark blue, 81 would be blinking dark blue on black, and FF would be blinking white on light grey. This allows for 16 foreground colours to be used with 8 background colours instead of the normal 7 foreground, 0 background limitation of the ZZT Editor.