

keymap

COLLABORATORS

	<i>TITLE :</i> keymap		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		July 18, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	keymap	1
1.1	keymap.doc	1
1.2	keymap.library/AskKeyMapDefault	1
1.3	keymap.library/MapANSI	1
1.4	keymap.library/MapRawKey	3
1.5	keymap.library/SetKeyMapDefault	5

Chapter 1

keymap

1.1 keymap.doc

AskKeyMapDefault ()	MapRawKey ()
MapANSI ()	SetKeyMapDefault ()

1.2 keymap.library/AskKeyMapDefault

NAME

AskKeyMapDefault -- Ask for a pointer to the current default
keymap. (V36)

SYNOPSIS

```
keyMap = AskKeyMapDefault ()

struct KeyMap *AskKeyMapDefault ( VOID );
```

FUNCTION

Return a pointer to the keymap used by the keymap library for
MapRawKey and MapANSI when a keymap is not specified.

RESULTS

keyMap - a pointer to a keyMap structure. This key map is
guaranteed to be permanently allocated: it will remain in
memory till the machine is reset.

BUGS

The keymap.h include file should be in the libraries/ or perhaps
resources/ directory, but is in the devices/ directory for
compatability reasons.

SEE ALSO

devices/keymap.h, keymap.library/SetKeyMapDefault (),
console.device ...KEYMAP functions

1.3 keymap.library/MapANSI

NAME

MapANSI -- Encode an ANSI string into keycodes. (V36)

SYNOPSIS

```
actual = MapANSI( string, count, buffer, length, keyMap )
D0          A0      D0      A1      D1      A2

LONG MapANSI( STRPTR, LONG, STRPTR, LONG, struct KeyMap * );
```

FUNCTION

This console function converts an ANSI byte string to the code/qualifier pairs of type IECLASS_RAWKEY that would generate the string and places those pairs in a buffer. A code/qualifier pair is a pair of bytes suitable for putting in the ie_Code and low byte of ie_Qualifier, and for subsequent events, shifted to the ie_Prev1DownCode/ie_Prev1DownQual then ie_Prev2DownCode/ie_Prev2DownQual pairs for any dead or double dead key mapping.

INPUTS

string - the ANSI string to convert.
 count - the number of characters in the string.
 buffer - a byte buffer large enough to hold all anticipated code/qualifier pairs generated by this conversion.
 length - maximum anticipation, i.e. the buffer size in bytes divided by two (the size of the code/qualifier pair).
 keyMap - a KeyMap structure pointer, or null if the default key map is to be used.

RESULT

actual - the number of code/qualifier pairs in the buffer, or negative to describe an error (see below).

EXAMPLE

```
...
#include <devices/inpotevent.h>

#define STIMSIZE      3      /* two dead keys, one key */
unsigned char rBuffer[STIMSIZE*2];
...
KeymapBase = (struct Library *) OpenLibrary("keymap.library", 0);
...
event.ie_NextEvent = 0;
event.ie_Class = IECLASS_RAWKEY;
event.ie_SubClass = 0;

/* prove keymap code completeness and MapANSI reversibility */
for (code = 0; code < 256; code++) {
    buffer[0] = code;
    actual = MapANSI(buffer, 1, rBuffer, STIMSIZE, 0);
    r = rBuffer;
    event.ie_Prev2DownCode = 0;
    event.ie_Prev2DownQual = 0;
    event.ie_Prev1DownCode = 0;
    event.ie_Prev1DownQual = 0;
```

```

switch (actual) {
    case -2:
        printf("MapANSI internal error");
        goto reportChar;
    case -1:
        printf("MapANSI overflow error");
        goto reportChar;
    case 0:
        printf("MapANSI ungeneratable code");
        goto reportChar;

    case 3:
        event.ie_Prev2DownCode = *r++;
        event.ie_Prev2DownQual = *r++;
    case 2:
        event.ie_Prev1DownCode = *r++;
        event.ie_Prev1DownQual = *r++;
    case 1:
        event.ie_Code = *r++;
        event.ie_Qualifier = *r;

        actual = MapRawKey(&event, buffer, BUFFERLEN, 0);
        if ((actual != 1) || (buffer[0] != code)) {
            printf("MapANSI not reversible");
            for (i = 0; i < actual; i++)
                ReportChar(buffer[i]);
            printf(" from");
reportChar:
            ReportChar(code);
            printf("\n");
        }
    }
}
...

```

ERRORS

if actual is 0, a character in the string was not generatable from the keyMap.
 if actual is -1, a buffer overflow condition was detected.
 if actual is -2, an internal error occurred (e.g. no memory)

SEE ALSO

devices/inputevent.h, devices/keymap.h

1.4 keymap.library/MapRawKey

NAME

MapRawKey -- Decode single raw key input event to an ANSI string. (V36)

SYNOPSIS

```

actual = MapRawKey(event, buffer, length, keyMap)
D0          A0      A1      D1      A2

```

WORD MapRawKey(struct InputEvent *, STRPTR, WORD,

```
struct Keymap * );
```

FUNCTION

This console function converts input events of type IECLASS_RAWKEY to ANSI bytes, based on the keyMap, and places the result into the buffer.

INPUTS

event - an InputEvent structure pointer. The event list is not traversed.
 buffer - a byte buffer large enough to hold all anticipated characters generated by this conversion.
 length - maximum anticipation, i.e. the buffer size in bytes.
 keyMap - a KeyMap structure pointer, or null if the default key map is to be used.

RESULT

actual - the number of characters in the buffer, or -1 if a buffer overflow was about to occur.

EXAMPLE

```
...
#define BUFFERLEN      80      /* length of longest expected mapping */
char buffer[BUFFERLEN];
struct InputEvent ie;
...
KeymapBase = OpenLibrary("keymap.library", 0);
...
ie.ie_Class = IECLASS_RAWKEY;
ie.ie_SubClass = 0;
for (;;) {
    WaitPort(window->UserPort);
    while (im = (struct IntuiMessage *) GetMsg(window->UserPort)) {
        switch (im->Class) {
            case RAWKEY:
                ie.ie_Code = im->Code;
                ie.ie_Qualifier = im->Qualifier;
                /* recover dead key codes & qualifiers */
                ie.ie_EventAddress = (APTR *) *im->IAddress;
                actual = MapRawKey(&ie, buffer, BUFFERLEN, 0);
                for (i = 0; i < actual; i++)
                    ReportChar(buffer[i]);
                break;
            ...
        }
    }
    ...
}
```

ERRORS

if actual is -1, a buffer overflow condition was detected.
 Not all of the characters in the buffer are valid.

SEE ALSO

devices/inputevent.h, devices/keymap.h

1.5 keymap.library/SetKeyMapDefault

NAME

SetKeyMapDefault -- Set the current default keymap. (V36)

SYNOPSIS

```
SetKeyMapDefault(keyMap)
```

```
void SetKeyMapDefault( struct KeyMap * );
```

FUNCTION

A pointer to key map specified is cached by the keymap library for use by MapRawKey and MapANSI when a keymap is not specified.

INPUTS

keyMap - a pointer to a keyMap structure. This key map must be permanently allocated: it must remain in memory till the machine is reset. It is appropriate that this keyMap be a node on the keymap.resource list.

BUGS

The keymap.h include file should be in the libraries/ or perhaps resources/ directory, but is in the devices/ directory for compatability reasons.

SEE ALSO

devices/keymap.h, keymap.library/AskKeyMapDefault(), console.device ...KEYMAP functions