██████████████████████▄ ▄▄████████████████▄▄ █████████████████████▄▄

████████████▀▀▀▀▀▀█████ █████████████▀▀▀▀█████ ▀█████████████▀▀▀▀▀█████

██████████████████████▀ ████████████████████████ █████████████ █████

████████████ █████ █████████████ █████ █████████████ ▄█████

██████████████ ███████ ███████████████ ███████ █████████████████████████

██████████████ ███████ ███████████████ ███████ ████████████████████████

██████████████ ███████ ███████████████ ███████ █████████████████████▀▀

┌────────── ── ─ -∙· Reality ADlib Tracker Version 1.1a ·∙- ─ ── ──────────┐

│ │

│ (c)1995 by Shayde, Rogue, & Void. │

| -REALITY PRODUCTIONS- │

: │

· |

:

∙

INTRODUCTION

▀ ▀▀▀▀▀▀▀▀▀▀▀▀▀▀ ▀

Why?

~~~~

Why? Well take a look around and see if YOU can find any decent Adlib

trackers around!

Being a demo group, Reality had occasion to need Adlib tunes for our

· intros. Void (our musician) went to work producing a tune in

. Scream-Tracker 3, despite the fact that it was designed more as a sample

∙ tracker and Adlib seemed to be a tacked on after-thought. But that

: wasn't the main problem. When it came time to put the tune into the

| intro....where the hell do you find player code?!

│ That's right. Future Crew didn't see fit to include player code with ST3 ·

hence making it useless for demo/intro development. .

│ The technical documentation wasn't thorough enough either so my first :

│ attempt at writing player code for it went miserably awry. |

│ │

│ Then we heard that a few groups were using a package called HSC-Tracker :

│ in their intros. We saw it on the boards so we downloaded it, noting │

that it was described as being "The Ultimate Adlib Tracker" including │

│ player code. Well this sounded like our problems were over so we gave it │

| to our musician to try.... Hmmm. I think I heard him scream from across │

: the Tasman! To the average user, HSC-Tracker looks like quite a complete │

. package but the actual editor program was too simplistic and unfriendly │

· for use (sorry Chicken!). What's more, the included "linkable" player │

code was in OBJect format which doesn't make customising it very easy, │

and is unusable if you use a different MODEL or want to use it in │

protected mode. │

|

So after that, the others in the group said "Fuck it. We'll just have :

to use a sample module (not that it's easy finding decent player code for ∙

that!) and forget about Adlib." But instead I decided to stop work on .

· the intro and knock up a quick Adlib editor for our musician to use.

. ·

∙ Hence was born RAD, which stands for Reality ADlib tracker. In the end,

: we got so involved in putting everything WE wanted into RAD that it has

| had a lot more effort put into it than was originally intended. It is

│ now a fully-fledged tracker, and we think it is the best Adlib tracker

│ you're likely to find in the scene. But not just that, for the demo

│ crews out there, RAD comes complete with full player source-code for

║ playing RAD tunes in your intros/demos. .

│ Just like the old days on Amiga...! :) :

║ │

╠═─══─═─── ── ─ - ∙ · ─ ── ───- ─────═──═─══─═══──══════════════┤

│ |

│ What Does It Do? :

│ ~~~~~~~~~~~~~~~~ ∙

│

| RAD, as with all trackers, is a program that allows you to author musical

: arrangements. It was based more-or-less on Protracker (on the Amiga)

. which is the model that most music trackers seem to be based on.

Tracker tunes are made up of three main components:

·

\* Instruments

\* Order list ∙

\* Patterns .

:

These will be described later. |

│

RAD has the following features: │

║

· \* Up to 31 separate instruments │

∙ \* Up to 32 separate patterns ║

: \* Up to 128 patterns in the order ║

│ \* 9 channels ║

│ \* Compact tune file size. │

║ (A fraction of the size of OTHER trackers!) ║

│ \* Easy to use ║

║ │

╠═════─═══─══─═────── ─ -∙ · · ∙ ─ ── ───╣

║ ║

║ │

│ INSTRUCTIONS ║

║ ▀ ▀▀▀▀▀▀▀▀▀▀▀▀▀▀ ▀ │

║ |

│ Starting RAD │

║ ~~~~~~~~~~~~ |

│ :

│ To start RAD, run the .EXE file, ie: ∙

| ·

: RAD<enter>

.

· You will then see the main RAD screen with credits information in the ·

lower half. Press a key to clear the credits window. .

· Obtaining help while using RAD

. ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

|

│ To enter the multi-page help system, press "?". Keys handy to know

│ while traversing the pages:

│

│ ESC - Return to the pattern editor

│ ? - Go back to the last major menu

│ BACKSPACE - Go back to the last page you were on (up to 64 times)

| WHITE letters - Keys to press to move to another page.

│

. As from V1.1a, a new quick-key reference has been added. Press F12 and ·

| you will see the start of a large scrollable document where the Pattern ∙

: Editor used to be, which gives a run down on all the keys you can use .

. within RAD. :

· |

│

│

Layout of RAD |

~~~~~~~~~~~~~ :

.

As tracker tunes have three main parts, RAD's screen is split up into ·

three areas corresponding to those parts. In the top half of the screen

· in the left-hand window is the Instrument Editor. To the right of this

∙ window is another, thinner window that is the Order-List Editor.

. The entire lower half of the screen is (as in most trackers) the Pattern

: Editor. The first key to know is the " ` " (back tick) key. This

| Switches between the three areas of the screen as you can only be using

│ one area at a time.

│ ·

│ When you first enter RAD, you start off in the Pattern Editor. .

: :

├────── ── ─- - ∙ · · -─ ────┤

| |

: Pattern Editor :

. ~~~~~~~~~~~~~~ ·

· .

The Pattern Editor is where you enter the actual notes that will be ∙

played. RAD allows up to 32 separate patterns and each pattern is made :

up of 64 lines. Each line has 9 entries corresponding to the 9 Adlib │

channels. As RAD plays a pattern it steps through each line at the given .

speed (described later) outputting the note in each column (if there IS │

one) to the corresponding Adlib channel. Each entry in the pattern is |

made up of FOUR items, eg. │

│

C#3 2C30 ║

├┘│ ││├┘ │

└─┼─┼┼┼────── 1) The NOTE that is played. "#" = sharp, "-" = normal. ║

│ │││ ║

└─┼┼┼────── 2) The OCTAVE of the note. From 0 to 7. ║

│││ ║

└┼┼────── 3) The INSTRUMENT the note is played in (in Hex). │

· ││ ║

. └┼────── 4) The SPECIAL EFFECT, (if any) to perform. ║

: └───────── and the parameter for the special effect. │

| ║

│ │

│ The note item follows usual musical notation. There are twelve available │

| notes per octave: .

∙ │

: C, C#, D, D#, E, F, F#, G, G#, A, A#, B |

· :

You can write a note in a given position in the pattern by pressing .

one of the NOTE KEYS, as set out below: ·

Actual

Note- C# D# F# G# A# C# D# F# G# A# C# D#

▐██ ▐█ ▐██▐██ ▐█ ▐█ ▐██▐██ ▐█ ▐██▐██ ▐█ ▐█ ▐██▐██ ▐█ ▐██

Keys- ▐██ S▐█ D▐██▐██ G▐█ H▐█ J▐██▐██ 2▐█ 3▐██▐██ 5▐█ 6▐█ 7▐██▐██ 9▐█ 0▐██

▐██ ▐█ ▐██▐██ ▐█ ▐█ ▐██▐██ ▐█ ▐██▐██ ▐█ ▐█ ▐██▐██ ▐█ ▐██

▐██ ▐█ ▐██▐██ ▐█ ▐█ ▐██▐██ ▐█ ▐██▐██ ▐█ ▐█ ▐██▐██ ▐█ ▐██

▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███

Keys- ▐▌Z█▐▌X█▐▌C█▐▌V█▐▌B█▐▌N█▐▌M█▐▌Q█▐▌W█▐▌E█▐▌R█▐▌T█▐▌Y█▐▌U█▐▌I█▐▌O█▐▌P█

▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███▐███

Actual

Note-C D E F G A B C D E F G A B C D E

When you write a note, the octave is also written. You can change the

current octave using the "," and "." keys. The currently selected

instrument is also written and you will hear the instrument played at

the given pitch.

·

. You can change instruments by pressing "[" and "]".

: ·

| You can navigate the Pattern Editor by using the CURSOR KEYS, ∙

│ and the following: .

│ :

∙ TAB - move to the next channel |

: BackTAB - move to the previous channel │

. PgUp - move up 16 lines │

· PgDown - move down 16 lines │

Home - move to the top of the pattern :

End - move to the end of the pattern │

|

To add/alter special effects, cursor over to the special effects :

column and use one of the following numbers/letters: .

·

0 - No effect

1 - Slide note down (pitch bend). Parameter gives speed of

slide.

· 2 - Slide note up (pitch bend). Parameter gives speed of slide.

∙

. 3 - Slide to note. This slides the LAST note played till it

: reaches the note given with this effect. The parameter

| gives the speed of the slide. Use parameter "00" in

│ subsequent lines to continue the slide.

:

│ 5 - Combined slide to note & Volume slide. This command has

│ the same functionality as effect 3 with parameter "00" but

∙ in this case, the parameter details a volume slide.

│ 1-49 fades the volume of the instrument down.

│ 51-99 fades the volume of the instrument up.

║

│ A - Volume slide. 1-49 fades the volume of the instrument down

║ at the given speed. 51-99 fades the volume of the

║ instrument up at the given speed.

│

║ C - Set volume. Sets the absolute volume of the instrument.

║ Values range from 0 (silent) to 64 (loudest).

║

║ D - Break. When RAD encounters this command it skips the rest

║ of the pattern and moves to the next pattern as specified

║ in the Order List. The parameter, if given, specifies a

║ line in the next pattern to start from.

║ eg. Effect "D12" would break to line 12 of the next pattern

║ in the Order List.

│ This command is handy for "cutting short" a pattern if

║ you don't want to use all 64 lines. Just place a D00 on the

║ last line you used and the rest will be ignored.

│

║ F - Set speed. By default RAD starts up with a speed of 6 but

│ you can use this command to slow down and speed up the

│ playback rate of your tune. To understand the speed value,

| know that RAD uses a 50Hz timer which is delayed by the

│ speed value between each line of the pattern. ·

| What this means is that, say, if you set the speed to 50,

: (using "F50"), then each line will play at the rate of one ·

∙ per second. Special effects, however, are modified ∙

· 50 times a second. .

:

To play the current pattern (to see how it sounds) press F6. |

Pressing F6 again will stop the playback. To move to a new pattern │

press "-" and "+/=". .

│

You can use the following editing keys to help you in writing patterns: │

│

INSERT - insert a line in the current channel ║

DELETE - delete the current line in the current channel │

SPACEBAR- wipe the current item only (also stops playback). ║

· F1 - Move to the top left corner of the region you wish to copy ║

. and press F1. Then move to the lower right corner and ║

: press F1 again. The highlighted region will then be moved │

| into a copy buffer. ║

: F2 - Will paste the contents of the copy buffer to the current │

│ cursor location. |

. F3 - Will copy the entire pattern into the copy buffer. :

│ F4 - Will replace the entire current pattern with the contents .

│ of the copy buffer (be careful!).

│ F9 - decrement the current speed value ·

│ F10 - increment the current speed value :

│ |

╞═══════════─═══─══─═──═── ─── ── ─- ─ - ∙ · · - ─ -─ ── ────╡

| │

: Instrument Editor │

. ~~~~~~~~~~~~~~~~~ │

· │

To edit your instruments you need to switch to the Instrument Editor. │

Use the " ` " (back tick) key to switch between windows until you see a │

highlight bar appear over the top-left item in the instrument window. |

:

RAD supports up to 31 instruments and it is these instruments that ∙

actually produce the sound. Each note in your patterns specifies an ·

instrument that that note is played on. Instruments can vary

significantly in sound from each other and, accordingly, your tune's

mood will change dramatically depending on the instrument mix you use.

Some instruments sound good as lead, other as bass and others are only

useful as percussion.

To begin with you will probably find it easier just to load

pre-written tunes and use the instruments that came with them (you can

do this by loading the tune and pressing Control-C to clear the patterns

leaving the instruments intact).

However, to design your own instruments, once you are in the Instrument

· Editor you can interactively alter the currently selected instrument's

. properties (remember "[" and "]" select instruments). Use Cursor Up and

∙ Down to select the property to alter, and Cursor Left and Right to

: decrease or increase or toggle the property's value. Alternatively,

. you can press ENTER on the parameter you wish to change, and input the

value manually with the numeric keys. There isn't much to say regarding

: how to correctly set properties, as different things produce different

| effects. Simply alter things and press a Note Key to see how it sounds!

│ The one property's effect you SHOULD be aware of is the CARRIER's volume.

. This governs the output volume of the instrument and, in fact, this is

│ the property that is modified by the Set Volume and Volume Slide commands.

║

│ Other than that, Good Luck! :-)

║

║ If you create a cool sounding instrument, you'll probably want to put

║ it aside somewhere for future use. Use Control-G (Get instrument) and

│ Control-P (Put instrument) to load and save instruments. Refer to .

║ Loading and Saving below on how to use the interactive directory. :

│ |

├─────── ── ─- ─ - · ─ ── ─── ───────────────────────╣

: ║

. Order List Editor ║

· ~~~~~~~~~~~~~~~~~ ║

║

Once you have created a few patterns you will want to string them │

together to create a complete tune. By moving to the Order Editor (use ║

" ` " remember) you can define when each pattern will be played. Line 00 ║

corresponds to the first pattern to be played, line 01 the second, etc. │

By using Cursor Up & Down to select a line, with Insert or Delete to ║

remove or add a line as necessary, you can use Cursor Left and Right to │

set which pattern will be played at that position in the song. │

for example, suppose you had the following Order List: :

│

00│03 .

01│01 :

· 02│04 ∙

. 03│01 ·

: 04│02

| 0502

│

: When you start your tune playing (with F5), it will start by playing

│ pattern 03 (specifed by line 00). Once it has played all 64 lines of

║ pattern 03 (or less if you used a Break command) it will then play the

│ pattern specified by the next line, in this case, pattern 01.

║ Each line will be played one after the other, playing the pattern listed

║ for that line.

║ As you can see in the above example you can re-use patterns (Pattern 01

│ is used in line 01 and 03) if there is a chorus or refrain you want to

║ repeat in your tune.

│

| If you notice in the above example, there is a line that contains a

│ right arrow between the line and the pattern. This is a JUMP MARKER

: and it tells RAD to jump to a new line. So when RAD gets to line 05 in .

. the example, rather than playing pattern 02 it will instead jump to LINE :

· 02. Hence the list of patterns that will be played with the above Order |

List is: │

│

03, 01, 04, 01, 02, 04, 01, 02, 04, 01, 02.... │

│ │

└────── Repeat caused by Jump Marker │

│

You can mark the current line as a jump marker by pressing '\' and ║

then using Cursor Left & Right to set the line the marker will jump to. │

A Jump Marker cannot be placed on Line 00. ║

║

· RAD will repeat the song from Line 00 when it gets to the end of the │

. Order List so you don't need to place a Jump Marker that jumps to Line ║

: 00 at the end of your Order List if you want your tune to repeat. ║

| ║

├── ─- ─ ∙ · · ∙ ─ ── ─────═─══─══╣

│ │

│ Loading & Saving Tunes ║

: ~~~~~~~~~~~~~~~~~~~~~~ │

│ │

| Naturally enough, once you have completed your Symphony No. 5, you'll │

: want to keep a permanent record of it. To save your tune press │

∙ Control-S to bring up the directory. You can select an existing :

· filename by positioning the highlight bar on it, using the cursor keys, │

and then pressing Enter. By selecting a directory or a device you can |

move around your disk structure. If you are saving a new file then you :

will need to type a new filename. Press TAB to switch to the filename ∙

field and enter the desired filename, using the editing keys to fix ·

typing mistakes as necessary. Press Enter to save. If you are saving

over an existing file you will be asked to confirm.

Loading follows a similar procedure. Press Control-L and use the

directory and filename field to select the desired tune file. Whereas

· saving, however, only creates RAD compatible files, you may load not

∙ only RAD files, but ScreamTracker 3 and HSC-Tracker compatible files.

.

: Note:

| Due to differences in structure between the different tracker

│ packages, converted tunes will sometimes not sound like their

│ originals. Therefore loading .S3M and .HSC tunes should really be

│ viewed as a tool for converting these formats to RAD format, rather

: than for general purpose playing.

│

|

: Adding a Description

. ~~~~~~~~~~~~~~~~~~~~

·

Because tracker tunes circulate widely and are often not accompanied

by documentation, traditionally musicians have placed author information

and greets in the instrument fields of their compositions. RAD does not

have instrument fields but you can add a description to your tune by

entering the Description Editor. Press Control-D to toggle between

editing the Description and editing the Pattern. The Description Editor

is a simple free-form one page text editor. Type whatever you like,

where you like, even if it's just the name of the tune!

·

Because RAD was written as an intro tune development system, the tune ∙

files were designed to be fairly compact so as not to take up much room :

· in any programs it was used in. Adding a Description to a tune will |

. add to the length of the tune. In the bottom right corner of the .

: Description Editor is a counter that displays the number of bytes that │

│ the current Description text will occupy in the saved file so you'll .

. always know just how much space your hot air is going to take up! :-) │

│ │

│ You may not see much use for a Description especially if the tune is │

║ ultimately destined for use in an intro but the fact is module-ripping │

│ is a popular past-time and chances are, if your tune is awesome, it WILL │

║ be ripped out of your intro and spread around. You can at least make |

║ sure everyone knows who wrote the ripped tune because the description :

│ will be ripped along with the tune! ∙

║ ·

╠═─═──══──════════─═══─══──═──── ── ─- ─ ∙ ·

║

│ ·

║ CODING NOTES ∙

│ ▀ ▀▀▀▀▀▀▀▀▀▀▀▀▀▀ ▀ :

│ |

│ Yo coding-type dude! │

│ This section is geared more towards the seasoned coder, so skip this if │

│ you're not! │

: │

│ RAD's whole reason for being, is as a tool for writing tunes for use │

| in intros. To this end, the FULL player source-code for playing RAD │

: tunes in your productions is included. The format follows that |

∙ traditionally set by trackers on the Amiga. Three routines are :

provided: .

·

InitPlayer

EndPlayer

PlayMusic

During your initialisation step in your intro, include a call to

"InitPlayer" with ES:0 pointing to a RAD module. When you close down

your intro or want the tune to stop, call "EndPlayer". Now to have the

tune play, you need to call PlayMusic 50 times a second for a tune

written in "normal timer mode" or 18.2 times a second (the standard PC

timer cycle) for a tune written in "slow timer mode". You can check bit

· 6 of the byte at offset 17 in the RAD module to see which timer mode the

∙ tune was written for. If the bit is set then it is a "slow timer" tune.

:

| The "slow timer mode" was provided for those people that just want a

│ simple interrupt structure for playing the music, as the normal 50Hz rate

: requires a re-program of the hardware clock which may be needed as a

│ Vertical Refresh timer. As a lot of intros provide support for calling

│ the system interrupt 18.2 times a second it is easy enough to add a call

║ to PlayMusic there. If you don't then you'll have to use one of the

│ Adlib timers to get a 50Hz or 18.2Hz reference signal for when to call

║ PlayMusic during your Vertical Refresh interrupt.

║

│ By pressing '\*' in RAD you toggle normal and slow timer modes for the

║ current tune. HSC tunes are automatically loaded as slow-timer tunes

║ for compatibility.

║

║ The provided player code was written as a real-mode player and to

║ reference variables through CS: because I didn't want to use ASSUMEs

║ which would replace any ASSUMEs you had set up in your main code. Be

│ careful if you wish to remove the CS: overrides as the player uses DS:

║ as a pointer to the tune file. You will HAVE to remove the CS: ·

║ overrides if you wish to use the player in protected mode. ∙

│ :

║ But then, that's the beauty of having the full source code to the │

│ player - you can hack it around all you like to fit in with your code, .

| something that OTHER tracker writers don't seem to realise with their │

: .OBJ format players, WHEN they bother to release players for their │

. trackers at all... │

· ║

Oh... If you DO use RAD or its player in your production, all we ask is │

that you give REALITY a mega-greet or credits somewhere! :-) ║

║

\*\* NOTE: Apologies to all who used the V1.0 player code only to find it │

didn't play tunes exactly as they played within RAD. This bug ║

in the PLAYER.ASM code has been mashed! ║

│

Since the first release of RAD, some coders have asked why I require the ║

player code to go through the timer interrupt. What a few people don't ║

seem to realise is that I've only SUGGESTED this as an example, as this ║

· is the most likely method of calling the player code. For correct play- ║

∙ back of your RAD tune, you SHOULD call PlayMusic 50/s (or 18.2/s) but how ║

. you supply these rates is entirely up to you! ║

| ║

├─═─══─═══─═─── ─── ── ─- ─ - ∙ · · ∙ - ── ─── ─═╣

. │

· |

TECHNICAL SPECIFICATIONS :

▀ ▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀ ▀ ∙

RAD V1.0+ file format is as follows: (values in hex)

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ HEADER █

█────────────────────────────────────────────────────────────────────────█

█ Offset 00..0F:"RAD by REALiTY!!" - Use this to recognize a RAD tune. █

█ Just detecting the initial "RAD" █

█ should be sufficient though! ;-) █

█ █

█ Offset 10:BCD - Version of RAD file (not RAD ver.) █

█ Currently 10h for v1.0. █

█ █

█ Offset 11:Bit 7 - Set if a description follows. █

█ Bit 6 - Set if it's a "slow-timer" tune. █

█ Bits 4..0 - The initial speed of the tune. █

█ █

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ DESCRIPTION █

█────────────────────────────────────────────────────────────────────────█

█ The description follows the header if Bit 7 of Offset 11h is set. The █

█ description is a null-terminated string. To save some space in the █

█ tune file, the characters mean: █

█ █

█ Char 00 - End of description (null-terminator) █

█ Char 01 - Move to start of next line (Carriage return) █

█ Char 02..1F - Output this many spaces █

█ Char 20..FF - Output this character █

█ █

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ INSTRUMENTS █

█────────────────────────────────────────────────────────────────────────█

█ Next come the instruments: █

█ █

█ Offset 00:No. of Adlib instrument that follows (or 0 for no more) █

█ Offset 01..0B:The instrument definition in HSC .INS format: █

█ █

█ ╔══════╦═══════════════════╦═══════════════════════╗ █

█ ║ ║ ║ Adlib Register ║ █

█ ║ Byte ║ Carrier/Modulator ║ (Relative to Chan. 0) ║ █

█ ╠══════╩═══════════════════╩═══════════════════════╣ █

█ ║ 00 │ Carrier │ 23h ║ █

█ ║ 01 │ Modulator │ 20h ║ █

█ ║ 02 │ Carrier │ 43h - Instr. Volume ║ █

█ ║ 03 │ Modulator │ 40h ║ █

█ ║ 04 │ Carrier │ 63h ║ █

█ ║ 05 │ Modulator │ 60h ║ █

█ ║ 06 │ Carrier │ 83h ║ █

█ ║ 07 │ Modulator │ 80h ║ █

█ ║ 08 │ Both │ C0h ║ █

█ ║ 09 │ Carrier │ E3h ║ █

█ ║ 0A │ Modulator │ E0h ║ █

█ ╚══════════════════════════════════════════════════╝ █

█ █

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ ORDER LIST █

█────────────────────────────────────────────────────────────────────────█

█ The Order List follows the instruments: █

█ █

█ Offset 00:Length of Order List (up to 128) █

█ █

█ Offset 01..nn:List of orders, one byte for each line: █

█ █

█ 00..1F:The pattern to play. █

█ 80..FF:The line to jump to minus 80h (Jump Marker) █

█ █

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ PATTERN OFFSET TABLE █

█────────────────────────────────────────────────────────────────────────█

█ For all 32 patterns there is a word in this table which gives the █

█ offset from the start of the file of the pattern data. █

█ If a pattern is empty then the offset is zero. █

█ █

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ PATTERNS █

█────────────────────────────────────────────────────────────────────────█

█ Offset 00:Line number whose notes follow. Bit 7 set means that █

█ this is the last line. Note, lines that have no notes █

█ or effects on do not appear. █

█ █

█ Offset 01....:List of notes for that line: █

█ █

█ Offset 00:Channel no. note appears in (0 to 8). █

█ Bit 7 set means this is the last note for █

█ this line. █

█ Offset 01..02:The note/effect for this channel. █

█ If the effect nibble is non-zero then █

█ Offset 03 contains the parameter, otherwise █

█ the next channel starts at Offset 03. █

█ █

█▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀▀█

█ NOTES █

█────────────────────────────────────────────────────────────────────────█

█ Notes occupy three bytes (two if the effect nibble is zero): █

█ █

█ Byte 00:Bit 7: Bit 4 of the instrument number. █

█ Bit 6..4: Octave of note. █

█ Bit 3..0: Note (1-12) in this order: █

█ C#, D, D#, E, F, F#, G, G#, A, A#, B, C █

█ A zero denotes no note and 15 denotes a █

█ KEY-OFF. █

█ █

█ Byte 01:Bit 7..4: Bits 3 to 0 of the instrument number. █

█ Bit 3..0: The effect nibble (the effect column █

█ in RAD is the hex equivalent of this █

█ nibble). █

█ █

█ Byte 02:The parameter to the effect (if any). █

█ Because parameters only go from 0 to 99, bit 7 is █

█ not used, but may be used in a later version of RAD. █

█ █

█▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄█

╔═══─═══─══─═──═── ─── ─- ─ - · · ∙ ─ ── ─── ─═─═╗

║ │

║ Last Words ║

║ ~~~~~~~~~~ ║

║ │

║ Well we hope you get a lot of use out of RAD. Although RAD started ║

║ as a simple project by me (Shayde) it soon turned into something a lot │

│ more complex and RAD would not be wot it is without the considerable |

║ input from all members of Reality. The graphics and a lot of design :

║ ideas were put forward by Rogue (grafix artist) and Void (musician). ∙

│ Because WE knew want we wanted out of a music development package, RAD ·

║ has evolved into something we are very happy to use for developing our

│ productions. We hope you dudez also discover how to be RADical...!

│

| ·

: RAD woz Kiwi-coded in 100% assembler by SHAYDE .

∙ :

· RAD's graphics were Oz-drawn in 100% ANSI by ROGUE and VOID. │

|

Documentation by SHAYDE, Doc-Ansi by ROGUE. │

│

If you'd like to send us any of your quality tunes made with RAD, ║

please do so, and we'll include them in the next release pack! │

(Our contact address can be found within RAD) ║

║

L8'er dudes! │

║

║

║

· ∙ ─ ─- ── ─────┤ REALITY PRODUCTIONS - April 1995 ├─ ─ ── ───═─══─═══─══╝