NAME

encode — interactive Humdrum encoding from MIDI input

SYNOPSIS

encode [**-r** *controlfile.rc*] [*editfile*]

DESCRIPTION

The **encode** command provides an interactive editor for capturing Humdrum data from a MIDI input device. The operation of **encode** can be tailored by the user to generate a wide range of Humdrum representations — including representations designed by the user.

The **encode** command is limited to encoding information one spine at a time. A typical use of **encode** might be to encode individual musical "parts" or "voices" in a representation such as `**kern`; the user might then use the Humdrum **timebase** and **assemble** commands to construct a full score from the encoded parts. The user must have access to an appropriate MIDI controller connected via a Roland MPU-401 (or compatible) MIDI controller card. Currently, **encode** is available only for IBM PC or compatible hardware running under the DOS operating system.

The **encode** command implements a full-screen interactive text editor — similar to the **vi** text editor. When invoked, the screen is divided into three display regions: a *status window*, a *command window*, and a larger *text window*. The *status window* displays various pieces of information, including the name of the file being edited, the current size of the file (in bytes), the run-control (`.rc`) file used to configure the editor, the number of beats per measure, the output signifier assigned to each beat, the current metronome rate, and mode. The *command window* is used to construct and execute general-purpose commands. The *text window* is used to display and edit the encoded Humdrum text.

The editor has four modes of operation: *edit mode*, *input mode*, *MIDI input mode*, and *command mode*.

EDIT MODE

When **encode** is first invoked, the editor is placed in *edit mode*. If an edit-file has been specified in the command-line, then the beginning of any existing text is displayed in the text window. Edit mode allows the user to move to particular locations in the text and execute certain types of text manipulation instructions. Movement through the text can be achieved by either scrolling the display, or by relative cursor motions within the text window. The cursor position is indicated by a pulsing underscore character.

Four scrolling commands are available. **Control-F** is used to scroll forward in the document by one page. **Control-B** is used to scroll backward by one page. **Control-D** is used to scroll

down by half a page. **Control-U** is used to scroll up by half a page. The **page-down** and **page-up** keys can be used instead of CTRL-F and CTRL-B respectively. If any of these commands is preceded by a typed *number*, then the action is repeated *number* times. Hence 5<*page-down*> will cause the display to advance five pages.

Movement to specific line numbers can be achieved using the upper-case '**G**' command preceded by the desired line number. If no line number is specified, the **G** command causes the cursor to be moved to the end of the file.

Vertical motions of the cursor relative to the displayed text can be controlled by a variety of means. The up (↑) and down (↓) cursor-control keys move the cursor up one line or down one line respectively. Alternatively, the lower-case letters '**k**' and '**j**' may be used for up and down movements — as in the **vi** text editor. If the cursor is at the bottom or top of the screen, cursor movement down or up a line (respectively) will cause the screen to scroll one line as needed. The carriage return or **ENTER** key will cause the cursor to move to the beginning of the next line. Once again, if any of these commands is preceded by a typed *number*, then the action is repeated *number* times.

The upper-case letter '**H**' positions the cursor at the highest line displayed on the screen, whereas the upper-case letter '**L**' positions the cursor at the lowest line displayed on the current screen. Alternatively, the '**home**' and '**end**' keys can be used to move the cursor to the top and bottom respectively. The upper-case letter '**M**' positions the cursor in the middle of the current screen. In all of these commands, the cursor is positioned at the horizontal beginning of the appropriate line.

Horizontal motions of the cursor within a line can be carried out using the left (←) and right (→) arrow keys. Alternatively, the lower-case '**h**' and '**l**' keys can be used as aliases for the left and right cursor motions — as in the **vi** text editor. The **space** bar can also be used to move the cursor to the right. The dollar sign (**$**) causes the cursor to move to the end of the current line. The number zero (**0**) or the caret (^) causes the cursor to move to the beginning of the current line.

Horizontal cursor movements may also be defined by context. Where multiple data tokens are encoded on a single line, the '**w**' command causes the cursor to move forward word-by-word, whereas the '**b**' command causes backward movement word-by-word. A *word* is defined as any character string separated by spaces (or by new lines). In both the '**w**' and '**b**' commands the cursor is placed at the beginning of the appropriate word.

Three forms of simple text deletions are possible in *edit mode*. The character corresponding to the current cursor position can be deleted by typing the lower-case letter '**x**'. The character immediately before the current cursor position can be deleted by typing the upper-case letter '**X**'. The lower-case letter '**d**' will cause the current line to be deleted; line deletions may also be achieved by typing the '**Del**' key. Preceding any of these commands by a typed *number* causes the action to be repeated *number* times. The most recent deletion can be restored or "undone" by typing the lower-case letter '**u**'.

INPUT MODE

The *input mode* allows the manual inputting of typed characters into the text file. This mode is invoked from the *edit mode* by typing either '**a**', '**i**', '**o**', or '**O**'. When any of these commands is invoked, the cursor increases in size and the status window reports the new mode. Typing the letter '**i**' will cause all subsequently typed material to be *inserted* immediately prior to the current cursor position. Typing the letter '**a**' will cause all subsequently typed material to be *appended* following the current cursor position. This can also be achieved by typing the '**Ins**' or '**INSERT**' key. Typing the letter '**o**' will cause a new line to be *opened* following the current line; subsequently typed material will begin on this new line. Typing the upper-case letter '**O**' will cause a new line to be opened immediately *prior* to the current line.

Once the input mode is invoked, typed ASCII characters will be added at the current cursor position until the ESCape or TAB key is pressed. Pressing ESCape or TAB will return **encode** to the *edit mode*. Tabs are explicitly forbidden as input in order to reduce the possibility of creating text that does not conform to the Humdrum syntax.

MIDI INPUT MODE

The most characteristic feature of **encode** is the *MIDI input mode*. The MIDI input mode can be invoked from *edit mode* by typing either of the upper-case letters '**A**' or '**I**'. The letter '**I**' will cause all subsequent MIDI-driven input to be *inserted* immediately prior to the current line. The letter '**A**' will cause all subsequent MIDI-driven input to be *appended* immediately following the current line.

In the *MIDI input mode*, data is entered into the current document according to user-defined mappings between *MIDI events* (such as caused by playing on a MIDI keyboard) and ASCII character strings. Predefined mappings are specified in a *run-time control* (`.rc`) file. When the **encode** command is invoked, it must have access to a `.rc` file containing configuration information. The user can identify a specific `.rc` file on the command line via the **-r** option (see OPTIONS). Alternatively, **encode** will seek a default `.rc` file (named `encode.rc`) residing in the the current directory, or if absent, in the `$HUMDRUM/etc` directory. The run-time control file is essential to the operation of **encode,** and inability to locate a `.rc` file will cause **encode** to abort (see discussion in OPTIONS below).

The run-control file contains a series of definitions mapping MIDI events to output strings. Three classes of MIDI events can be mapped: *key number* (`KEY`), *delta-time* (`DEL`), and *key velocity* (`VEL`). Normally, a `.rc` file contains a number of definitions for each class of event, although some events may not appear at all in some `.rc` files.

By way of example, the run-control instruction:

```
KEY 60 middle-C
```

assigns the key-on event for MIDI key #60 to the string `middle-C`. When in *MIDI input mode*, each key-on event for key #60 will cause the string `middle-C` to be prepared for insertion into the text window. Typically, a number of `KEY` definitions will appear in a

given run-control file — often one definition for each MIDI key (0 to 127).

A second class of MIDI events is key-down velocity (VEL). Key velocities can range between 0 (low key velocity) and 127 (high key velocity). Each VEL mapping specifies a range of values for which a given string will be output. By way of example, the following assignment maps key velocity values between 90 and 127 to a string consisting merely of the apostrophe (the **kern signifier for a staccato note):

```
VEL 90 127 '
```

Given this mapping, key-down velocities within the specified range will cause the apostrophe character to be prepared for insertion into the text window.

A third class of MIDI events is delta-time (DEL). When determing the "duration" of a performed note, the durations of individual key-presses are confound by the articulation. In general, performing musicians are less concerned by the *duration* of individual key-presses, than by the keypress-to-keypress time spans; the elapsed time between one key-onset and the next key-onset provides a better estimate of the nominal musical duration of a note than the actual held duration. The variable DEL contains the *difference between successive key-onset times* — expressed in MIDI clock ticks. Values of DEL may range from 0 upward. For a tempo of 60 beats per minute, inter-onset durations of one second correspond to DEL values of about 100.

The following sample .rc file illustrates a simple run-control file. Notice that a series of DEL ranges have been defined and mapped to **kern- or **recip-type durations. For example, inter-onset times lying between 48 and 80 clock ticks generate the output string '8'; values between 113 and 160 generate the string '4' and so on. Notice that this file restricts the number of possible output "durations" to just five.

```
# Sample .rc file
KEY 60 c
KEY 62 d
KEY 64 e
KEY 65 f
KEY 67 g
KEY 69 a
KEY 71 b
KEY 72 cc

DEL 48 80 8
DEL 81 112 8.
DEL* 113 160 4
DEL 161 224 4.
DEL 225 320 2

VEL 90 127 '

ORDER DEL KEY VEL
```

Any records in the run-control file beginning with a # character are treated as comments. Empty lines are ignored.

The effect of the above run-control file can be illustrated by example. Imagine that **encode** received two key-on events (key #60 followed by key #62), where the first key exhibited a velocity value of 94 and the inter-onset time (DEL) was 100. The first key (#60) would be mapped to the string 'c'; the delta-time would be mapped to the string '8.'; and the key-velocity (VEL) would be mapped to the apostrophe. At the moment of the key-onset for key #62, these three strings would be amalgamated according to the ORDER instruction (DEL first, KEY second, and VEL third) — producing the output string: `8.c'`

Notice that **encode** outputs assembled strings only when the *next* key-on event occurs. This means that the text display is always one "note" behind the performer. Note that if musical durations are based on key inter-onset times, it is impossible to output a note prior to the onset of the next note. The last note in a buffer can be flushed by typing the ESCape key. (The timing of the last note is based on the DEL between key-onset and the moment of pressing ESCape.)

In addition to mapping velocities, inter-onset times, and key numbers, run-control files can define a number of other attributes. The MIDI channel number attended to by **encode** can be set by the RECEIVE instruction. Any one of 16 channels (1-16) can be selected. A default channel 1 is assumed if no RECEIVE instruction is present in the run-control file.

The **encode** command has a built-in metronome for assisting real-time encoding. The metronome sends commands to the MIDI instrument generating metronome tones. Two types of tones are generated — tones marking each beat, and tones marking the beginning of each measure. The metronome rate (in beats per minute) is set by the TEMPO command. The beat is specified in two ways. The default beat is indicated by the presence of an asterisk

following *one* of the DEL instructions. In the above example, the signifier '4' is assigned to the default beat. Apart from the default beat, the beat may also be explicitly assigned using the BEAT instruction. This instruction is followed by a single argument identifying the output *signifier* intended to coincide with each metronome beat. For example, `BEAT 4.` would set the beat in the above example to the dotted quarter, rather than the quarter. Note that the specified signifier in the BEAT command must correspond to one of the existing signifiers defined using a DEL instruction.

The following table lists all of the types of instructions permitted in a run-control file.

# text	unexecutable comment
BEAT *string*	set beat to **DEL** whose output signifier is *string* beat
BUFFER *n text*	define (potentially multi-line) text buffer # *in* (0-9)
DEL *min max string*	key inter-onset times between *min* and *max* clock ticks cause *string* to be output
KEY *n string*	MIDI key-on #*n* causes *string* to be output
METER *n*	define number of beats per measure as *n*
METRE *n*	same as **METER**
MM *on*|*off*	switch metronome *on* or *off*; default is *on*
ORDER *codeword1 codeword2 ...*	define order of string outputs, where codewords are selected from: BUFFER, DEL, KEY, SRING, VEL
RECEIVE *n*	define the MIDI channel from which data is accepted
STRING *n text*	define string constant # *in* (0-9)
TEMPO *n*	set metronome to *n* beats per minute
VEL *min max string*	key-down velocities between *min* and *max* cause *string* to be output

Definition types for **encode**

The `BUFFER`, `DEL`, `KEY`, `STRING`, and `VEL` instructions can be repeated multiple times within the `.rc` file. All other instructions (`BEAT`, `METER`, `MM`, `TEMPO`, `ORDER` and `RECEIVE`) should appear only once in the `.rc` file. The `BEAT` and `TEMPO` instructions cannot appear in the `.rc` file until after the default beat (`DEL*`) has been defined.

COMMAND MODE

The **encode** *command mode* allows a number of general-purpose commands to be executed — such as editing a specified file, changing a default mapping, or auditioning the encoded material. The *command mode* can be invoked from the **encode** *edit mode* by typing the colon character (:).

Each command is formulated in the *command window* and launched by pressing the ENTER key. After execution, **encode** is returned to *edit mode*.

Most commands consist of a single character; some commands require one or more parameters.

The '**w**' command causes the current text to be written to disk. If there is currently no active filename, then an error is displayed indicating that **encode** is unable to write a file without knowledge of the filename. The command '**w** *filename*' will cause the current contents to be written to the file *filename*. If **encode** was invoked with a specified filename, then that filename is active throughout the session.

If the user attempts to write to an existing file (that was not specified when **encode** was invoked), then an error message is issued. Overwriting an existing file can be achieved by appending an exclamation mark following the write instruction — as in '**w!** *filename*.'

The '**q**' command causes **encode** to terminate. If the current file has been modified without writing to disk, then a warning will be displayed and the quit instruction ignored. Appending an exclamation mark ('**q!**') will cause **encode** to terminate without saving any recent modifications.

Note that the *quit* and *write* commands can be combined as a single instruction — '**wq**'.

The '**r** *filename*' command causes **encode** to read the file *filename* into the text, beginning at the line following the current cursor position.

The '**v**' command causes **encode** to spawn a **vi** text editing session — importing the current **encode** text. The **vi** text editor provides text manipulation capabilities, including searching, substitution, and macro-instruction facilities not available in **encode.** (Refer to the UNIX **vi** reference document for further information.) When the **vi** session is closed, the edited text file is returned to the **encode** session.

The '**m**' command invokes a Humdrum pipeline that is suitable for auditioning text data conforming to either the `**kern` or `**MIDI` representation formats. Specifically, '**m**' causes the current text to be passed to the Humdrum pipeline: `midi -c | perform`. Any `**kern` data will be translated to `**MIDI` data and sent to the `perform` command. The user can then interactively proof-listen or audition the encoded data. Refer to the **perform (4)** for information regarding the types of interactive commands available during proof-listening. The **perform** command is terminated when the end-of-file is reached, or if the user presses either the ESCape key or the letter 'q'. In either case, control is returned to **encode.**

The '**b**' command is used to read buffer text defined in the `.rc` file. Up to ten numbered buffers (0-9) can be defined. The command:

```
b 1
```

will cause any text denoted `BUFFER 1` to be output following the current cursor position. Use of the upper-case '**B**' rather than '**b**' causes the buffer contents to be inserted prior to the current line rather than following the current line.

Buffer zero (0) has a special status. When the **encode** command is invoked, if the current text is empty (i.e. empty file or no filename specified), then the contents of `BUFFER 0` are automatically inserted into the text. This provides a convenient way to import header

information for a newly encoded file.

The '**rc** *filename*' command causes **encode** to use a different run-control file *filename*. This allows the **encode** environment to be entirely reconfigured without interrupting the encoding session. This command can prove useful, for example, when the music being encoded changes key.

The **set** command can be used to define (or redefine) any parameters permitted in a `.rc` file. For example, the tempo may be changed, the metronome turned-off, the metronome beat redefined, a string variable assigned, or a specific key re-mapped, e.g.

```
set TEMPO 92
set MM off
set BEAT 4.
set STRING 3 !! Variation No. ...
set KEY 60 C4
```

For the `BUFFER` command, **set** defines an *additional* buffer record, rather than replacing any existing buffer definitions.

Note that run-control keywords, such as `TEMPO`, may be spelled using either upper-case or lower-case.

Note that due to the small size of the command window, especially long command lines may wrap around within the window. This wrap-around has no affect on the command operation.

SUMMARY

The various built-in commands in **encode** are summarized in the following table.

EDIT MODE

CTRL-F	scroll forward one page
CTRL-B	scroll backward one page
CTRL-D	scroll down by half a page
CTRL-U	scroll up by half a page
<PAGE-DOWN>	scroll forward one page
<PAGE-UP>	scroll backward one page
↓	move cursor down one line
↑	move cursor up one line
j	move cursor down one line
k	move cursor up one line
<ENTER>	move cursor to the beginning of the next line
H	move cursor to the top of the display
M	move cursor to the middle of the display

L	move cursor to the bottom of the display
<HOME>	move cursor to the top ~~of~~ the display
<END>	move cursor to the bottom of the display
→	move cursor one character to the right
←	move cursor one character to the left
l	move cursor one character to the right
h	move cursor one character to the left
<SPACE>	move cursor one character to the right
x	delete character at current cursor position
X	delete character immediately preceding currrent cursor position
d	delete current line
u	undo most recent deletion or insertion command
<INSERT>	insert text prior to current cursor position (invoke INPUT MODE)
i	insert text prior to current cursor position (invoke INPUT MODE)
a	insert text after to current cursor position (invoke INPUT MODE)
o	insert text beginning with the next line (invoke INPUT MODE)
O	insert text beginning with the previous line (invoke INPUT MODE)
A	invoke MIDI INPUT MODE; insert data beginning with the next line
I	invoke MIDI INPUT MODE; insert data beginning with the previous line
<*number*>	repeat ensuing command <*number*> times

INPUT MODE

ESC	return to EDIT MODE
<TAB>	return to EDIT MODE

MIDI INPUT MODE

ESC	complete last MIDI event and return to EDIT MODE

COMMAND MODE

b *n*	append run-control buffer number *n* following current cursor position
B *n*	insert run-control buffer number *n* before current cursor position
m	invoke interactive proof-listening for `**kern` or `**MIDI` text
set *rc-command*	set or reset a `.rc` mapping
q	quit **encode**
q!	quit **encode** without saving modifications since last write
r *filename*	read input file *filename* at current cursor position
rc *controlfile*	use run-control file *controlfile* rather than current `.rc` file
s *n*	append run-control string number *n* following current cursor position
S *n*	insert run-control string number *n* before current cursor position
v	spawn **vi** text editing session using current text
w [*filename*]	write file *filename* to disk; defaul filename is current file
w! *filename*	overwrite existing file *filename*
wq	write current file and quit

OPTIONS

The **encode** command provides the following options:

-h displays a help screen summarizing the command syntax
-r *file.rc* invoke using the run-control file *file.rc*

Options are specified in the command line.

The **-r** option permits the user to identify a specific run-control file to configure the **encode** editor. If this option is omitted, **encode** will seek a default run-control file named `encode.rc` in the current directory, or in the directory `$HUMDRUM/etc` if not present in the current directory. If the option is specified, **encode** will search the current directory for the specified run-control file. If this search fails to locate the file, **encode** will search `$HUMDRUM/etc`. If this fails, **encode** will treat the input filename as an absolute file path. If this fails, encode will issue an error message indicating that it failed to locate the specified run-control file.

FILES

A number of predefined `.rc` files are maintained in the `$HUMDRUM/etc` directory. Exploration is encouraged. The default file is `$HUMDRUM/etc/encode.rc`.

PORTABILITY

DOS 2.0 and up. An appropriate MIDI controller (such as a keyboard synthesizer) connected via a Roland MPU-401 (or compatible) MIDI controller card. The `vi` text editor must be available in order to invoke the 'v' edit command.

WARNINGS

Unlike the UNIX **vi** text editor, only a single '**d**' is required in edit mode to delete a line (rather than two d's). Experienced **vi** users should be careful when deleting lines.

Note that the `BEAT` and `TEMPO` instructions cannot appear in the `.rc` file until after the default beat (`DEL*`) has been defined.

SEE ALSO

assemble (4), **cleave** (4), **encode.rc** (5), **humdrum** (4), ****kern** (2), ****MIDI** (2), **midi** (4), **num** (4), **perform** (4), **proof** (4), **record** (4), **timebase** (4), **vi** (UNIX)

REFERENCES

Use the the Music Quest Inc. MIDI library functions is gratefully acknowledged.

NOTE

Especially long input lines may exceed the size of the text window. Although the characters to the right of the text screen may not be visible, they remain encoded in the file.