The unofficial LMP format description

Uwe Girlich

uwe@half-empty.de

v2.1.3, 3/12/1998

This document describes the LMP file format. It is the result of "recording" a game in DOOM, DOOM][, HERETIC, HEXEN and STRIFE. This documentation covers all these games in all known versions.

Table of Contents

1. Introduction

The LMP file format appears first with the game DOOM on December 10th, 1993 and remains relatively unchanged in all its successors.

I explain in this document the different flavors of LMP files from:

DOOM

- DOOM 0.99, 1.1, 1.2
- DOOM 1.4beta
- DOOM 1.5beta
- DOOM 1.6beta, DOOM 1.666
- DOOM 1.7a
- DOOM 1.8
- DOOM 1.9
- DOOM 1.10

Ultimate DOOM

- Ultimate DOOM 1.9

DOOM II

- DOOM][1.666
- DOOM][1.7, 1.7a
- DOOM][1.8
- DOOM][1.9

HERETIC

- HERETIC 1.0, 1.2, 1.3

HEXEN

- HEXEN beta
- HEXEN 1.0
- HEXEN 1.1

STRIFE

- STRIFE 1.0, 1.1

The LMP files created with games which appear on one line are not distinguishable.

To create a LMP file start the game with the command line switch `-record foo` (and `-warp`, `-skill` etc.) and play as usual. If you press `q` the record (and the game) stops. In old DOOM (< 1.4), HERETIC and HEXEN this happens at the end of a level too. You will find a file `foo.lmp` in the current directory. To play it back start the game with the command line switch `-playdemo foo`.

A LMP file records all player actions. The monster movements, respawn positions etc. are totally deterministic. The messages during a multiplayer game (chat mode) do not appear in the LMP.

It's a pity but the conversations in STRIFE, which form an important part of the game aren't recorded. You can only record sequences *without* these talks. In fact only the conversation start action will be recorded. If you play a LMP back, you can now choose the answers for yourself (and hope that the original player used the same answers).

A LMP file consists of a header, a data area and a quit byte.

- The header contains the skill, episode, start map, number of players etc.
- The data area is divided in game tics. For each player there are
  - 4 (DOOM and DOOM][) or
  - 6 (HERETIC, HEXEN and STRIFE)

  movement bytes per game tic.
- Each LMP should end with a quit byte (0x80).

The duration of a gametic is round 1/35s. With time measurements with 4dos:

```
timer ^ doom -playdemo name ^ timer
```

of many different LMP's and linear curve fitting I found the slightly different frequency 35.003Hz. So it is not wrong to speak of 35Hz but it is not totally correct.

2. Header

The easiest way to check the length of the header is to start the games with the `-record` option. Then you have to press "`q`" during the startup sequence. In the moment, when the screen switches from text mode to graphics mode, the game stops and you receive the shortest LMP file. It consists only of the header and the quit game byte.

The LMP header format is a subject of change.

2.1. Old DOOM, HERETIC

Table 1. 7 byte Header of old DOOM up to 1.2 and HERETIC.

address	type	contents	range
`0x0000`	`byte`	skill	0-4
`0x0001`	`byte`	episode	1-3/4
`0x0002`	`byte`	map	1-9
`0x0003`	`byte`	green/green player	0: inactive, 1: active
`0x0004`	`byte`	indigo/yellow player	0: inactive, 1: active
`0x0005`	`byte`	brown/red player	0: inactive, 1: active
`0x0006`	`byte`	red/blue player	0: inactive, 1: active

2.2. New DOOM, DOOM][

Table 2. From DOOM 1.4 on and in DOOM][the header consists of 13 bytes.

address	type	contents	range
`0x0000`	`byte`	version	look in the table below
`0x0001`	`byte`	skill	0-4
`0x0002`	`byte`	episode	1-4
`0x0003`	`byte`	map	1-32
`0x0004`	`byte`	multiplayer rule	0: Cooperative, 1: DeathMatch, 2: AltDeath
`0x0005`	`byte`	respawn	0: off, !=0: on, `argv` - number of `-respawn`
`0x0006`	`byte`	fast	0: off, !=0: on, `argv` - number of `-fast`
`0x0007`	`byte`	nomonsters	0: off, !=0: on, `argv` - number of `-nomonsters`
`0x0008`	`byte`	recording player	0-3
`0x0009`	`byte`	green player	0: inactive, 1: active
`0x000A`	`byte`	indigo player	0: inactive, 1: active
`0x000B`	`byte`	brown player	0: inactive, 1: active
`0x000C`	`byte`	red player	0: inactive, 1: active

Table 3. Known version bytes.

dec	hex	game(s)
104	0x68	DOOM 1.4beta
105	0x69	DOOM 1.5beta
106	0x6A	DOOM 1.6beta, DOOM 1.666 and DOOM][1.666
107	0x6B	DOOM][1.7 and DOOM][1.7a
108	0x6C	DOOM 1.8 and DOOM][1.8
109	0x6D	(Ultimate) DOOM 1.9 and DOOM][1.9
110	0x6E	Published DOOM source

Now it is very easy to change an existing LMP from one format to the other. The movement algorithm didn't changed from DOOM 1.2 to DOOM 1.666. This is only true if the map also didn't changed. So you can't recycle LMP's of E1M4. It is very interesting, that DOOM 1.666 LMP's, recorded even with the `-turbo` option work with DOOM 1.2.

2.3. HEXEN demo, HEXEN 1.0

Table 4. 11 byte header of HEXEN 1.0

address	type	contents	range
`0x0000`	`byte`	skill	0-4
`0x0001`	`byte`	episode	1
`0x0002`	`byte`	map	1-40
`0x0003`	`byte`	blue player	0: inactive, 1: active
`0x0004`	`byte`	blue player's class	0-2
`0x0005`	`byte`	red player	0: inactive, 1: active
`0x0006`	`byte`	red player's class	0-2
`0x0007`	`byte`	yellow player	0: inactive, 1: active
`0x0008`	`byte`	yellow player's class	0-2
`0x0009`	`byte`	green player	0: inactive, 1: active
`0x000A`	`byte`	green player's class	0-2

2.4. HEXEN 1.1

Table 5. 19 byte header for HEXEN 1.1 (with up to 8 players).

address	type	contents	range
`0x0000`	`byte`	skill	0-4
`0x0001`	`byte`	episode	1
`0x0002`	`byte`	map	1-40
`0x0003`	`byte`	blue player	0: inactive, 1: active
`0x0004`	`byte`	blue player's class	0-2
`0x0005`	`byte`	red player	0: inactive, 1: active
`0x0006`	`byte`	red player's class	0-2
`0x0007`	`byte`	yellow player	0: inactive, 1: active
`0x0008`	`byte`	yellow player's class	0-2
`0x0009`	`byte`	green player	0: inactive, 1: active
`0x000A`	`byte`	green player's class	0-2
`0x000B`	`byte`	jade player	0: inactive, 1: active
`0x000C`	`byte`	jade player's class	0-2
`0x000D`	`byte`	white player	0: inactive, 1: active
`0x000E`	`byte`	white player's class	0-2
`0x000F`	`byte`	hazel player	0: inactive, 1: active
`0x0010`	`byte`	hazel player's class	0-2
`0x0011`	`byte`	purple player	0: inactive, 1: active
`0x0012`	`byte`	purple player's class	0-2

2.5. STRIFE

Table 6. 16 byte header of STRIFE with an improved (8 players) DOOM][engine.

address	type	contents	range
`0x0000`	`byte`	version	101 (version 1.0)
`0x0001`	`byte`	skill	0-4
`0x0002`	`byte`	map	32-??
`0x0003`	`byte`	multiplayer rule	0: Cooperative, 1: DeathMatch, 2: AltDeath
`0x0004`	`byte`	respawn	0: off, !=0: on, `argv` - number of `-respawn`

0x0005	byte	fast	0: off, !=0: on, argv - number of -fast
0x0006	byte	nomonsters	0: off, !=0: on, argv - number of -nomonsters
0x0007	byte	recording player	0-7
0x0008	byte	brown player	0: inactive, 1: active
0x0009	byte	red player	0: inactive, 1: active
0x000A	byte	redbrown player	0: inactive, 1: active
0x000B	byte	grey player	0: inactive, 1: active
0x000C	byte	dark green player	0: inactive, 1: active
0x000D	byte	gold player	0: inactive, 1: active
0x000E	byte	bright green player	0: inactive, 1: active
0x000F	byte	super-hero blue player	0: inactive, 1: active

3. Data

3.1. Number of game tics

This formula for the number of game tics (NGT) follows from the section Section 1. section:

$$NGT = \frac{filelength - headerlength - 1}{bytes\ per\ gametic * number\ of\ players}$$

The NGT stored in a LMP can also be found out with the -timedemo option. At the replay end, the game shows the number of game tics and time tics. This game tic number corresponds with NGT only in newer versions of DOOM (>=1.4), in DOOM][and STRIFE. Older versions of DOOM report the wrong number (NGT+1). In HERETIC and HEXEN the -timedemo option doesn't work at all.

With -timedemo the game plays *all* scenes in the LMP file back but without any synchronizing. So you can calculate the frames per second also with:

$$fps = \frac{NGT}{time\ tics} * 35$$

This has nothing to do with your frame rate during the game. This frame rate can be computed with the formula (from the DOOM FAQ v6.666 [9-2-1], thanks to Hank Leukart (ap641@cleveland.freenet.edu (mailto:ap641@cleveland.freenet.edu)))

$$fps = \frac{}{dots + 1}$$

The dots are displayed in DOOM and DOOM][with the `-devparm` option in the lower left edge and in HEXEN with the `TICKER` cheat code (4 level demo: `RRETTENMUND`) in the upper left edge of the screen. In STRIFE the `TIC` cheat code toggles these dots. There are only 35 screens per second (each game tic), so one dot is always displayed. With every additional dot, the game shows how many vertical sync impulses (1/70s) were necessary to compute a whole screen. The game is not slower but you don't see each screen: the game becomes jerky.

3.2. Multiplayer LMP

In a 2, 3 or 4 player game the game tics are stored alternately, ie for a 2 player game:

```
game tic 1 of player 0
game tic 1 of player 1
game tic 2 of player 0
game tic 2 of player 1
game tic 3 of player 0
game tic 3 of player 1
          .
          .
          .
```

The problem with multiplayer records is the last game tic. It is indeed possible, to end a LMP file not with the actions of the last player. So the formula with the filelength in the section Section 3.1 is sometimes not correct.

3.3. The `-turbo` parameter (new DOOM, DOOM][and STRIFE only)

This parameter speeds up all moves (forward, backward, strafe sideways) but not the turn. Due to the value sets described in the section Section 3.4 all move bytes can be multiplied by 2.55 (or 255%) until they don't fit in a short int. This is exactly what can happen.

All normal move bytes are multiplied by the turbo value in percent. So you can play with `-turbo 200` a normal game but your speed is the same as the shift key is always pressed. If you now press the shift key, you are running like hell.

The `-turbo` parameter isn't stored in the LMP but the game shows in the message line, if a player plays with `-turbo` (ie "green is turbo"). The game concludes this from very large move bytes.

3.4. Data description

There are 3 different kinds of game tics in all the described games:

DOOM, DOOM II

```
----------------------------------------
|    0    |    1    |    2    |    3    |
|   go    | strafe  |  turn   |   use   |
----------------------------------------
```

HERETIC, HEXEN

```
------------------------------------------------------------
|    0    |    1    |    2    |    3    |    4    |    5    |
|   go    | strafe  |  turn   |   use   | vertical|artifacts|
------------------------------------------------------------
```

STRIFE

```
------------------------------------------------------------
|    0    |    1    |    2    |    3    |    4    |    5    |
|   go    | strafe  |  turn   |   use   | special |artifacts|
------------------------------------------------------------
```

3.4.1. Byte 0x00: Go Forward and Backward

The go byte contains a signed number (-127 ... 127), where positive numbers describe a forward and negative numbers a backward move.

The possible range of the values in the go byte varies a lot:

```
                   keyboard/gamepad/joystick      mouse
                      speed off  speed on
old DOOM (<1.4)        -25 +25   -50 +50       -50 .. +50
new DOOM (>=1.4)       -25 +25   -50 +50       -50 .. +50  *turbo/100
DOOM ][                -25 +25   -50 +50       -50 .. +50  *turbo/100
HERETIC                -25 +25   -50 +50       -50 .. +50
HEXEN fighter          -29 +29   -60 +60       -60 .. +60
HEXEN fighter (boots)  -44 +43   -90 +90       -90 .. +90
HEXEN cleric           -25 +25   -50 +50       -50 .. +50
HEXEN cleric (boots)   -38 +37   -75 +75       -75 .. +75
HEXEN mage             -21 +21   -45 +45       -45 .. +45
HEXEN mage (boots)     -32 +31   -68 +67       -68 .. +67
STRIFE                 -25 +25   -50 +50       -50 .. +50  *turbo/100
```

3.4.2. Byte 0x01: Strafe Sideways

The strafe byte contains a signed number (-127 ... 127), where positive numbers describe a right and negative numbers a left move.

The possible range of the values in the strafe byte varies a lot:

	keyboard/gamepad/joystick		mouse
	speed off	speed on	(strafe on + turn)
old DOOM (<1.4)	-24 +24	-40 +40	-50 .. +50
new DOOM (>=1.4)	-24 +24	-40 +40	-50 .. +50 *turbo/100
DOOM][	-24 +24	-40 +40	-50 .. +50 *turbo/100
HERETIC	-24 +24	-40 +40	-50 .. +50
HEXEN fighter	-27 +27	-59 +59	-60 .. +60
HEXEN fighter (boots)	-41 +40	-89 +88	-90 .. +90
HEXEN cleric	-24 +24	-40 +40	-50 .. +50
HEXEN cleric (boots)	-36 +36	-60 +60	-75 .. +75
HEXEN mage	-21 +21	-37 +37	-45 .. +45
HEXEN mage (boots)	-32 +31	-56 +55	-68 .. +67
STRIFE	-24 +24	-40 +40	-50 .. +50 *turbo/100

3.4.3. Byte 0x02: Turn

The turn byte contains a signed number (-127 ... 127), where positive numbers describe a left and negative numbers a right turn. The possible range of the values in the turn byte varies a lot:

	keyboard/gamepad/joystick				mouse	DOOM mouse spinner
	speed off		speed on			
	start	turn	start	turn		
old DOOM (<1.4)	-2 +1	-3 +2	-2 +1	-5 +5	-127 .. +127	+128
HERETIC	-2 +1	-3 +2	-2 +1	-5 +5	-127 .. +127	+128
HEXEN	-2 +1	-3 +2	-2 +1	-5 +5	-127 .. +127	+128
new DOOM (>=1.4)	-1 +1	-2 +3	-1 +1	-5 +5	-127 .. +127	+128
DOOM][	-1 +1	-2 +3	-1 +1	-5 +5	-127 .. +127	+128
STRIFE	-1 +1	-2 +3	-1 +1	-5 +5	-127 .. +127	+128

Therefore a right turn is in old DOOM, HERETIC and HEXEN faster and in new DOOM, DOOM][and STRIFE slower than a left turn.

The turn angle is

$$\frac{360}{256} * \text{turn\_byte degree}$$

With the DOOM cheat code `IDMYPOS` you can check your current viewing direction. During recording the games use only one byte out of a long for the viewing angle:

```
(0x40)
  N
```

```
(0x80) W   +   E (0x00)

          S
        (0xC0)
```

The turn byte is the increment of the viewing angle.

3.4.4. Byte 0x03: Use

In the use byte are coded the fire weapon, the press button/open door and the change weapon actions:

```
-----------------------------------------
| 7     | 6  | 5  | 4  | 3  | 2  | 1  | 0  |
| SA=0  | 0  |    Weapon    | WB | UB | FB |
-----------------------------------------
```

The Fire Bit (FB) is 1 during a shot. The Use Bit (UB) indicates the use of anything (press a button, open a door etc.). The Weapon Bit (WB) is 1 if the player changes manually his weapon. The weapon number is stored in the following bits.

The internal weapon numbers are beginning with 0; not with 1 as is the status bar:

Code := 4 + 8* (Weapon-1)

The codes for the different games are in the tables.

Table 7. DOOM

no	hex	name
1	0x04	Fist/Chainsaw
2	0x0C	Pistol
3	0x14	Shotgun
4	0x1C	Chaingun
5	0x24	Rocket Launcher
6	0x2C	Plasma Rifle
7	0x34	BFG 9000
8	0x3C	Chainsaw

Table 8. DOOM II

no	hex	name
1	0x04	Fist/Chainsaw
2	0x0C	Pistol
3	0x14	Shotgun/Super Shotgun
4	0x1C	Chaingun

5	0x24	Rocket Launcher
6	0x2C	Plasma Rifle
7	0x34	BFG 9000
8	0x3C	Chainsaw

Table 9. HERETIC

no	hex	name
1	0x04	Staff/Gauntlets
2	0x0C	Elvenwand
3	0x14	Ethereal Crossbow
4	0x1C	Dragon Claw
5	0x24	Hellstaff
6	0x2C	Phoenix Rod
7	0x34	Mace
8	0x3C	Gauntlets

Table 10. HEXEN: fighter

no	hex	name
1	0x04	Spiked Gauntlets
2	0x0C	Timon's Axe
3	0x14	Hammer of Retribution
4	0x1C	Quietus

Table 11. HEXEN: cleric

no	hex	name
1	0x04	Mace of Contrition
2	0x0C	Serpent
3	0x14	Firestorm
4	0x1C	Justifier

Table 12. HEXEN: mage

no	hex	name
1	0x04	Sapphire Wand
2	0x0C	Frost Shards
3	0x14	Arc of Death
4	0x1C	Bloodscourge

Table 13. STRIFE

no	hex	name
1	0x04	Punch Dagger
2	0x0C	Crossbow
3	0x14	Pulse Rifle
4	0x1C	Missile Launcher
5	0x24	Grenade Launcher
6	0x2C	Flame Thrower
7	0x34	Blaster
8	0x3C	Sigil
9	0x44	Punch Dagger

The bits can be composed. So you can with a use byte of 0x03 fire and use at the same time.

If the Special Actions (SA) bit is 1, the meaning of the other bits changes totally:

In the bits 0 and 1 is now coded the Special Action itself.

```
-----------------
|1|x|x|x|x|x|0|1|       stands for pause/unpause
-----------------
```

and

```
-----------------
|1|x|x| SLN |1|0|       stands for save game in slot SLN (0<=SLN<=7).
-----------------
```

The x-bits aren't important at all. The games themselves let them 0 but do not analyze them afterwards.

The games save a game state even during the replay of a LMP. So make sure you have no important game states in your directory if you plan to play back a unknown LMP file. You can remove the "Save Game" commands with `LMPC`, the Little Movie Processing Centre. Get the current version from my Demo Specs page (http://demospecs.half-empty.de).

3.4.5. Byte 0x04: Look and Fly

This byte appears in HERETIC and HEXEN LMP files *only*.

The Look an Fly byte is divided in 2 parts:

```
-----------------
|7|6|5|4|3|2|1|0|
|  fly  | look  |
-----------------
```

Both parts contain a signed 4-bit number, where positive numbers describe a upwards and negative numbers a downwards move.

Table 14. Look- and Fly-values.

game	speed off	speed on	center/drop
HERETIC	-3, +3	-5, +5	+8
HEXEN	-1, +1	-2, +2	+8

The pure “drop” (landing) code (0x80) could be ambiguous (the quit byte is 0x80). Moreover it is always good to look forward after landing. Therefore creates a “drop” always a “look forward”. This results in the code 0x88.

3.4.6. Byte 0x05: Use artifacts

This byte appears in HERETIC and HEXEN LMP files *only*.

The cursor movements in the inventory isn’t recorded. Only the use of an artifact is recorded. The “Use artifacts” byte contains simply the number of the artifact equivalent to the HERETIC “`gimme`”-cheat.

Table 15. HERETIC artifacts.

no	code	letter	name
1	`0x01`	a	ring of invincibility
2	`0x02`	b	shadowsphere
3	`0x03`	c	quartz flask
4	`0x04`	d	chaos device
5	`0x05`	e	tome of power
6	`0x06`	f	torch
7	`0x07`	g	time bomb of the ancients
8	`0x08`	h	morph ovum
9	`0x09`	i	wings of wrath
10	`0x0A`	j	mystic urn

A similar cheat code for HEXEN isn't known.

Table 16. HEXEN artifacts.

no	code	letter	name
1	`0x01`	a	icon of the defender
2	`0x02`	b	quartz flask
3	`0x03`	c	mystic urn
4	`0x04`	d	clerical healing
5	`0x05`	e	dark servant
6	`0x06`	f	torch
7	`0x07`	g	porkalator
8	`0x08`	h	wings of wrath
9	`0x09`	i	chaos device
10	`0x0A`	j	flechette
11	`0x0B`	k	banishment device
12	`0x0C`	l	boots of speed
13	`0x0D`	m	krater of might
14	`0x0E`	n	dragonskin bracers
15	`0x0F`	o	disc of repulsion
-	`0x21`		panic button (use all you have)

The HEXEN-specific "Jump" is coded in the Use artifact byte.

Table 17. HEXEN jump.

dec	hex	name
128	0x80	jump

A jump can be combined (with "binary or") with the use of an artifact.

3.4.7. Byte 0x04: Special Use

This byte appears in STRIFE LMP files *only*.

Table 18. The 8 toggle actions of STRIFE.

bit no	hex	meaning
0	`0x01`	look up

1	`0x02`	look down
2	`0x04`	run
3	`0x08`	use inventory (see section Section 3.4.8)
4	`0x10`	drop inventory (see section Section 3.4.8)
5	`0x20`	jump
6	`0x40`	unknown
7	`0x80`	use health

Note bit 2. STRIFE is the only game with a special run code. It seems senseless, because the go/strafe/turn-values are already "run-changed".

The cursor movements in the inventory isn't recorded. Only the use of an artifact is recorded. If bit 3 or 4 is 1 the next byte in the STRIFE game tic byte becomes important.

3.4.8. Byte 0x05: STRIFE Artifacts

This byte appears in STRIFE LMP files *only*.

This byte describes the artifact to use or to drop. 1 byte can hold up to 255 artifacts.

Table 19. 12 known STRFE artifacts.

code	name
`0x74`	toughness
`0x75`	accuracy
`0x76`	full health
`0x7B`	teleportor beacon
`0x7C`	metal armor
`0x7D`	leather armor
`0xA1`	med patch
`0xA2`	medical kit
`0xA3`	coin
`0xA7`	shadow armor
`0xA8`	environmental suit
`0xB7`	offering chalice

3.4.9. Special entries

If there are no actions at all, all 4 (or 6) bytes contain 0x00. This is a wait tic.

The end of the LMP is coded in the go byte of an incomplete game tic. Because there is no go value of -128 (0x80), the game can't make a mistake.

4. Version History and Acknowledgments

1.0, 30 August, 1994

- first public release
- many thanks to Steffen Winterfeldt (wfeldt@itp.uni-leipzig.de (mailto:wfeldt@itp.uni-leipzig.de)) for his reverse engineering work

1.10, 24 October, 1994

- general release for DOOM
- corresponds now to the LMP compiler `lmpc` 2.1

1.20, 16 January, 1995

- included information about DOOM][and HERETIC
- first HTML version
- thanks to Frans P. de Vries (fpv@xymph.iaf.nl (mailto:fpv@xymph.iaf.nl)), who corrected some orthographical bugs

1.21, 22 February, 1995

- weapon names for HERETIC
- artifact names for registered HERETIC

1.22, 8 October, 1995

- correct multiplayer colors
- included first information about HEXEN (taken from the 4-level-beta)

1.23, 9 October, 1995

- artifact names for HEXEN

1.24, 10 October, 1995

- go/strafe ranges reorganized (“boots of speed” included)

1.25, 25 October, 1995

- first Linuxdoc-SGML version, text and ps-versions are available too

1.26, 20 November, 1995

- some small mistakes in the movement byte table corrected, thanks to Rob McCartney (argon@netcom.com (mailto:argon@netcom.com)) for his bug report

2.0.0, 21 November, 1995, uploaded to ftp.cdrom.com (ftp://ftp.cdrom.com/) and announced in rec.games.comp.doom.announce (news:rec.games.comp.doom.announce)

- new 3 part version number
- rearrangements of the tables

2.1.0, 10 March, 1996, uploaded to ftp.cdrom.com (ftp://ftp.cdrom.com/) and announced in rec.games.comp.doom.announce (news:rec.games.comp.doom.announce)

- STRIFE information included
- rearrangements of the tables
- strafe table corrected

2.1.1, 7 April, 1996

- HEXEN 1.1 information included
- HEXEN panic button described

2.1.2, 20 April, 1996

- STRIFE 1.1 information included (player colors)
- table of all covered game versions

2.1.3, 12 March, 1998

- PlanetQuake is the new home.
- SGML-Tools 1.0.5 used.