X Input Device Extension Library

Mark Patrick, Ardent Computer
George Sachs
Hewlett-Packard

X Input Device Extension Library

!

!"

$ © %&% %%" %% # '$ (# '$)

') $ ' *))' * $) $) *
$ + + $ $ ' $$ $! (
' $)) * $) $ * * ')' ! , $ + -
-) .$ '$!

$ / %&% %%" %% %%0 #)'

' * * $ $ * *)
' * 1 22 * 334 *))) '
) $ ' * ' $))) 5 $ * * $ ' $
' * *)) 6 * 7

8 + $ $ ') $) $ *
* !

8 9 :;8<(9 , : ,=9= >(, ? <,8 :@8 <((A8B :; (AB C,A= 9 9 : , D,9= ,A#D@=,A
E@8 A:8 D, ,89= 8: 8 9 <((A8,9 :; 9 # (A8(E,D,8B :,8A9 :: ((8,#@D(@ : 9 (A=
A:A,A; ,A 9 9A8! ,A A: 9 9A8 (DD 8 9 #:A : 8,@ E9 D,(ED9 ;: (AB #D(, =((9 :
:8 9 D,(E,D,8B < 98 9 ,A (A (#8,:A :; #:A8 (#8 8: 8 : :8 9 <, 9 (, ,A ; : :@8 :;
: ,A #:AA9#8,:A <,8 8 9 :;8<(9 : 8 9 @ 9 : :8 9 =9(D,A ,A 8 9 :;8<(9!

9. $ ' * #)') +
$ ') *) $) F * ' #)'!

< ' ' * 8 :$)$!

Table of Contents

! , $) 9. !!!!

Chapter 1. Input Extension

Overview

8)' . + ! 8 $) $ * .
)$$) * $) + $
+ * $! 8 * + + +
* $) . ! 8 * $ $ % "
HH< ;) II HH9+ II HH9+
;) II * HH # D) , * II '))
$) + . !

Design Approach

8 $$ * . * *) +)
*) + ! 8 . $) + +
+)) * ' * ' $) +
+ ! 8 *) + ') * + *)$$
$ * ' ')
$ + $) + !

Core Input Devices

8 + $)$$ $) + 7 $!
8 $ + '6 *) !; ') '
* ' $ ' ! ')
) *) * ! 8 '$ +
)) $ * $! @
.$ *))) *) *
!

8) $) $ ' !

8 $ *)'
$) + (KeyPress , KeyRelease , ButtonPress ,
ButtonRelease , MotionNotify) !(
$) + * $) +
* ! 8 $) .
+ *) * $) + +
$ G) . $ * G) ! 8 G) '
** F * + * ' . + ! .$
HH9+ F # !II

* $ +) + +
* '$ ' $! ;) * $ '
$ +) + !

Extension Input Devices

8 $) . $) +
$!, $ ' $) * ' + $
* ' $ * ' + ! , $) + * '

+ * . $ (DeviceKeyPress , DeviceKeyRelease , Device-
ButtonPress , DeviceButtonRelease , DeviceMotionNotify , 4
+ * + * ' $ ' * ' **
$) +) !

9. $) + ' F F * + J0
+ ! 9. $) + ') + '
F +) * ' G) * + !
8 + * ') $ * '
' 9+ !

($) +) ' ') .
$) + ! 9. $) + ' + F ' F
') ' $ F ' . *' ! ' $
+ '+' * ' $ +) $) $! (+)
$ ' * ' * + . $) + ') $
' *' * ' 1) + 4!

8 . " $ * $) + '
*)) ! 8 $ G) * $ * *
$) + F * ' ! + '$ '
' * $) +) $ G) !

(. $) + '
*) ! 8 + * +
+) *))
! , ' * $!
D ' ' .$ *) ! ,* .$
*) *) ' * $!

Input Device Classes

' * $) . G) + $) +
*) ! 8 * $) + *
') ' * *) ! 8 * $)
+) * 7

C9B 8 + $ + !

E@88:A 8 + $) + !

(D@(8 + $ +) ' + !
8:

: , , 8 + $ $.' + !
8B

;:#@ 8 + *) !

;99= 8 +)$$ * !
E(#C

(' *)) ! ;))$$ ') $ $)
) XListInputDevices *) + $)
+ F) $) ! # $ ')
*))) ' * '

'$!,) * .
) *) '$!

Using Extension Input Devices

($) +) *)
* * !($ G) * G))
' * 7

K XListInputDevices * + $) + !; ' * '
) G) ' $) +
+ !; $ * XListInputDevices G)
HHD (+ =+ !II

K XOpenDevice G) + $ + * !
8 G)) XDevice))) ' $) .
G) * $ * + !; $ * XOpen-
Device G) HH9 = 9. =
+ !II

K = ' + $ + $) .
+ * ' + !8 + '
' $ + * .'$ DeviceKeyPress .
; $ * ' HH 9.
=+ 9+ !II

K XSelectExtensionEvent + * ' + ! ;
$ * XSelextExtensionEvent G)
HH 9. =+ 9+ !II

K XNextEvent + . + + !8 XNextEvent *)
$ + !

: G) * *) . +
) ' * '$$ $ $ * $) . +
' * ' . + $) . +
!8 *) * !

Library Extension Requests

9. $) + $ ')) *
$ G) !8 * G) $ + . !#
)) * *) ' *) * <X11/
extensions/XI.h> <X11/extensions/XInput.h>
)' ($$.(!

8) NoSuchExtension * . G) ' +
)$$ $) . !

, $) . G)) $
+ !

Window Manager Functions

8) * , $) 9. < $ 7

K # +

K 9+ F

K 9. +

K +

K +)

K 8 +

K # + *)

K # + *

K $) +

K # +

K #) '$$

K : * +

Changing the Core Devices

8 *) $ + $ +) $
!

@ *) ' * + !
8 $ + ' + **)' *) * '
+ ' + **)' *
$ ** * ! # $ ' ')
$! ; .'$ $ ')
)' *) $ +
)))' +) + !
+))) $ ' + '
$ $ ' ' * ChangeDeviceNotify
+ G) !

8 *) $ +
XChangeDeviceNotify + MappingNotify + !8 $ * +
' $ + !8 * +
) * G) !

8 G) *) AlreadyGrabbed * $ * +
+) $ ' !8 *)
GrabFrozen * + * F + * !

8 G) * BadDevice * $ * + +
$ +) $ + XOpenDevice ,)$$ +
+ '$ ' !

: +) *) $ * +
+)) $ ChangeDevice G))

+ ' !8 ' * +
) !('$) XCloseDevice G)
+ * BadDevice !

8 $ +)) XChangeKeyboardDevice *) !8 $ * + '))$$ $) Keys 1
$ ListInputDevices G) 4 G) * BadMatch
!

int **XChangeKeyboardDevice**(**display*, **device*);

$ * + !

$ * + !

,*) XChangeKeyboardDevice) Success .(ChangeDeviceNotify + G) * NewKeyboard
+ !(MappingNotify + G) * MappingKeyboard
!8 G) + '
' + . $) + !8 *) *
' *) * !

XChangeKeyboardDevice AlreadyGrabbed , BadDevice , BadMatch , GrabFrozen !

8 $ +) $) XChangePointerDevice *) !8 $ * + '))$$ $) Valuators 1
$ XListInputDevices G) 4 $. *'
G) * BadMatch !,* $ * + $ '
. $ * .. .)') != * ' +) + !

,* $ * + $) $ * ' + '
$ ') +) $ G)
* BadDevice !

int **XChangePointerDevice**(**display*, **device*, *xaxis*, *yaxis*);

$ * + !

$ * + !

$ * F . * .)
. . * $ + !

$ * F . * .)
. * $ + !

,*) XChangePointerDevice) Success .(ChangeDeviceNotify + G) * NewPointer
+ !(MappingNotify + G) * MappingPointer
!8 G) + ' $ $
' + . $) + !

XChangePointerDevice AlreadyGrabbed , BadDevice , BadMatch , GrabFrozen !

Event Synchronization and Core Grabs

,'$ ' * $) . G) . * ' * +
F * G) ! 8
' * F $) + G) !

8 G) G) $ L' L')' ! 8
' * ' $) . ! ; XGrabPointer
XGrabButton G) $ L' F * $
+ L' F * $) + !
; XGrabKeyboard XGrabKey G) $ L'
F * $) + . $ L'
F * ! <) * G)
F * . + ' $ * *
+ !

Extension Active Grabs

(+ * . +)$$ + XGrabDevice *)
' +) XGrabKeyboard *)
. $. $) + $ *) $ ' ! 8
XUngrabDevice *) $ +) + * . +
!

+ *) . +)$$ +
XGrabDeviceButton XGrabDeviceKey *) ! 8 $ +
+ XUngrabDeviceKey XUngrabDeviceButton *) !

8 . +) XGrabDevice *) ! 8 + ') +
$ +) $) XOpenDevice *) !

int **XGrabDevice**(*display, *device, grab_window, owner_events, event_count, *event_list, this_device_mode, other_device_mode, time);

- - $ * + !

$ * + !

$ * ,= *
+ $ * + !

$ * +) * True False .

$ *)' * ' + L
!

$ * $ * +
+ + ! 8
+ ') +)
+ !

*) $ * + * '
+ ! B) $ * 7 GrabMod-
eSync GrabModeAsync .

*) $ * + * '
+ ! B) $ * 7 Grab-
ModeSync GrabModeAsync .

$ * '! 8 ' ' '$
.$ ' CurrentTime .

XGrabDevice + . $) + DeviceFo-
cusIn DeviceFocusOut + ! ;) $) + * ' +
$! 8 *) + $ +) +
* + !

8 + L $ ' $ * + ! 8
+ + + ! ,* L +
False , $) + * ' + $ $ L
$ * $ * + L ! ,* L + True , *
+) ' $ $ '
! : + $ $ L
$ * $ * + L !

8 L + L')' *) $ * + * '
+ L + L')' *) $ *
$) + * ' + !

K ,* L + L')' GrabModeAsync , + + $
) ' M * +) * F $
* + +)' ! ,* L + L')' GrabModeSync ,
* + 1 $$ 4 $$ * F
*) + + +)
) XAllowDeviceEvents) + !
( ) + $) + + * F M '$
G)) * $!

K ,* L + L' GrabModeAsync , + $ * ' $)
+) ** + * ! ,* L + L' GrabMod-
eSync , * + . $ + 1 $$
4 $$ * F *) + +)
) XAllowEvents XAllowDeviceEvents
) + ! ( ) +
+ * F M '$ G)) * $!

XGrabDevice * * 7

K ,* + + ') AlreadyGrabbed .

K ,* L +) GrabNotViewable .

K ,* $ * ' ' * $ * +
) + ') GrabInvalidTime .:
' * $ * + $ * ' CurrentTime
$) + '!

K ,* + * F + *) GrabFrozen .

,* + +
** + ! ($ +

! ,* + * F + * G)
!

XGrabDevice BadClass , BadDevice , BadValue , BadWindow
!

8 * . +) XUngrabDevice *) !

int **XUngrabDevice**(**display*, **device*, *time*);

$ * + !

$ * + !

$ * '! 8 ' ' '$.
$ ' CurrentTime .

XUngrabDevice . $) + G))
+ * * ' XGrabDevice XGrabDeviceKey .,*
+ * F XUngrabDevice '!8 *)
+ G)) + * $ * '
+ ') + '! ,
DeviceFocusIn DeviceFocusOut + !8 +) '
$ * ' XUngrabDevice * + * + + '
+ * ') !

XUngrabDevice BadDevice !

Passively Grabbing a Key

8 $ + . +) XGrabDeviceKey . 8
+ ') + $ +) $) XOpenDevice *)
G) * BadDevice ! ,* $ * +)$$
$) Keys , G) * BadMatch !

int **XGrabDeviceKey**(**display*, **device*, *keycode*, *modifiers*, **modifier_device*, *grab_window*, *owner_events*, *event_count*, **event_list*, *this_device_mode*, *other_device_mode*);

$ * + !

$ * + !

$ * * !
B) $ AnyKey .

$ * * ' !8 '
) + : * ' 7 ShiftMask ,
LockMask , ControlMask , Mod1Mask , Mod2Mask ,
Mod3Mask , Mod4Mask , Mod5Mask .

B) $ AnyModifier , G) +
) G) * $ '
* ' 1) ' *
' * 4!

$ * + ' *) !
,* A@DD $ *)
' * L + !

$ * ,= *
+ $ * +!

$ * +) * True False .

$ *)' * ' + L
!

$ * $ * +
+ +!

*) $ * + * '
+ ! B) $ * 7 GrabMod-
eSync GrabModeAsync .

*) $ * + * '
+ ! B) $ * 7 Grab-
ModeSync GrabModeAsync .

XGrabDeviceKey) XGrabKey *) !, .$
$ + * . + ! 8 XGrabDeviceKey *)
$ + + ! # G) *))

K ,; + $ * * ' *
$ $ * ' *
$ * ' * + 1 4

K (A= ' *

K (A= 9,8 9 * 1 4 *)
* *) $

K (A= $ + ' + ' .
*

K 8 9A + + * XGrabDevice , +
' ' $ 1 ' De-
viceKeyPress + 4 DeviceKeyPress + $!

8 $ * ')' * XGrabDevice . 8 +
') ' * + $
* 1 $ * * ' * 4!

A * + 1 ' * $ 4'
$ * + + $ * F !

(' * * AnyModifier G) +) G) * $ '
* ' 1) ' * ' * 4!, G)
' * $ * +) !(* AnyKey G) +
) G) * $!: ')
$ * ' L '.L * ')
XListInputDevices *) !,* XGrabDeviceKey
BadValue !

XGrabDeviceKey BadAccess * ')
XGrabDeviceKey ' + ' ' !
<) AnyModifier AnyKey , G) * '$ +
BadAccess * *
* ' !

XGrabDeviceKey) Success)$) *) '$ * G) !

XGrabDeviceKey BadAccess , BadClass , BadDevice , BadMatch , BadValue , BadWindow !

8 $ + * . +) XUngrabDeviceKey .

int **XUngrabDeviceKey**(*display, *device, keycode, modifiers, *modifier_device, ungrab_window);

$ * + !

$ * + !

$ * *
) ! B) $
AnyKey .

$ * * ' ! 8 '
) + : * ' 7 ShiftMask , LockMask , ControlMask , Mod1Mask , Mod2Mask , Mod3Mask , Mod4Mask , Mod5Mask .

B) $ AnyModifier , G) +
)) G) * $
' * ' 1) ' *
' * 4!

$ * + ' *) !
,* A@DD $ *)
' * L + !

$ * ,= *
+ $ * +!

XUngrabDeviceKey) XUngrabKey *) !,
.$ $ + * . $) + !

XUngrabDeviceKey BadAlloc , BadDevice , BadMatch , BadValue , BadWindow !

Passively Grabbing a Button

8 $ + *) . +) XGrabDeviceButton . 8 $ * + ') + $ +) $)
XOpenDevice *) G) * BadDevice !,* $
* +)$$ $) Buttons , G) * BadMatch !

int **XGrabDeviceButton**(**display*, **device*, *button*, *modifiers*, , *grab_window*, *owner_events*, *event_count*, **event_list*, *this_device_mode*, *other_device_mode*);

$ * + !

$ * + !

$ * *) !
B) $) AnyButton .

$ * * ' ! 8 '
) + : * ' 7 ShiftMask , LockMask , ControlMask , Mod1Mask , Mod2Mask , Mod3Mask , Mod4Mask , Mod5Mask .

B) $ AnyModifier , G) +
) G) * $ ' *
' 1) ' * '
* 4!

$ * + ' *) !
,* A@DD $ *)
' * L + !

$ * ,= *
+ $ * + !

$ * +) * True False .

$ *)' * ' + L
!

$ * * +
+ + $!

*) $ * + * '
+ ! B) $ * 7 GrabModeSync GrabModeAsync .

*) $ * + * '
+ ! B) $ * 7 GrabModeSync GrabModeAsync .

XGrabDeviceButton) XGrabButton *) !,
.$ $ + *) . $) + ! E) +
. +) $ * G) ! ;
' * L $ ' ! 8 + ') + $ +)
$) XOpenDevice *) !

8 XGrabDeviceButton *) $ + + ! #
G) *))

K ,; + $ *) $
$ * ' * 1) ' *
4

K (A= 9,8 9 * 1 4 *) :
* *) $

K (A= $ + ' +) 5 ' .
*

K 8 9A + + * XGrabDevice , '
') $ 1 ' Device-
ButtonPress + 4 DeviceButtonPress + $!

8 $ * ')' * XGrabDevice . 8 +
') ' * +)
1 $ * * ' * 4!

A * + 1 ' * $ 4'
$ * + + $ * F !

(' * * AnyModifier G) +) G) * $ '
* ' 1) ' * ' * 4! , G)
' * $ * +) !( ) * AnyBut-
ton G) +) G) * $) !:
G) $ *) $) !

XGrabDeviceButton BadAccess * ')
XGrabDeviceButton ' +) ' '
!<) AnyModifier AnyButton , G) * '$
+ BadAccess *
* * ' !

XGrabDeviceButton BadAccess , BadClass , BadDevice , Bad-
Match , BadValue , BadWindow !

8 $ + *) . +) XUngrabDevice-
Button .

int **XUngrabDeviceButton**(**display*, **device*, *button*, *modifiers*, **modifier_device*, *ungrab_window*);

$ * + !

$ * + !

$ * *))
!B) $) AnyButton .

$ * * ' !8 '
) + : * ' 7 ShiftMask ,
LockMask , ControlMask , Mod1Mask , Mod2Mask ,
Mod3Mask , Mod4Mask , Mod5Mask .

B) $ AnyModifier , G) +
)) G) * $
' * ' 1) ' *
' * 4!

$ * + ' *) !
,* A@DD $ *)
' * L + !

$ * ,= *
+ $ * +!

XUngrabDeviceButton) XUngrabButton*) !,
.$ $ + *) . + !8 + ') +
$ +) $) XOpenDevice*) BadDevice
) !

(' * * AnyModifier G)+) G) * $ '
* ' 1) ' * ' * 4!

XUngrabDeviceButton BadAlloc , BadDevice , BadMatch , Bad-
Value , BadWindow !

Thawing a Device

8 *) + $ + * F) XAl-
lowDeviceEvents .

int **XAllowDeviceEvents**(**display*, **device*, *event_mode*, *time*);

$ * +!

$ * +!

$ * + '!B) $ *
7 AsyncThisDevice , SyncThisDevice ,
AsyncOtherDevices , ReplayThisDevice , Asyn-
cAll , SyncAll .

$ * '!8 ' ' '$.
$ ' CurrentTime .

XAllowDeviceEvents 'G)) + *) +
* F!, ** * $ * ' ' *
' + * + * $ * '
) + '!8 * $)
$)$ + L')' 7

K AsyncThisDevice

K,* $ * + * F + $ *)
)) !,* + * F ') $ ' * *') $
$ AsyncThisDevice * ! AsyncThisDevice ** *
$ * + * F) +
!

K SyncThisDevice

K,* $ * + * F + + $
* +) ') .) + $
!(' $ * + $$ * F!
+ * $ +) $ * +
* F! SyncThisDevice ** * $ * + * F
!

K ReplayThisDevice

K ,* $ * + + * F
) * + + 1 * ' + *
GrabDeviceButton * ' $ +) AllowDeviceEvents ' SyncThis-
Device ,) * ' Grab), + '$
$! 8 ' + G) $ +
+ 1 4 * 6) ! 8 G)
** * $ * + * * F
) * + !

K AsyncOtherDevices

K ,* ' + * F + $ * '
))) !,* + * F ') $ ' *
*') $ $ AsyncOtherDevices HH II * ! AsyncOtherDe-
vices ** * + * F) +
!

K SyncAll

K ,* + * F + $ 1* + 4)
') .) + $ *
+ ' + $$ * F ! + * $
+) + * F 1) *
+) G) + *) +
* F 4! SyncAll **) + * F !,*
+ * F * * $ SyncAll
- -* 1)) G) * F * SyncAll * F +
4!

K AsyncAll

K ,* + * F + $ 1* + 4)
' !,* + * F ') $ ' * *') $
$ AsyncAll HH II* !,* + * F
* * $ AsyncAll HH II * ! AsyncAll
**) + * F !

AsyncThisDevice , SyncThisDevice , ReplayThisDevice + **
$ * + * ' ' + ! AsyncOtherDevices
** $ * + * ' $ * + !< + L'
SyncAll AsyncAll , + $ ' !

, $ * + * ** + 1 ' ** 4
+ ')) !,* + * F * * +
$ $ * ' * + !, $ * + * F
) * + !, * F ') * *
* + $!

XAllowDeviceEvents BadDevice BadValue !

Controlling Device Focus

8) *) * . $) + ')
XGetDeviceFocus *) ! 9. + *)) XSetDe-
viceFocus *) ' *))

XSetInputFocus*) . $ + ,= $ *) $ ' !
: *) FollowKeyboard , $ + * . + !

8) *) + *) ' * . +
) XGetDeviceFocus .

int **XGetDeviceFocus**(*display, *device, *focus_return,
*revert_to_return, *focus_time_return);

$ * + !

$ * + !

$ * * +
+) ,= *
+ *) * None ,
PointerRoot , FollowKeyboard .

$ * * +
+)) + L) *
+ !

$ * * +
+) *) ' *
+ !

XGetDeviceFocus) *) + *
) ' * . $) + !

XGetDeviceFocus BadDevice BadMatch !

8 *) * . +) XSetDeviceFocus .

int **XSetDeviceFocus**(*display, *device, focus, revert_to, time);

$ * + !

$ * + !

$ * ,= * + I *
)) ! 8 ' ,= Point-
erRoot , FollowKeyboard , None .

$ * *) * +)
+ * *) ' + !:
* * ' $ 7 RevertToPar-
ent , RevertToPointerRoot , RevertToNone , Re-
vertToFollowKeyboard .

$ * '! B) $ ' '$.
$ ' CurrentTime .

XSetDeviceFocus *) * . $) +
*) '!, ** * $ * ' *
) ') + '!: *
) ' $ * '! 8 *)) +
DeviceFocusIn DeviceFocusOut + !

8 + *) G) $ +)
* *))' 7

K ,* *))' None , $) + * ' +
) *) !, + L)' !

K ,* *))' ,= ' *) * + !
,* $) + * ' +) ' $
* * + $ ' !: + $
+ *) !

K ,* *))' PointerRoot , *) '
* + $ $) + !,
+ L)' !

K ,* *))' FollowKeyboard , *) '
' *) * $) + !

8 $ * *) ') + ' XSetDeviceFocus
!: BadMatch !,* *) '
+ + +) + L)' '
*) !

K ,* + L)' RevertToParent , *) + $
1 + 4 + L +)
RevertToNone .

K ,* + L)' RevertToPointerRoot , RevertToFollowKeyboard ,
RevertToNone , *) + +)!

< *) + + DeviceFocusIn DeviceFocusOut
+) *) ' ** !

XSetDeviceFocus BadDevice , BadMatch , BadValue , Bad-
Window !

Controlling Device Feedback

8 ') * * . $) +) XGet-
FeedbackControl .

XFeedbackState * **XGetFeedbackControl**(*display, *device,
*num_feedbacks_return);

$ * + !

$ * + !

))' * *)$$
+ !

XGetFeedbackControl) * FeedbackState))
*)$$ $ * + !8 XFeedbackState))
* ** !8 *+) * ''
" !

```
typedef struct {
     XID class;
     int length;
     XID id;
} XFeedbackState;
```

8 " '' * 7

K 8 '' * * * ! , ' '$
* * < X11/extensions/XI.h >. #) * *
) 7 KbdFeedbackClass , PtrFeedbackClass , StringFeedback-
Class , IntegerFeedbackClass , LedFeedbackClass , BellFeedback-
Class .

K 8 '' $ * * FeedbackState))
) + !

K 8 '') G) * * * + + ! 8
+)$$ ' * * ' !: *
* + ' + ' ,=!

8 * G) +)$$ $
KbdFeedback) XKbdFeedbackState)) *
* 7

```
typedef struct {
     XID class;
     int length;
     XID id;
     int click;
     int percent;
     int pitch;
     int duration;
     int led_mask;
     int global_auto_repeat;
     char auto_repeats[32];
} XKbdFeedbackState;
```

8 '' * XKbdFeedbackState)) $)
* * 7

K 8 '' $ * +)' +) "
1 **4 "" 1) 4!

K 8 $ '' $ * +)' +) "1 **4
"" 1) 4!

K 8 $ '' $ * $ F! 8 * +) '$
' $!

K 8) '' $ *) ' * !

K 8 L' '' ') *)$ J0
D9= ! (+) * $ D9= !

K 8 L) L $ '' +) * AutoRepeatModeOn AutoRepeat-
ModeOff .

K 8) L $ '' + ! 9)
$ * $! 8 + $ J0 !
E A 1* ' "4 * &A &A N *
$ &A!

8 * G) +)$$ $ $
PtrFeedback) XPtrFeedbackState)) *
* 7

```
typedef struct {
     XID class;
     int length;
     XID id;
     int accelNum;
     int accelDenom;
     int threshold;
} XPtrFeedbackState;
```

8 '' * XPtrFeedbackState)) $)
* * 7

K 8 A)'''))' * ') $!

K 8 = ''') ' * ') $!

K 8 = ''') * !

, * $ * $)' $
+ XIntegerFeedbackState)) ! 8 ' ')' '.')'+)
$ $!

```
typedef struct {
     XID class;
     int length;
     XID id;
     int resolution;
     int minVal;
     int maxVal;
} XIntegerFeedbackState;
```

8 '' * XIntegerFeedbackState)) $ $
* * 7

K 8) '' $ *)' * *
$!

K 8 ' '' $ * ' ')' +) * $!

K 8 '. $ * '. ')' +) * $!

* $ * ' $
+ XStringFeedbackState)) ! # * $
* KeySyms $! 8 XGetFeedbackControl *))
* ' * $ '. ')')'
* ' $! 8 XStringFeedbackState)) *
* 7

```
typedef struct {
     XID class;
     int length;
     XID id;
     int max_symbols;
     int  num_syms_supported;
     KeySym *syms_supported;
} XStringFeedbackState;
```

8 '' * XStringFeedbackState)) $ $
* * 7

K 8 '.L ' '' $ * '. ')')' * '
$!

K 8 'L)$$ '' $ *)$$ ' !

K 8)'L 'L)$$ '' $ * * *)$$
' !

E *) $ +
XBellFeedbackState)) ! ' '$ ' ')$$ $
* KbdFeedback * ! # BellFeedback $ + * '$ '
* +)$$ ') $ *
$)) ! 8 ' * '' ' * $
* XKbdFeedbackState)) !

```
typedef struct {
     XID class;
     int length;
     XID id;
     int percent;
     int pitch;
     int duration;
} XBellFeedbackState;
```

D * $ + XLed-
FeedbackState)) ! @$ J0 $ *)$$! 9
L' $)$$ $
L+)) 1 4 ** 1"4! ' '$
' ')$$ $ * KbdFeedback * !# LedFeedback
$ + * '$ ' * +
)$$ ') $ * !

```
typedef struct {
     XID class;
     int length;
     XID id;
     Mask led_values;
     Mask led_mask;
} XLedFeedbackState;
```

XGetFeedbackControl BadDevice BadMatch !

8 * * ') XGetFeedbackControl *))
XFreeFeedbackList .

void **XFreeFeedbackList**(*list);

$ * $ XFeedbackState))
) $ +) XGetFeedbackControl .

XFreeFeedbackList * * * ' !

8 * * . +) XChangeFeedback-
Control . 8 *) ' *) +) * $ * *
) * ' $ $$ $ XFeedbackControl)) *
* !< +) ' * $ +)' $!

int **XChangeFeedbackControl**(*display, *device, valuemask, *value);

$ * + !

$ * + !

$ * +) * ' 1 '
* 4! 8 +) *
* $ * + !

$ * $ XFeedbackControl)) !

XChangeFeedbackControl +
XFeedbackControl)) ! 8 XFeedbackControl)) *
* * ! 8 * +) * ''
" * 7

```
typedef struct {
```

```
     XID class;
     int length;
     XID id;
} XFeedbackControl;
```

; KbdFeedback * G) + $ +
) KbdFeedbackControl)) *
* 7!

```
typedef struct {
     XID class;
     int length;
     XID id;
     int click;
     int percent;
     int pitch;
     int duration;
     int led_mask;
     int led_value;
     int key;
     int auto_repeat_mode;
} XKbdFeedbackControl;
```

8 + XKbdFeedbackCon-
trol)) ! 8) L L$ L) L $
+)) $! ' AutoRepeatModeOn , AutoRepeatModeOff ,
AutoRepeatModeDefault .

' * 7

```
#define   DvKeyClickPercent          (1><<0)
#define   DvPercent                  (1><<0)
#define   DvPitch                    (1><<0)
#define   DvDuration                 (1><<0)
#define   DvLed                      (1><<0)
#define   DvLedMode                  (1><<0)
#define   DvKey                      (1><<0)
#define   DvAutoRepeatMode           (1><<0)
```

; PtrFeedback * G) + $ +
$) PtrFeedbackControl)) * * 7

```
typedef struct {
     XID class;
     int length;
     XID id;
     int accelNum;
```

```
     int accelDenom;
     int threshold;
} XPtrFeedbackControl;
```

< +) ' * $ +)' $!

' * 7

```
#define    DvAccelnum         (1L<<0)
#define    DvAccelDenom       (1L<<1)
#define    DvThreshold        (1L<<2)
```

8 .$ * ') $ * '+' !; .'
$ $ * J5 ' + '+ ' * ' ! 8
* ') + !(** *
+ '+ ' $. $$ ')
+))' ! +)
*) ! 8 +) * A)' * ') F *
$ +) !: $ ') !A +
+) BadValue F +) * = '
* !

8 G) * BadMatch * $ * +)
$ + ' !,* + $ * $ +
) ' ' * ' Relative Absolute XSetDe-
viceMode G) +) +) +
+ ' !

; IntegerFeedback * $ $)
+ $ + IntegerFeedbackControl)) *
* 7

```
typedef struct {
     XID class;
     int length;
     XID id;
     int int_to_display;
} XIntegerFeedbackControl;
```

' * 7

```
#define   DvInteger          (1L<<0)
```

; StringFeedback * $ $)
+ $ + StringFeedbackControl)) *
* 7

```
typedef struct {
     XID class;
     int length;
     XID id;
     int num_keysyms;
     KeySym *syms_to_display;
} XStringFeedbackControl;
```

```
  '     *   7
```

```
#define   DvString            (1L<<0)
```

```
;             BellFeedback           $)  +      $   +
    BellFeedbackControl  ) )       *    *   7
```

```
typedef struct {
     XID class;
     int length;
     XID id;
     int percent;
     int pitch;
     int duration;
} XBellFeedbackControl;
```

```
  '     *   7
```

```
#define   DvPercent                    (1L<<1)
#define   DvPitch                      (1L<<2)
#define   DvDuration                   (1L<<3)
```

```
;             LedFeedback           $)  +      $   +
    LedFeedbackControl  ) )       *    *   7
```

```
typedef struct {
     XID class;
     int length;
     XID id;
     int led_mask;
     int led_values;
} XLedFeedbackControl;
```

```
  '     *   7
```

```
#define   DvLed                        (1L<<4)
```

```
#define   DvLedMode                    (1L<<5)

XChangeFeedbackControl            BadDevice , BadFeedBack , BadMatch ,
    BadValue   !
```

Ringing a Bell on an Input Device

```
8         .    $)  +  )                    XDeviceBell .

int  XDeviceBell( *display,  *device,  feedbackid,  percent);

                          $ *              + !

                          $ *        + !

                          $ *    *     !    + )                    Kbd-
                          FeedbackClass    BellFeedbackClass .

                          $ *    ,= *   *            !

                          $ *    + )'        "" 1G)  4   ""
                          $    1 ) 4!

XDeviceBell      )                  XBell *)   ! ,       $ *
    $ *   $)  +  *    )     $ *  + )' ! 8  $ *
+ )'     +       + )' *   *   ! ,*  + ) *   $
 )'            ""   ""   ) +                        BadValue     ) ! 8
+ )'               $     )'       +  7


       base - [(base * percent) / 100] + percent

8  + )'               $     )'      +  7


       base + [(base * percent) / 100]

8        + )' *     )                  XChangeFeedbackControl .

XDeviceBell            BadDevice    BadValue   !
```

Controlling Device Encoding

```
8      ' $$  *  .    +    )$$   $)                                Keys , )
XGetDeviceKeyMapping .

KeySym    *        XGetDeviceKeyMapping(    *display,        *device,
first_keycode_wanted,  keycode_count,  *keysyms_per_keycode_return);

                          $ *              + !

                          $ *        + !

                          $ *    *             )  !

                          $ *    )'  *
                          )  !
```

))' * ' $!

XGetDeviceKeyMapping) XGetKeyboardMapping*) !,
) ' * $ *)' * * $ * .
+ !

XGetDeviceKeyMapping) ' * $ *)' *
* $ * . + $ * ! 8
* L L ') G) ')
XListInputDevices G) 1 BadValue) 4! 8 *
+) 7

```
first_keycode_wanted + keycode_count - 1
```

') G) '.) XListInputDevices
G) 1 BadValue) 4!

8)' * ' ' * 7

```
keycode_count * keysyms_per_keycode_return
```

(C9B B)' A 1) * 'F 4 * C . 1)
* 'F 4 ' * * 7

```
(K - first_keycode_wanted) * keysyms_per_keycode_return + N
```

8 'L$ L L) +) +
) $ G) ' !($ C9B B +) * NoSymbol
) *)) ' * +) !

8 *) *)) XFree!

,* $ * + * $ + XOpenDevice ,
G) * BadDevice !,* +)$$ $)
Keys , G) * BadMatch !

XGetDeviceKeyMapping BadDevice , BadMatch , BadValue
!

8 '$$ * . +)$$ $)
Keys ,) XChangeDeviceKeyMapping .

int **XChangeDeviceKeyMapping**(*display, *device, *first_keycode, *keysyms_per_keycode, *keysyms, num_codes);

$ * + !

$ * + !

$ * * !

$ * ') !

$ * $ * '!

$ *)' *
!

XChangeDeviceKeyMapping) XChangeKeyboardMapping
*) !, * ' * $ *)' * * $
* . + !

,* $ * + * $ + XOpenDevice ,
G) * BadDevice !,* $ * +)$$
$) Keys , G) * BadMatch !

8)' * ' ' ') ') $ *
'L$ L ! : XChangeDeviceKeyMapping
BadLength !8 $ * * L ') G)
' L +)) ListInputDevices G) G)
* BadValue !, * * .$
'.L +)) ListInputDevices G) G)
* BadValue 7

first_keycode + (num_codes / keysyms_per_keycode) - 1

XChangeDeviceKeyMapping BadAlloc , BadDevice , BadMatch ,
BadValue !

8) ' * . +)$
$ $) Keys ,) XGetDeviceModifierMapping .

XModifierKeymap * **XGetDeviceModifierMapping**(*display, *device);

$ * + !

$ * + !

XGetDeviceModifierMapping) XGetModifierMapping*)
!8 XGetDeviceModifierMapping*)) XModi-
fierKeymap))) ' * * $
* + !8))) * *) XFreeModifierMapping .
,* F +) $$ * ' * ' * !

XGetDeviceModifierMapping BadDevice BadMatch !

8) ' * * . +)
XSetDeviceModifierMapping .

int **XSetDeviceModifierMapping**(*display, *device, *modmap);

$ * + !

$ * + !

$ * $ XModifierKeymap)) !

XSetDeviceModifierMapping) XSetModifierMapping*)
!8 XSetDeviceModifierMapping*) $ * *
*) ' * !(F +) ')
) !A)' + ' F +)!:

XSetDeviceModifierMapping BadValue ! 8 '
* ' * '$'' * XModifierKeymap))
*'.L $ ' * ' * Shift ,
Lock , Control , Mod1 , Mod2 , Mod3 , Mod4 , Mod5 .: F
+ ' F ! , *
F ') $ * ' L '.L
$ XListInputDevices *) !: XSetModifierMapping
BadValue ! A ' $$ '$!:
BadValue !

(+ '$ ' * * .'$
*)$ *') $ ' *
)$$!,* ') +) $ Map-
pingFailed , * ' * !,* $ *
* ' * ** * ') * 1) 4 *
' *) $ MappingBusy ,
* ' * ! XSetModifierMapping DeviceMap-
pingNotify + MappingSuccess)!

XSetDeviceModifierMapping BadAlloc , BadDevice , BadMatch ,
BadValue !

Controlling Button Mapping

8 '$$ *) . +) XSetDeviceButton-
Mapping .

int **XSetDeviceButtonMapping**(*display, *device, map[], nmap);

$ * + !

$ * + !

$ * '$$!

$ *)' * ' '$$!

XSetDeviceButtonMapping '$$ *) . + !
,*) + DeviceMappingNotify + XSet-
DeviceButtonMapping) MappingSuccess . 9 ' * .
* ' !8 * ') ' XGetDeviceButton-
Mapping)) BadValue) !8 .))'
' * * ** +)' !(F '
) ' +))' *$) !
+ ' + ' F +) BadValue
) !,* *) XSetDe-
viceButtonMapping) MappingBusy , '$$!

XSetDeviceButtonMapping BadDevice , BadMatch , BadValue
!

8) '$$) XGetDeviceButtonMapping .

int **XGetDeviceButtonMapping**(*display, *device, map_return[], nmap);

$ * + !

$ * + !

$ * '$$!

$ *)' * ' '$$!

XGetDeviceButtonMapping)) '$$ * $ * .
+ !9 ' * . * ' ! XGetDeviceButtonMap-
ping))' * $)) $! 8 '
'$$ *) '$$ 7 '$O PQ ! 8 '$)'
$ * *) '$$)
* '$ ') '$L) !

XGetDeviceButtonMapping BadDevice BadMatch !

Obtaining the State of a Device

8 * ' *) +) *
. +) XQueryDeviceState .

```
XDeviceState *  XQueryDeviceState( *display,  *device);
```

$ * + !

$ * + !

XQueryDeviceState) $ XDeviceState)) $
*)) *) +)
+ 7

```
typedef struct {
     XID device_id;
     int num_classes;
     XInputClass *data;
} XDeviceState;
```

8)) *+) * '' "
* 7

```
typedef struct {
     unsigned char class;
     unsigned char length;
} XInputClass;
```

8 '' * !8 * '$
* * < X11/extensions/XI.h >. #) *
7 KeyClass , ButtonClass , ValuatorClass .

8 '' *)))
+ !

8 XValuatorState))) * +)
+ !8)'L+) '')' *+) + !
8 ' '' ' $ ' *
' * + !8 *) * 7

DeviceMode 1 << 0 Relative = 0, Absolute = 1
ProximityState 1 << 1 InProximity = 0, OutOfProximity = 1

8 +) '' $ *
) +) * +) !,* ' Relative , +)) * !

```
typedef struct {
    unsigned char class;
    unsigned char length;
    unsigned char num_valuators;
    unsigned char mode;
    int *valuators;
} XValuatorState;
```

8 XKeyState))) * + !E
A 1* ' "4 * &A &A N *
$ &A!

```
typedef struct {
    unsigned char class;
    unsigned char length;
    short num_keys;
    char keys[32];
} XKeyState;
```

8 XButtonState))) *)
+ !E A 1* ' "4 *) &A &A N
* $) &A!

```
typedef struct {
    unsigned char class;
    unsigned char length;
    short num_buttons;
    char buttons[32];
} XButtonState;
```

XQueryDeviceState BadDevice !

8 *) *)) XFreeDeviceState .

```
void  XFreeDeviceState( *state);
```

$ * $ XDeviceState)
$ +) XQueryDeviceState .

XFreeDeviceState * + !

Events

8 $) . $) +) $) + !8
. $) + ' $) * . $)
+ !8 ' *) * , $) 9.
+ $ 7

K 9+ $

K 9+

K 9+))

Event Types

9+ $)') ' *
+ + !8 '$ $ * * +))
+ $ ' ' !

8 $) + $ * * < X11/
X.h >. 9. + $!, '
. I G) + . F !E
) * . + $ ') * ' + !

8 $ ' ' + $ * . +)
* ') XOpenDevice G) !8 $) *
'$ $ * * + + !

9. + $ $)$ ' '
+ !,* . +)) $ $
) L$ $ $!
* . + ' * + $)
. + !

8 * + + $ $!

= +	DeviceKeyPress
V	DeviceKeyRelease
= + '	DeviceButtonPress
V	DeviceButtonRelease
V	DeviceMotionNotify
= + $) *)	DeviceFocusIn
V	DeviceFocusOut
= + *	DeviceStateNotify

=+ $.'	ProximityIn
V	ProximityOut
=+ '$$	DeviceMappingNotify
=+	ChangeDeviceNotify

Event Classes

9+)') ' +
' !8) $ ' + +
$ ' +!

8 $) + ' * * < X11/
X.h >. 9. + !, '
. I G) + . F !
E) * . + ') * '
+!

8 + * . + + * ' * '
) XOpenDevice *) !8) XSelectEx-
tensionEvent G) + * $ * ' +
$ '!

; DeviceButtonPress + ' $ * '$
$ +)) $!,*
) + DeviceButtonRelease + * DeviceBut-
tonPress + +) $ * DeviceButtonPressGrab
DeviceButtonPress !8
'' G) DeviceButtonPress + * ' ' +
* $ * !

,* $ * DeviceButtonPressGrab G)
$ * ' + $ * DeviceBut-
tonPress DeviceButtonPressGrab) Access !

,* DeviceButtonPress $ * '$ $ +
) $ + !) $ ')
$ * ' + ' !

8 ' DeviceMotion +) !
, $ * + DeviceButton1Motion)
DeviceButton5Motion . ($) +)$$ ') '
) !

Event Structures

9 . + $ $)) < X11/exten-
sions/XInput.h >. (+)) + * " '' 7

$ + $)') G) * !
; .'$ + $ DeviceKey-
Press + $$ XDe-
viceKeyPressEvent)) !

* ')' $ $)
.$ * ' R * *)
J0 +) !

L + True * + ' * ' XSendEvent
G) !

$ $)) * $
+ !

9. +)) $) $ * $!,
* + $ ' $)) +)
* +) + $!

Device Key Events

C + * ' . + * '
+ * ' !, + ,= $
) +) * +) + * + $)
!,* * ' .+) $ ' +
! 8 . L) '')' * .
$! 8 + ' * + $
+ !9 +)' * . $. L)
'' * . $) + * L . '' !,*
+)$$ $) Valuators ,) $) '
. L) '' F 1"4!

8 $ . .L L '' *
$!

8 XDeviceKeyEvent)) * * 7

```
typedef struct {
     int type;                      /* of event */
     unsigned long serial;          /* # of last request processed */
     Bool send_event;               /* true if from SendEvent request */
     Display *display;              /* Display the event was read from */
     Window window;                 /* "event" window reported relative to */
     XID deviceid;
     Window root;                   /* root window event occurred on */
     Window subwindow;              /* child window */
     Time time;                     /* milliseconds */
     int x, y;                      /* x, y coordinates in event window */
     int x_root;                    /* coordinates relative to root */
     int y_root;                    /* coordinates relative to root */
     unsigned int state;            /* key or button mask */
     unsigned int keycode;          /* detail */
     Bool same_screen;              /* same screen flag */
     unsigned int device_state;     /* device key or button mask */
     unsigned char axes_count;
     unsigned char first_axis;
     int axis_data[6];
```

```
} XDeviceKeyEvent;

typedef XDeviceKeyEvent XDeviceKeyPressedEvent;
typedef XDeviceKeyEvent XDeviceKeyReleasedEvent;
```

Device Button Events

E) + * ' . + * '
) + * ' $!, + ,= $
) +) * +) + * + $)
! ,* * ' . +) $ ')
+ ' ! 8 . L) '')' * .
$!8 + ' * + $
+ !9 +)' * . $. L)
'' * . $) + * L . '' !,*
+)$$ $) Valuators ,) $) '
. L) '' F 1"4!

8 $ . .L L '' *
$!

```
typedef struct {
     int type;                      /* of event */
     unsigned long serial;          /* # of last request processed by server */
     Bool send_event;               /* true if from a SendEvent request */
     Display *display;              /* Display the event was read from */
     Window window;                 /* "event" window reported relative to */
     XID deviceid;
     Window root;                   /* root window that the event occurred on */
     Window subwindow;              /* child window */
     Time time;                     /* milliseconds */
     int x, y;                      /* x, y coordinates in event window */
     int x_root;                    /* coordinates relative to root */
     int y_root;                    /* coordinates relative to root */
     unsigned int state;            /* key or button mask */
     unsigned int button;           /* detail */
     Bool same_screen;              /* same screen flag */
     unsigned int device_state;     /* device key or button mask */
     unsigned char axes_count;
     unsigned char first_axis;
     int axis_data[6];
} XDeviceButtonEvent;

typedef XDeviceButtonEvent XDeviceButtonPressedEvent;
typedef XDeviceButtonEvent XDeviceButtonReleasedEvent;
```

Device Motion Events

+ * ' . + * '
' + * ' $!, + ,= $
) +) * +) + !

8 $ . .L L '' *
$ 0 ' !

9. ' + ' $ ' * +)' * . !
8 . L) '')' * . $!8
+ ' * + $ + !9
+)' * . $. L) ''
* . $) + * L . '' !

```
typedef struct {
     int type;                          /* of event */
     unsigned long serial;              /* # of last request processed by server *
     Bool send_event;                   /* true if from a SendEvent request */
     Display *display;                  /* Display the event was read from */
     Window window;                     /* "event" window reported relative to */
     XID deviceid;
     Window root;                       /* root window that the event occurred on
     Window subwindow;                  /* child window */
     Time time;                         /* milliseconds */
     int x, y;                          /* x, y coordinates in event window */
     int x_root;                        /* coordinates relative to root */
     int y_root;                        /* coordinates relative to root */
     unsigned int state;                /* key or button mask */
     char is_hint;                      /* detail */
     Bool same_screen;                  /* same screen flag */
     unsigned int device_state;         /* device key or button mask */
     unsigned char axes_count;
     unsigned char first_axis;
     int axis_data[6];
} XDeviceMotionEvent;
```

Device Focus Events

8 + G) + *) + !8 ' *
' * + ,= * + *)
' '$!

DeviceFocusIn DeviceFocusOut + * *) * .
+ '' *) + !

```
typedef struct {
     int type;                      /* of event */
     unsigned long serial;          /* # of last request processed by server */
     Bool send_event;               /* true if this came from a SendEvent request
     Display *display;              /* Display the event was read from */
     Window window;                 /* "event" window it is reported relative to *
     XID deviceid;
     int mode;                      /* NotifyNormal, NotifyGrab, NotifyUngrab */
     int detail;
```

```
                                        /*
                                         * NotifyAncestor, NotifyVirtual, NotifyInferi
                                         * NotifyNonLinear,NotifyNonLinearVirtual, Not
                                         * NotifyPointerRoot, NotifyDetailNone
                                         */
     Time time;
} XDeviceFocusChangeEvent;

typedef XDeviceFocusChangeEvent XDeviceFocusInEvent;
typedef XDeviceFocusChangeEvent XDeviceFocusOutEvent;
```

Device StateNotify Event

8 +) '$ +) $) *
+ * $))$$!, * + Device-
FocusIn + EnterNotify + + +
XDeviceStateNotify + !

,* +)$$ $) Valuators , ' '' XValua-
torStatus)) ' $ + ' $.'
* ' !8 *) * 7

```
     0x01     Relative = 0, Absolute = 1
     0x02     InProximity = 0, OutOfProximity = 1
```

,* +)$$ ' +) $ XEvent ,
') $ XDeviceStateNotify + !

```
typedef struct {
     unsigned char class;
     unsigned char length;
} XInputClass;

typedef struct {
     int type;
     unsigned long serial;     /* # of last request processed by server */
     Bool send_event;          /* true if this came from a SendEvent request */
     Display *display;         /* Display the event was read from */
     Window window;
     XID deviceid;
     Time time;
     int num_classes;
     char data[64];
} XDeviceStateNotifyEvent;

typedef struct {
     unsigned char class;
     unsigned char length;
     unsigned char num_valuators;
```

```
     unsigned char mode;
     int valuators[6];
} XValuatorStatus;

typedef struct {
     unsigned char class;
     unsigned char length;
     short num_keys;
     char keys[32];
} XKeyStatus;

typedef struct {
     unsigned char class;
     unsigned char length;
     short num_buttons;
     char buttons[32];
} XButtonStatus;
```

Device Mapping Event

8 + G) + MappingNotify + !, * $
' '$$ * ' *) . +
!

```
typedef struct {
     int type;
     unsigned long serial;
     Bool send_event;
     Display *display;
     Window window;
     XID deviceid;
     Time time;
     int request;
     int first_keycode;
     int count;
} XDeviceMappingEvent;
```

ChangeDeviceNotify Event

8 + G) + $!, * $ '
* + !

```
typedef struct {
     int type;
     unsigned long serial;
     Bool send_event;
     Display *display;
     Window window;
     XID deviceid;
```

```
    Time time;
    int request;
} XChangeDeviceNotifyEvent;
```

Proximity Events

8 + + G) + $! ' $) +)
$) ' +)
'+) * $) * !

8 +) +) * +) + * +
$) ! ,* * ' . +) $
' $.' + ' ! 8 . L) ''
)' * . $! 8 + ' * +
$ + ! 9 +)' *
. $. L) '' * . $)
+ * L . '' !,* +)$$ $) Valuators ,)
$) ' . L) '' F 1"4!

```
typedef struct {
    int type;                /* ProximityIn or ProximityOut */
    unsigned long serial;    /* # of last request processed by server */
    Bool send_event;         /* true if this came from a SendEvent request */
    Display *display;        /* Display the event was read from */
    Window window;
    XID deviceid;
    Window root;
    Window subwindow;
    Time time;
    int x, y;
    int x_root, y_root;
    unsigned int state;
    Bool same_screen;
    unsigned int device_state;   /* device key or button mask */
    unsigned char axes_count;
    unsigned char first_axis;
    int axis_data[6];
} XProximityNotifyEvent;

typedef XProximityNotifyEvent XProximityInEvent;
typedef XProximityNotifyEvent XProximityOutEvent;
```

Event Handling Functions

8) , $) 9. + *)
) 7

K = ' . +

K D + +

K 9 . +

K # ' * +

K , F +) $) +

K $) +

K # $) +

K . + +

K = ' + +

K # + $ $

K + +

K '

Determining the Extension Version

```
XExtensionVersion *  XGetExtensionVersion( *display,  *name);
```

$ * + !

$ * ' * . !

XGetExtensionVersion ' +)$$
+ * $) . !

8 XExtensionVersion))) * ') + * .
)$$ + * * 7

```
typedef struct {
     Bool present;
     short major_version;
     short minor_version;
} XExtensionVersion;
```

8 '6 ' + '$ *
* < X11/extensions/XI.h >. 9 +)$ * $ +) + !

B))) XFree! *) *) !

Listing Available Devices

($ ' $ * + ') * '
+ + ! 8) XListIn-
putDevices *)) * + $
+ ! 8 $ ') * ' * < X11/
extensions/XI.h > * XInternAtom G) ' +
$ * + ! 8 $ '$ + $
) XListInputDevices G) !

```
XDeviceInfo *  XListInputDevices( *display,  *ndevices);
```

$ * + !

$ * * + +
))' * $) + +
+ !

XListInputDevices ' + + *
$) * ') + !(* XDeviceInfo))
) ' * + ! 8)' * +
) +)' !

8 $ + + $ +
. $) + ! 8) '' * XDeviceInfo)) $ *
)) * + !,* +) * '' IsXPointer , +
$ + !,* +) IsXKeyboard , +
+ !,* +) IsXExtensionDevice , + + *)
. $) + !

9 XDeviceInfo $ *))
* * $))$$ + ! 8)'L
'')' * !

,* +)$$ $) Valuators , *)) $
XDeviceInfo)) XValuatorInfo)) ! 8 . ''
*)) * * XAxisInfo)) !8
' * . *' $ + ! 8)'
* ')'L . ' * XValua-
torInfo)) ! 8 F * ')** * + $
' L)** '' * XValuatorInfo)) !

8 XDeviceInfo)) * * 7

```
typedef struct _XDeviceInfo {
     XID id;
     Atom type;
     char *name;
     int num_classes;
     int use;
     XAnyClassPtr inputclassinfo;
} XDeviceInfo;
```

8)) $ XDeviceInfo)) * * 7

```
typedef struct _XKeyInfo {
     XID class;
     int length;
     unsigned short min_keycode;
     unsigned short max_keycode;
     unsigned short num_keys;
```

```
} XKeyInfo;

typedef struct _XButtonInfo {
     XID class;
     int length;
     short num_buttons;
} XButtonInfo;

typedef struct _XValuatorInfo {
     XID class;
     int length;
     unsigned char num_axes;
     unsigned char mode;
     unsigned long motion_buffer;
     XAxisInfoPtr axes;
} XValuatorInfo;
```

8 XAxisInfo)) $ XValuatorInfo)) *
* 7

```
typedef struct _XAxisInfo {
     int resolution;
     int min_value;
     int max_value;
} XAxisInfo;
```

8 * ' ' * < X11/extensions/XI.h > * !

```
MOUSE     QUADRATURE
TABLET     SPACEBALL
KEYBOARD     DATAGLOVE
TOUCHSCREEN     EYETRACKER
TOUCHPAD     CURSORKEYS
BUTTONBOX     FOOTMOUSE
BARCODE     ID_MODULE
KNOB_BOX     ONE_KNOB
TRACKBALL     NINE_KNOB\s+1
```

8 ') XInternAtom G)) '
) * '$ $ '' * XDeviceInfo)) !

XListInputDevices) A@DD * $) + !

8 *) XListInputDevices ,) XFreeDeviceList .

void **XFreeDeviceList**(**list*);

$ * $ XDeviceInfo)
$ +) XListInputDevices .

XFreeDeviceList * * $) + * ' !

Enabling and Disabling Extension Devices

9 $ ' . + ') G)
+ $ + XOpenDevice *) !

```
XDevice *  XOpenDevice( *display,  device_id);
```

$ * + !

$ * ,=) G) * +
$! 8 ,= * ' XListInputDevices G) !

XOpenDevice $ + * G)))
XDevice)) * * 7

```
typedef struct {
     XID device_id;
     int num_classes;
     XInputClassInfo *classes;
} XDevice;
```

8 XDevice)) $ * XInputClassInfo)
) !9 ' * ') + * $)
$))$$ $) + !

8 XInputClassInfo)) * * 7

```
typedef struct {
     unsigned char input_class;
     unsigned char event_type_base;
} XInputClassInfo;
```

($ ' ' + $ + * + +
) ' * $) . !8 ' * ' $
+ ' $)) +
* ' $) * .'$ 7

```
     DeviceKeyPress(XDevice *device, event_type, event_class)
```

8 ' * +) * +) XSelectExtensionEvent G) + + $) '$
+ $ * + + + XNextEvent!

XOpenDevice BadDevice !

E * ' $ ') G) +
+ XCloseDevice *) !

```
int XCloseDevice( *display, *device);
```

$ * + !

$ * + !

XCloseDevice + * G) *
XDevice)) !

(' $ ' . + ' ! G) *
*) *)) XDevice)) ' * '
*) + ** !(XCloseDevice G)
' I + !

+ + $ + G)
!,* + * F + * G)
G)) + !

,* $ ' ') + +) '
+ * * !8 **
' + !

XCloseDevice BadDevice !

Changing the Mode of a Device

' + $ * $ +) ' !8
' * + * ' +)) XSetDeviceMode .

```
int XSetDeviceMode( *display, *device, mode);
```

$ * + !

$ * + ') !

$ * ' !B) $ Absolute Relative .

XSetDeviceMode G) + ' *
+ $ * $) $ + '
!,* + + * $ +) G)
+ $ + + XOpenDevice G) G) *
BadDevice !,* +)$$ $) Valuators *
$ * $ $ * ' G) * BadMatch !

8 G) *) DeviceBusy * $
+ G) ** ' !

XSetDeviceMode BadDevice , BadMatch , BadMode , DeviceBusy
!

Initializing Valuators on an Input Device

' + $) $ F
+) ! = + $ * $ + ') $

' G) +) F +) * '
* + Absolute .

8 F +)) +) XSetDeviceValuators .

```
Status  XSetDeviceValuators( *display,  *device,  num_valuators);
```

$ * + !

$ * + +))
F !

$ * +) +))
!

$ * * +) !

$ *)' *+) !

XSetDeviceValuators F $ * +) $ * .
$) + !))' F ! : +)
$ * * L+))'L+) ! (BadValue
) *)' *+))$$ + *
.$ 7

first_valuator + num_valuators

,* G)) Success) !,* $ * +
' G) *) * AlreadyGrabbed
) !

XSetDeviceValuators BadDevice , BadLength , BadMatch , BadValue !

Getting Input Device Controls

' $) +)$$ +) *) G)
!8 *)$$ + * ' $) +
! G) ' $) * * +
$) +)$$ G) + !

9 +) G) * !,* ' $ +
+ '$$))) G) +
!

8 G) +) XGetDeviceControl .

```
XDeviceControl *  XGetDeviceControl( *display,  *device,  control);
```

$ * + !

$ * + *))
) !

, * $ * + G) !

XGetDeviceControl)) * $ * + !,*
+)$$ + BadValue) !
,* $ * +)$$ + BadMatch
) !

,* G)) *) $ XDeviceState))
) ! 8 * ') + $ *
'$$)) $$ $ * ! 8 * '' '
' + * * 7

```
typedef struct {
     XID control;
     int length;
} XDeviceState;
\fP
```

8 ' '$ * * < X11/exten-
sions/XI.h >. #) * +) =9 ,#9L 9 :D@8,:A!

8 * ') * =9 ,#9L 9 :D@8,:A *
XDeviceResolutionState)) * * 7

```
typedef struct {
     XID control;
     int length;
     int num_valuators;
     int *resolutions;
     int *min_resolutions;
     int *max_resolutions;
} XDeviceResolutionState;
```

8 +) *+) *+)
* !))' F 1"4!) *
+) +) !; +) +) O P
)) *) ' L) O P) '
')'+ '.L) O P) '.')'+ !

< $ * XGetDeviceControl * BadMatch *
$ * + +) !

XGetDeviceControl BadMatch BadValue !

Changing Input Device Controls

' $) +)$$ +) *)
! 8 $) F +
*) ! 8 *)$$ + * ' $) +

! G) ' $) * * +
$) +)$$ G) + !
+ * * $) +
$ * !

9 +) G) * ! , * ' $ +
+ '$$))) G) +
!

8 +) XChangeDeviceControl .

```
Status XChangeDeviceControl( *display, *device, control, *value);
```

$ * + !

$ * + *))
' * !

, * $ * + !

$ * $ XDeviceControl))

!

XChangeDeviceControl) * $ * + !
,* +)$$ + BadValue
) ! ,* $ * +)$$ + BadMatch
) ! ,* +) * Al-
readyGrabbed) ! ,* + $
*) * DeviceBusy) ! ,* G) * *
+ !

,* G)) *) +) * Suc-
cess) ! 8 * ' $ + $ *
'$$)) $$ $ * ! 8 * ''
" + 7

```
typedef struct {
    XID control;
    int length;
} XDeviceControl;
```

8 ') * < X11/extensions/XI.h >
* ! #) * +) =9 ,#9L 9 :D@8,:A!

8 * ' =9 ,#9L 9 :D@8,:A
* XDeviceResolutionControl)) * * 7

```
typedef struct {
    XID control;
```

```
    int length;
    int first_valuator;
    int num_valuators;
    int *resolutions;
} XDeviceResolutionControl;
```

8 +) * $ * +) $
* . $) + !))' F !:
+) $ * * L+))'L+) !(+)
*)) * +)
! 8)'L+) '' $ *)' * +)
) !

< $ * XChangeDeviceControl * BadMatch
* $ * + +) !,*) $ *
*+ +) 1) XGetDeviceControl), XChangeDevice-
Control * BadValue !(BadValue) *)' * +
))$$ + * .$ 7

```
    first_valuator + num_valuators,
```

XChangeDeviceControl BadMatch BadValue !

Selecting Extension Device Events

8 + $) +) XSelectExtensionEvent . 8 $ '
$ $ * * + $
+) *)' * ' ,= * * '
+ !

int **XSelectExtensionEvent**(**display*, *window*, **event_list*, *event_count*);

$ * + !

$ * ,= * * '
+ + !

$ * $ * +
$ * + !

$ *)' * ' + L !

XSelectExtensionEvent G) + + ' +
+ + ' * ' G) !
8 ' * XEventClass + L +)
+ ' $ XDevice))) XOpenDe-
vice G) !; .'$ DeviceKeyPress ')) XEvent-
Class * DeviceKeyPress + * ' $ * + * +
* * '7

DeviceKeyPress (XDevice *device, event_type, event_class)

* * * + 7

DeviceKeyPress
DeviceKeyRelease
DeviceButtonPress
DeviceButtonRelease
DeviceMotionNotify
DeviceFocusIn
DeviceFocusOut
ProximityIn
ProximityOut
DeviceStateNotify
DeviceMappingNotify
ChangeDeviceNotify
DevicePointerMotionHint
DeviceButton1Motion
DeviceButton2Motion
DeviceButton3Motion,
DeviceButton4Motion
DeviceButton5Motion
DeviceButtonMotion,
DeviceOwnerGrabButton
DeviceButtonPressGrab

8 . + + * ' $ ') XNex-
tEvent *) ! 8) . + ' * ' +
. + !

) XSelectExtensionEvent G)) + * '
+ $ * $ +) G) $ $ +) *
+ * ' + !

XSelectExtensionEvent BadAccess , BadClass , BadLength ,
BadWindow !

Determining Selected Device Events

8 ' . +) * ' +
) XGetSelectedExtensionEvents .

int **XGetSelectedExtensionEvents**(*display, window,
*this_client_count, **this_client, *all_clients_count,
**all_clients);

$ * + !

$ * ,= * * '
+ + !

))' * ' L !

) * XEventClasses $ *
+ !

))' * ' L !

) * XEventClasses $ *
+ !

XGetSelectedExtensionEvents) $ + !:
. + * ' $ * !8
. + * ' $ * !8
* ')))L+ L' L+ L'
* XWindowAttributes)) XGetWindowAttributes G)
' !8 *) *)) XFree!

XGetSelectedExtensionEvents BadWindow !

Controlling Event Propagation

9. + $ $)$ ' '
+ !,* . +)) $ $
) L$ $ $!
* . + ' * + $)
. + !

$ ' ' + $ $)) * *
*) 7 XChangeDeviceDontPropagateList XGetDeviceDontPropagateList .

int **XChangeDeviceDontPropagateList**(**display*, *window*, *event_count*, **events*, *mode*);

$ * + !

$ * !

$ *)' * ' + !

$ * $ * 9+ # !

$ * ' ! B) $ AddToList
DeleteFromList .

XChangeDeviceDontPropagateList + + * '
L L$ $ * . + * $ * !8
$ ' * * * !8 *
' !

)$$ * + $ $ * + !,* $ * XEvent-
Class +))$$ * + BadClass
) !

XChangeDeviceDontPropagateList BadClass , BadMode , Bad-
Window !

XEventClass * **XGetDeviceDontPropagateList**(**display*, *window*, **event_count*);

$ * + !

$ * !

))' * ')
*) !

XGetDeviceDontPropagateList ' L L$ $
* . + * $ * !,) * XEvent-
Class , XEventClass $ + 5+ $ $!8 *
) *)) XFree!

XGetDeviceDontPropagateList BadWindow !

Sending an Event

8 . +) XSendExtensionEvent .

int **XSendExtensionEvent**(*display, *device, window, propagate, event_count, *event_list, *event);

$ * + !

$ * + ,= + !

$ * ,=! B) $
,= PointerWindow InputFocus .

$ * +) True False .

$ *)' * ' + L
!

$ * $ * XEventClass .

$ * $ + !

XSendExtensionEvent * '
) + $ * + + !, G)
* XEventClass $ * !8 $ $) +
XOpenDevice G) !

XSendExtensionEvent))' *
* 7

K,*)$ PointerWindow ,
$!

K,*)$ InputFocus * *) $
$!,* *)
$ *) !

8 ') + $ * + XSendExtension-
Event) $ $)' * 7

K,*$ $ False , + + * '
* + $ * + L !

K ,* $ $ True + * '
* + $ * + L $
* * '
* $ * + * + +
L L$ $ ' !,*) . *
* *) InputFocus $ *
+ !: + $ +
* * + $ * + L !

8 + XEvent)) ') * + * $)
. + $!
8 * +)) +
. $ * L + True * + G)
)' + !

XSendExtensionEvent) F * + $ * M
) F !

XSendExtensionEvent BadClass , BadDevice , BadValue , BadWindow !

Getting Motion History

XDeviceTimeCoord * **XGetDeviceMotionEvents**(*axis_count_return),* *display, *device, stop, *nevents_return, *mode_return, *axis_count_return);

$ * + !

$ * + !

$ * '!

$ * $ '!

))' *$ ')**
) * G) !

) ' * + * ' ! 8
' * * 7 Absolute Relative .

))' * . $ *
$) !

XGetDeviceMotionEvents) $ + I ')**
* $ * $ ') +!,* '
*)) $ ' $) !

8) $ * *) XDeviceTimeCoord)) *
* 7

typedef struct {

```
    Time time;
    unsigned int *data;
} XDeviceTimeCoord;
```

8 '' $ * '! 9 ' * $
'$. *' $ + ! 8)'
* . $ +) . L) + !

8 +) * ' $ ' * + ! 8 '
) ' + !,* ' Absolute , '
+) + ! 8 ' $ ')
'.')' +) + * . *'
$!8 '.')'+) * . $ '.L+ '' *
XAxisInfo)) $ * * ') XListInputDevices G) !

,* ' Relative , ' + +)
+ !8 $ '') $ * + '
) $)') + +) !

) + XGetDeviceMotionEvents) * ** '
* ' $ ' ' * + + XSetDeviceMode G) !

XGetDeviceMotionEvents BadDevice BadMatch !

8 *) XGetDeviceMotionEvents ,) XFreeDeviceMotionEvents .

void **XFreeDeviceMotionEvents**(**events*);

$ * $ XDeviceTimeCoord
) $ +) XGetDeviceMotionEvents .

XFreeDeviceMotionEvents * $ * *' * ' ! Appendix A

8 * * ' <X11/extensions/XInput.h>
<X11/extensions/XI.h> * 7

```
/* Definitions used by the library and client */

#ifndef _XINPUT_H_
#define _XINPUT_H_

#ifndef _XLIB_H_
#include <X11/Xlib.h>
#endif

#ifndef _XI_H_
#include "XI.h"
#endif
```

```
#define _deviceKeyPress          0
#define _deviceKeyRelease     1

#define _deviceButtonPress     0
#define _deviceButtonRelease      1

#define _deviceMotionNotify     0

#define _deviceFocusIn          0
#define _deviceFocusOut          1

#define _proximityIn          0
#define _proximityOut          1

#define _deviceStateNotify     0
#define _deviceMappingNotify      1
#define _changeDeviceNotify     2

#define FindTypeAndClass(d, type, class, classid, offset) \
    { int i; XInputClassInfo *ip; \
    type = 0; class = 0; \
    for (i=0, ip= ((XDevice *) d)->classes; \
      i< ((XDevice *) d)->num_classes; \
      i++, ip++) \
     if (ip->input_class == classid) \
         {type =  ip->event_type_base + offset; \
          class =  ((XDevice *) d)->device_id << 8 | type;}}

#define DeviceKeyPress(d, type, class) \
    FindTypeAndClass(d, type, class, KeyClass, _deviceKeyPress)

#define DeviceKeyRelease(d, type, class) \
    FindTypeAndClass(d, type, class, KeyClass, _deviceKeyRelease)

#define DeviceButtonPress(d, type, class) \
    FindTypeAndClass(d, type, class, ButtonClass, _deviceButtonPress)

#define DeviceButtonRelease(d, type, class) \
    FindTypeAndClass(d, type, class, ButtonClass, _deviceButtonRelease)

#define DeviceMotionNotify(d, type, class) \
    FindTypeAndClass(d, type, class, ValuatorClass, _deviceMotionNotify)

#define DeviceFocusIn(d, type, class) \
    FindTypeAndClass(d, type, class, FocusClass, _deviceFocusIn)

#define DeviceFocusOut(d, type, class) \
    FindTypeAndClass(d, type, class, FocusClass, _deviceFocusOut)

#define ProximityIn(d, type, class) \
    FindTypeAndClass(d, type, class, ProximityClass, _proximityIn)

#define ProximityOut(d, type, class) \
```

```
    FindTypeAndClass(d, type, class, ProximityClass, _proximityOut)

#define DeviceStateNotify(d, type, class) \
    FindTypeAndClass(d, type, class, OtherClass, _deviceStateNotify)

#define DeviceMappingNotify(d, type, class) \
    FindTypeAndClass(d, type, class, OtherClass, _deviceMappingNotify)

#define ChangeDeviceNotify(d, type, class) \
    FindTypeAndClass(d, type, class, OtherClass, _changeDeviceNotify)

#define DevicePointerMotionHint(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _devicePointerMotionHint;}

#define DeviceButton1Motion(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButton1Motion;}

#define DeviceButton2Motion(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButton2Motion;}

#define DeviceButton3Motion(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButton3Motion;}

#define DeviceButton4Motion(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButton4Motion;}

#define DeviceButton5Motion(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButton5Motion;}

#define DeviceButtonMotion(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButtonMotion;}

#define DeviceOwnerGrabButton(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceOwnerGrabButton;}

#define DeviceButtonPressGrab(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _deviceButtonGrab;}

#define NoExtensionEvent(d, type, class) \
    { class =  ((XDevice *) d)->device_id << 8 | _noExtensionEvent;}

#define BadDevice(dpy, error) _xibaddevice(dpy, &error)

#define BadClass(dpy, error) _xibadclass(dpy, &error)

#define BadEvent(dpy, error) _xibadevent(dpy, &error)

#define BadMode(dpy, error) _xibadmode(dpy, &error)

#define DeviceBusy(dpy, error) _xidevicebusy(dpy, &error)

/*******************************************************************
 *
 * DeviceKey events.  These events are sent by input devices that
```

```
 * support input class Keys.
 * The location of the X pointer is reported in the coordinate
 * fields of the x,y and x_root,y_root fields.
 *
 */

typedef struct
    {
    int             type;         /* of event */
    unsigned long   serial;       /* # of last request processed */
    Bool            send_event;   /* true if from SendEvent request */
    Display         *display;     /* Display the event was read from */
    Window          window;       /* "event" window reported relative to */
    XID             deviceid;
    Window          root;         /* root window event occured on */
    Window          subwindow;    /* child window */
    Time            time;         /* milliseconds */
    int             x, y;         /* x, y coordinates in event window */
    int             x_root;       /* coordinates relative to root */
    int             y_root;       /* coordinates relative to root */
    unsigned int    state;        /* key or button mask */
    unsigned int    keycode;      /* detail */
    Bool            same_screen;  /* same screen flag */
    unsigned int    device_state; /* device key or button mask */
    unsigned char   axes_count;
    unsigned char   first_axis;
    int             axis_data[6];
    } XDeviceKeyEvent;

typedef XDeviceKeyEvent XDeviceKeyPressedEvent;
typedef XDeviceKeyEvent XDeviceKeyReleasedEvent;

/*******************************************************************
 *
 * DeviceButton events.  These events are sent by extension devices
 * that support input class Buttons.
 *
 */

typedef struct {
    int           type;         /* of event */
    unsigned long serial;       /* # of last request processed by server */
    Bool          send_event;   /* true if from a SendEvent request */
    Display       *display;     /* Display the event was read from */
    Window        window;       /* "event" window reported relative to */
    XID           deviceid;
    Window        root;         /* root window that the event occured on */
    Window        subwindow;    /* child window */
    Time          time;         /* milliseconds */
    int           x, y;         /* x, y coordinates in event window */
    int           x_root;       /* coordinates relative to root */
    int           y_root;       /* coordinates relative to root */
    unsigned int  state;        /* key or button mask */
    unsigned int  button;       /* detail */
```

```
    Bool          same_screen;  /* same screen flag */
    unsigned int  device_state; /* device key or button mask */
    unsigned char axes_count;
    unsigned char first_axis;
    int           axis_data[6];
    } XDeviceButtonEvent;

typedef XDeviceButtonEvent XDeviceButtonPressedEvent;
typedef XDeviceButtonEvent XDeviceButtonReleasedEvent;

/*****************************************************************
 *
 * DeviceMotionNotify event.  These events are sent by extension devices
 * that support input class Valuators.
 *
 */

typedef struct
    {
    int           type;        /* of event */
    unsigned long serial;      /* # of last request processed by server */
    Bool          send_event;  /* true if from a SendEvent request */
    Display       *display;    /* Display the event was read from */
    Window        window;      /* "event" window reported relative to */
    XID           deviceid;
    Window        root;        /* root window that the event occured on */
    Window        subwindow;   /* child window */
    Time          time;        /* milliseconds */
    int           x, y;        /* x, y coordinates in event window */
    int           x_root;      /* coordinates relative to root */
    int           y_root;      /* coordinates relative to root */
    unsigned int  state;       /* key or button mask */
    char          is_hint;     /* detail */
    Bool          same_screen; /* same screen flag */
    unsigned int  device_state; /* device key or button mask */
    unsigned char axes_count;
    unsigned char first_axis;
    int           axis_data[6];
    } XDeviceMotionEvent;

/*****************************************************************
 *
 * DeviceFocusChange events.  These events are sent when the focus
 * of an extension device that can be focused is changed.
 *
 */

typedef struct
    {
    int           type;       /* of event */
    unsigned long serial;     /* # of last request processed by server */
    Bool          send_event; /* true if from a SendEvent request */
    Display       *display;   /* Display the event was read from */
    Window        window;     /* "event" window reported relative to */
```

```
    XID           deviceid;
    int           mode;       /* NotifyNormal, NotifyGrab, NotifyUngrab */
    int           detail;
     /*
      * NotifyAncestor, NotifyVirtual, NotifyInferior,
      * NotifyNonLinear,NotifyNonLinearVirtual, NotifyPointer,
      * NotifyPointerRoot, NotifyDetailNone
      */
    Time                time;
    } XDeviceFocusChangeEvent;

typedef XDeviceFocusChangeEvent XDeviceFocusInEvent;
typedef XDeviceFocusChangeEvent XDeviceFocusOutEvent;

/*******************************************************************
 *
 * ProximityNotify events.  These events are sent by those absolute
 * positioning devices that are capable of generating proximity information.
 *
 */

typedef struct
    {
    int             type;      /* ProximityIn or ProximityOut */
    unsigned long   serial;    /* # of last request processed by server */
    Bool            send_event; /* true if this came from a SendEvent request */
    Display         *display;  /* Display the event was read from */
    Window          window;
    XID                 deviceid;
    Window          root;
    Window          subwindow;
    Time            time;
    int             x, y;
    int             x_root, y_root;
    unsigned int    state;
    Bool            same_screen;
    unsigned int    device_state; /* device key or button mask */
    unsigned char   axes_count;
    unsigned char   first_axis;
    int             axis_data[6];
    } XProximityNotifyEvent;
typedef XProximityNotifyEvent XProximityInEvent;
typedef XProximityNotifyEvent XProximityOutEvent;

/*******************************************************************
 *
 * DeviceStateNotify events are generated on EnterWindow and FocusIn
 * for those clients who have selected DeviceState.
 *
 */

typedef struct
    {
    unsigned char     class;
```

```
    unsigned char     length;
    } XInputClass;

typedef struct {
    int           type;
    unsigned long serial;       /* # of last request processed by server */
    Bool          send_event;   /* true if this came from a SendEvent request */
    Display       *display;     /* Display the event was read from */
    Window        window;
    XID           deviceid;
    Time          time;
    int           num_classes;
    char       data[64];
} XDeviceStateNotifyEvent;

typedef struct {
    unsigned char     class;
    unsigned char     length;
    unsigned char     num_valuators;
    unsigned char     mode;
    int                  valuators[6];
} XValuatorStatus;

typedef struct {
    unsigned char     class;
    unsigned char     length;
    short          num_keys;
    char             keys[32];
} XKeyStatus;

typedef struct {
    unsigned char     class;
    unsigned char     length;
    short          num_buttons;
    char             buttons[32];
} XButtonStatus;

/*******************************************************************
 *
 * DeviceMappingNotify event.  This event is sent when the key mapping,
 * modifier mapping, or button mapping of an extension device is changed.
 *
 */

typedef struct {
    int           type;
    unsigned long serial;       /* # of last request processed by server */
    Bool          send_event;   /* true if this came from a SendEvent request */
    Display       *display;     /* Display the event was read from */
    Window        window;       /* unused */
    XID           deviceid;
    Time          time;
    int           request;      /* one of MappingModifier, MappingKeyboard,
                                    MappingPointer */
```

```
    int           first_keycode;/* first keycode */
    int           count;        /* defines range of change w. first_keycode*/
} XDeviceMappingEvent;

/*******************************************************************
 *
 * ChangeDeviceNotify event.  This event is sent when an
 * XChangeKeyboard or XChangePointer request is made.
 *
 */

typedef struct {
    int           type;
    unsigned long serial;       /* # of last request processed by server */
    Bool          send_event;   /* true if this came from a SendEvent request */
    Display       *display;     /* Display the event was read from */
    Window        window;       /* unused */
    XID           deviceid;
    Time          time;
    int           request;      /* NewPointer or NewKeyboard */
} XChangeDeviceNotifyEvent;

/*******************************************************************
 *
 * Control structures for input devices that support input class
 * Feedback.  These are used by the XGetFeedbackControl and
 * XChangeFeedbackControl functions.
 *
 */

typedef struct {
     XID            class;
     int            length;
     XID            id;
} XFeedbackState;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     click;
    int     percent;
    int     pitch;
    int     duration;
    int     led_mask;
    int     global_auto_repeat;
    char    auto_repeats[32];
} XKbdFeedbackState;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     accelNum;
```

```
    int     accelDenom;
    int     threshold;
} XPtrFeedbackState;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     resolution;
    int     minVal;
    int     maxVal;
} XIntegerFeedbackState;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     max_symbols;
    int     num_syms_supported;
    KeySym  *syms_supported;
} XStringFeedbackState;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     percent;
    int     pitch;
    int     duration;
} XBellFeedbackState;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     led_values;
    int     led_mask;
} XLedFeedbackState;

typedef struct {
     XID            class;
     int            length;
     XID         id;
} XFeedbackControl;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     accelNum;
    int     accelDenom;
    int     threshold;
} XPtrFeedbackControl;
```

```
typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     click;
    int     percent;
    int     pitch;
    int     duration;
    int     led_mask;
    int     led_value;
    int     key;
    int     auto_repeat_mode;
} XKbdFeedbackControl;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     num_keysyms;
    KeySym  *syms_to_display;
} XStringFeedbackControl;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     int_to_display;
} XIntegerFeedbackControl;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     percent;
    int     pitch;
    int     duration;
} XBellFeedbackControl;

typedef struct {
    XID     class;
    int     length;
    XID     id;
    int     led_mask;
    int     led_values;
} XLedFeedbackControl;

/*****************************************************************
 *
 * Device control structures.
 *
 */

typedef struct {
     XID            control;
```

```
     int            length;
} XDeviceControl;

typedef struct {
     XID            control;
     int            length;
     int            first_valuator;
     int            num_valuators;
     int            *resolutions;
} XDeviceResolutionControl;

typedef struct {
     XID            control;
     int            length;
     int            num_valuators;
     int            *resolutions;
     int            *min_resolutions;
     int            *max_resolutions;
} XDeviceResolutionState;

/*******************************************************************
 *
 * An array of XDeviceList structures is returned by the
 * XListInputDevices function.  Each entry contains information
 * about one input device.  Among that information is an array of
 * pointers to structures that describe the characteristics of
 * the input device.
 *
 */

typedef struct _XAnyClassinfo *XAnyClassPtr;

typedef struct _XAnyClassinfo {
    XID      class;
    int      length;
    } XAnyClassInfo;

typedef struct _XDeviceInfo *XDeviceInfoPtr;

typedef struct _XDeviceInfo
    {
    XID                 id;
    Atom                type;
    char                *name;
    int                 num_classes;
    int                 use;
    XAnyClassPtr      inputclassinfo;
    } XDeviceInfo;

typedef struct _XKeyInfo *XKeyInfoPtr;

typedef struct _XKeyInfo
    {
    XID               class;
```

```
    int               length;
    unsigned short      min_keycode;
    unsigned short      max_keycode;
    unsigned short      num_keys;
    } XKeyInfo;

typedef struct _XButtonInfo *XButtonInfoPtr;

typedef struct _XButtonInfo {
    XID          class;
    int          length;
    short      num_buttons;
    } XButtonInfo;

typedef struct _XAxisInfo *XAxisInfoPtr;

typedef struct _XAxisInfo {
    int      resolution;
    int      min_value;
    int      max_value;
    } XAxisInfo;

typedef struct _XValuatorInfo *XValuatorInfoPtr;

typedef struct     _XValuatorInfo
    {
    XID               class;
    int               length;
    unsigned char       num_axes;
    unsigned char       mode;
    unsigned long       motion_buffer;
    XAxisInfoPtr        axes;
    } XValuatorInfo;


/*******************************************************************
 *
 * An XDevice structure is returned by the XOpenDevice function.
 * It contains an array of pointers to XInputClassInfo structures.
 * Each contains information about a class of input supported by the
 * device, including a pointer to an array of data for each type of event
 * the device reports.
 *
 */


typedef struct {
        unsigned char   input_class;
        unsigned char   event_type_base;
} XInputClassInfo;

typedef struct {
        XID                     device_id;
        int                     num_classes;
```

```
        XInputClassInfo        *classes;
} XDevice;


/*******************************************************************
 *
 * The following structure is used to return information for the
 * XGetSelectedExtensionEvents function.
 *
 */

typedef struct {
        XEventClass     event_type;
        XID             device;
} XEventList;

/*******************************************************************
 *
 * The following structure is used to return motion history data from
 * an input device that supports the input class Valuators.
 * This information is returned by the XGetDeviceMotionEvents function.
 *
 */

typedef struct {
        Time   time;
        int    *data;
} XDeviceTimeCoord;


/*******************************************************************
 *
 * Device state structure.
 * This is returned by the XQueryDeviceState request.
 *
 */

typedef struct {
        XID          device_id;
        int          num_classes;
        XInputClass     *data;
} XDeviceState;

/*******************************************************************
 *
 * Note that the mode field is a bitfield that reports the Proximity
 * status of the device as well as the mode.  The mode field should
 * be OR'd with the mask DeviceMode and compared with the values
 * Absolute and Relative to determine the mode, and should be OR'd
 * with the mask ProximityState and compared with the values InProximity
 * and OutOfProximity to determine the proximity state.
 *
 */
```

```
typedef struct {
    unsigned char     class;
    unsigned char     length;
    unsigned char     num_valuators;
    unsigned char     mode;
    int                  *valuators;
} XValuatorState;

typedef struct {
    unsigned char     class;
    unsigned char     length;
    short          num_keys;
    char             keys[32];
} XKeyState;

typedef struct {
    unsigned char     class;
    unsigned char     length;
    short          num_buttons;
    char             buttons[32];
} XButtonState;

/*******************************************************************
 *
 * Function definitions.
 *
 */

_XFUNCPROTOBEGIN

extern int     XChangeKeyboardDevice(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */
#endif
);

extern int     XChangePointerDevice(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int               /* xaxis */,
    int               /* yaxis */
#endif
);

extern int     XGrabDevice(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    Window          /* grab_window */,
    Bool          /* ownerEvents */,
    int               /* event count */,
    XEventClass*     /* event_list */,
```

```
    int               /* this_device_mode */,
    int               /* other_devices_mode */,
    Time          /* time */
#endif
);

extern int     XUngrabDevice(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    Time           /* time */
#endif
);

extern int     XGrabDeviceKey(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned int     /* key */,
    unsigned int     /* modifiers */,
    XDevice*          /* modifier_device */,
    Window          /* grab_window */,
    Bool          /* owner_events */,
    unsigned int     /* event_count */,
    XEventClass*     /* event_list */,
    int               /* this_device_mode */,
    int               /* other_devices_mode */
#endif
);

extern int     XUngrabDeviceKey(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned int     /* key */,
    unsigned int     /* modifiers */,
    XDevice*          /* modifier_dev */,
    Window          /* grab_window */
#endif
);

extern int     XGrabDeviceButton(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned int     /* button */,
    unsigned int     /* modifiers */,
    XDevice*          /* modifier_device */,
    Window          /* grab_window */,
    Bool          /* owner_events */,
    unsigned int     /* event_count */,
    XEventClass*     /* event_list */,
    int               /* this_device_mode */,
    int               /* other_devices_mode */
```

```
#endif
);

extern int     XUngrabDeviceButton(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned int     /* button */,
    unsigned int     /* modifiers */,
    XDevice*          /* modifier_dev */,
    Window          /* grab_window */
#endif
);

extern int     XAllowDeviceEvents(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int               /* event_mode */,
    Time          /* time */
#endif
);

extern int     XGetDeviceFocus(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    Window*          /* focus */,
    int*          /* revert_to */,
    Time*          /* time */
#endif
);

extern int     XSetDeviceFocus(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    Window          /* focus */,
    int               /* revert_to */,
    Time          /* time */
#endif
);

extern XFeedbackState     *XGetFeedbackControl(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int*          /* num_feedbacks */
#endif
);

extern int     XFreeFeedbackList(
#if NeedFunctionPrototypes
    XFeedbackState*     /* list */
```

```
#endif
);

extern int     XChangeFeedbackControl(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned long     /* mask */,
    XFeedbackControl*     /* f */
#endif
);

extern int     XDeviceBell(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    XID               /* feedbackclass */,
    XID               /* feedbackid */,
    int               /* percent */
#endif
);

extern KeySym     *XGetDeviceKeyMapping(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
#if NeedWidePrototypes
    unsigned int     /* first */,
#else
    KeyCode          /* first */,
#endif
    int               /* keycount */,
    int*          /* syms_per_code */
#endif
);

extern int     XChangeDeviceKeyMapping(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int               /* first */,
    int               /* syms_per_code */,
    KeySym*          /* keysyms */,
    int               /* count */
#endif
);

extern XModifierKeymap     *XGetDeviceModifierMapping(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */
#endif
);
```

```
extern int     XSetDeviceModifierMapping(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    XModifierKeymap*     /* modmap */
#endif
);

extern int     XSetDeviceButtonMapping(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned char*     /* map[] */,
    int               /* nmap */
#endif
);

extern int     XGetDeviceButtonMapping(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    unsigned char*     /* map[] */,
    unsigned int     /* nmap */
#endif
);

extern XDeviceState     *XQueryDeviceState(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */
#endif
);

extern int     XFreeDeviceState(
#if NeedFunctionPrototypes
    XDeviceState*     /* list */
#endif
);

extern XExtensionVersion *XGetExtensionVersion(
#if NeedFunctionPrototypes
    Display*          /* display */,
    _Xconst char*     /* name */
#endif
);

extern XDeviceInfo     *XListInputDevices(
#if NeedFunctionPrototypes
    Display*          /* display */,
    int*          /* ndevices */
#endif
);

extern int     XFreeDeviceList(
```

```
#if NeedFunctionPrototypes
    XDeviceInfo*     /* list */
#endif
);

extern XDevice     *XOpenDevice(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XID               /* id */
#endif
);

extern int     XCloseDevice(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */
#endif
);

extern int     XSetDeviceMode(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int               /* mode */
#endif
);

extern int     XSetDeviceValuators(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int*          /* valuators */,
    int               /* first_valuator */,
    int               /* num_valuators */
#endif
);

extern XDeviceControl     *XGetDeviceControl(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int               /* control */
#endif
);

extern int     XChangeDeviceControl(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    int               /* control */,
    XDeviceControl*     /* d */
#endif
);
```

```
extern int     XSelectExtensionEvent(
#if NeedFunctionPrototypes
    Display*          /* display */,
    Window          /* w */,
    XEventClass*     /* event_list */,
    int               /* count */
#endif
);

extern int     XGetSelectedExtensionEvents(
#if NeedFunctionPrototypes
    Display*          /* display */,
    Window          /* w */,
    int*          /* this_client_count */,
    XEventClass**     /* this_client_list */,
    int*          /* all_clients_count */,
    XEventClass**     /* all_clients_list */
#endif
);

extern int     XChangeDeviceDontPropagateList(
#if NeedFunctionPrototypes
    Display*          /* display */,
    Window          /* window */,
    int               /* count */,
    XEventClass*     /* events */,
    int               /* mode */
#endif
);

extern XEventClass     *XGetDeviceDontPropagateList(
#if NeedFunctionPrototypes
    Display*          /* display */,
    Window          /* window */,
    int*          /* count */
#endif
);

extern Status     XSendExtensionEvent(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
    Window          /* dest */,
    Bool          /* prop */,
    int               /* count */,
    XEventClass*     /* list */,
    XEvent*          /* event */
#endif
);

extern XDeviceTimeCoord     *XGetDeviceMotionEvents(
#if NeedFunctionPrototypes
    Display*          /* display */,
    XDevice*          /* device */,
```

```
    Time          /* start */,
    Time          /* stop */,
    int*          /* nEvents */,
    int*          /* mode */,
    int*          /* axis_count */
#endif
);

extern int     XFreeDeviceMotionEvents(
#if NeedFunctionPrototypes
    XDeviceTimeCoord*     /* events */
#endif
);

extern int     XFreeDeviceControl(
#if NeedFunctionPrototypes
    XDeviceControl*     /* control */
#endif
);

_XFUNCPROTOEND

#endif /* _XINPUT_H_ */

/* Definitions used by the server, library and client */

#ifndef _XI_H_

#define _XI_H_

#define sz_xGetExtensionVersionReq          8
#define sz_xGetExtensionVersionReply          32
#define sz_xListInputDevicesReq               4
#define sz_xListInputDevicesReply           32
#define sz_xOpenDeviceReq                 8
#define sz_xOpenDeviceReply                 32
#define sz_xCloseDeviceReq                 8
#define sz_xSetDeviceModeReq                8
#define sz_xSetDeviceModeReply                32
#define sz_xSelectExtensionEventReq          12
#define sz_xGetSelectedExtensionEventsReq     8
#define sz_xGetSelectedExtensionEventsReply     32
#define sz_xChangeDeviceDontPropagateListReq     12
#define sz_xGetDeviceDontPropagateListReq     8
#define sz_xGetDeviceDontPropagateListReply     32
#define sz_xGetDeviceMotionEventsReq          16
#define sz_xGetDeviceMotionEventsReply          32
#define sz_xChangeKeyboardDeviceReq          8
#define sz_xChangeKeyboardDeviceReply          32
#define sz_xChangePointerDeviceReq          8
#define sz_xChangePointerDeviceReply          32
#define sz_xGrabDeviceReq                20
#define sz_xGrabDeviceReply                 32
#define sz_xUngrabDeviceReq                 12
```

```
#define sz_xGrabDeviceKeyReq               20
#define sz_xGrabDeviceKeyReply               32
#define sz_xUngrabDeviceKeyReq               16
#define sz_xGrabDeviceButtonReq               20
#define sz_xGrabDeviceButtonReply          32
#define sz_xUngrabDeviceButtonReq          16
#define sz_xAllowDeviceEventsReq          12
#define sz_xGetDeviceFocusReq               8
#define sz_xGetDeviceFocusReply               32
#define sz_xSetDeviceFocusReq              16
#define sz_xGetFeedbackControlReq          8
#define sz_xGetFeedbackControlReply          32
#define sz_xChangeFeedbackControlReq          12
#define sz_xGetDeviceKeyMappingReq          8
#define sz_xGetDeviceKeyMappingReply          32
#define sz_xChangeDeviceKeyMappingReq          8
#define sz_xGetDeviceModifierMappingReq          8
#define sz_xSetDeviceModifierMappingReq          8
#define sz_xSetDeviceModifierMappingReply     32
#define sz_xGetDeviceButtonMappingReq          8
#define sz_xGetDeviceButtonMappingReply          32
#define sz_xSetDeviceButtonMappingReq          8
#define sz_xSetDeviceButtonMappingReply          32
#define sz_xQueryDeviceStateReq               8
#define sz_xQueryDeviceStateReply          32
#define sz_xSendExtensionEventReq          16
#define sz_xDeviceBellReq               8
#define sz_xSetDeviceValuatorsReq          8
#define sz_xSetDeviceValuatorsReply          32
#define sz_xGetDeviceControlReq               8
#define sz_xGetDeviceControlReply          32
#define sz_xChangeDeviceControlReq          8
#define sz_xChangeDeviceControlReply          32

#define INAME                "XInputExtension"

#define XI_KEYBOARD     "KEYBOARD"
#define XI_MOUSE     "MOUSE"
#define XI_TABLET     "TABLET"
#define XI_TOUCHSCREEN     "TOUCHSCREEN"
#define XI_TOUCHPAD     "TOUCHPAD"
#define XI_BARCODE     "BARCODE"
#define XI_BUTTONBOX     "BUTTONBOX"
#define XI_KNOB_BOX     "KNOB_BOX"
#define XI_ONE_KNOB     "ONE_KNOB"
#define XI_NINE_KNOB     "NINE_KNOB"
#define XI_TRACKBALL     "TRACKBALL"
#define XI_QUADRATURE     "QUADRATURE"
#define XI_ID_MODULE     "ID_MODULE"
#define XI_SPACEBALL     "SPACEBALL"
#define XI_DATAGLOVE     "DATAGLOVE"
#define XI_EYETRACKER     "EYETRACKER"
#define XI_CURSORKEYS     "CURSORKEYS"
#define XI_FOOTMOUSE     "FOOTMOUSE"
```

```
#define Dont_Check               0
#define XInput_Initial_Release          1
#define XInput_Add_XDeviceBell          2
#define XInput_Add_XSetDeviceValuators     3
#define XInput_Add_XChangeDeviceControl     4

#define XI_Absent          0
#define XI_Present          1

#define XI_Initial_Release_Major          1
#define XI_Initial_Release_Minor          0

#define XI_Add_XDeviceBell_Major          1
#define XI_Add_XDeviceBell_Minor          1

#define XI_Add_XSetDeviceValuators_Major     1
#define XI_Add_XSetDeviceValuators_Minor     2

#define XI_Add_XChangeDeviceControl_Major     1
#define XI_Add_XChangeDeviceControl_Minor     3

#define DEVICE_RESOLUTION     1

#define NoSuchExtension          1

#define COUNT               0
#define CREATE               1

#define NewPointer          0
#define NewKeyboard          1

#define XPOINTER          0
#define XKEYBOARD          1

#define UseXKeyboard          0xFF

#define IsXPointer          0
#define IsXKeyboard          1
#define IsXExtensionDevice     2

#define AsyncThisDevice          0
#define SyncThisDevice          1
#define ReplayThisDevice     2
#define AsyncOtherDevices     3
#define AsyncAll          4
#define SyncAll               5

#define FollowKeyboard           3
#define RevertToFollowKeyboard      3

#define DvAccelNum              (1L << 0)
#define DvAccelDenom            (1L << 1)
#define DvThreshold             (1L << 2)
```

```
#define DvKeyClickPercent     (1L<<0)
#define DvPercent           (1L<<1)
#define DvPitch               (1L<<2)
#define DvDuration           (1L<<3)
#define DvLed                (1L<<4)
#define DvLedMode           (1L<<5)
#define DvKey                (1L<<6)
#define DvAutoRepeatMode     (1L<<7)

#define DvString                (1L << 0)

#define DvInteger               (1L << 0)

#define DeviceMode              (1L << 0)
#define Relative                0
#define Absolute                1

#define ProximityState          (1L << 1)
#define InProximity             (0L << 1)
#define OutOfProximity          (1L << 1)

#define AddToList               0
#define DeleteFromList          1

#define KeyClass            0
#define ButtonClass            1
#define ValuatorClass            2
#define FeedbackClass            3
#define ProximityClass       4
#define FocusClass            5
#define OtherClass            6

#define KbdFeedbackClass       0
#define PtrFeedbackClass       1
#define StringFeedbackClass       2
#define IntegerFeedbackClass       3
#define LedFeedbackClass       4
#define BellFeedbackClass       5

#define _devicePointerMotionHint 0
#define _deviceButton1Motion      1
#define _deviceButton2Motion      2
#define _deviceButton3Motion      3
#define _deviceButton4Motion      4
#define _deviceButton5Motion      5
#define _deviceButtonMotion      6
#define _deviceButtonGrab      7
#define _deviceOwnerGrabButton      8
#define _noExtensionEvent      9

#define XI_BadDevice     0
#define XI_BadEvent     1
#define XI_BadMode     2
```

```
#define XI_DeviceBusy     3
#define XI_BadClass     4

typedef     unsigned long     XEventClass;

/*******************************************************************
 *
 * Extension version structure.
 *
 */

typedef struct {
        int         present;
        short     major_version;
        short     minor_version;
} XExtensionVersion;

#endif /* _XI_H_ */
```

Appendix A. Input Extension Protocol Encoding

Syntactic Conventions

( )' ') $ * . U. .
' 1 R4!

8 .) G) $ + '$)
$ 7

```
NameofThing
  encode-form
  ...
  encode-form
```

9 * ' '$!

; '$ $ 7

name: TYPE

* ' 7

N TYPE name

A)' *)$ ' 8B 9 $
* !; .'$

depth: CARD8

' 7

1 CARD8 depth

; '$)' +) * ' 7

N value name

8 +) $ A) ! ; .'$
* * < F 1 4
1 < $) 47

```
1 0 Error
1 3 code
```

; '$ $ 7

```
name: {Name1, ..., NameI}
```

* ' 7

```
N  name
 value1 Name1
 ...
 valueI NameI
```

8 +) $ A) ! A F *
A ' ' G) +) !; .'$ 7

```
class: {InputOutput, InputOnly, CopyFromParent}
```

' 7

```
2   class
 0 CopyFromParent
 1 InputOutput
 2 InputOnly
```

; '$ $ 7

```
NAME: TYPE or Alternative1 ... or AlternativeI
```

* ' 7

```
N TYPE   NAME
 value1 Alternative1
 ...
 valueI AlternativeI
```

8 + +)) * * 8B 9!
; .'$ 7

destination: WINDOW or PointerWindow or InputFocus

' 7

4 WINDOW destination
0 PointerWindow
1 InputFocus

; '$ $ 7

value-mask: BITMASK

* ' 7

N BITMASK value-mask
mask1 mask-name1
...
maskI mask-nameI

8 +) ' $ * ' A 0 S! 8 '
* E,8 (C + *) * 1') 4
' .)' . G) ! 8 $ $
* *) $ ' '
A * ' * + '
$ * ' * ' * A)
A) $ ' + '
$$ +) A) !

; D, 8 * (D@9 G) * * * '7

VALUEs
encode-form
...
encode-form

* '* (D@9! 8 A(9 * '
$ E,8 (C ! 8 * (D@9)$ *)
))' * $ * * ' ' *
*)) M '))
+) ' !

, +))' *)$ '$ $ *
+ ' * $ *)' +) *

' '$ + +) $ * '$)' .$
+ + + ! # '$ $ *) .$
$) ! 8 $ *) + 6)
G) $ + '$) $)) ! ; . '$ 7

2 3+n request length
4n LISTofPOINT points

;)) 1 +) *) * ' 4
* ' 7

N unused

,*)' *)) + * ' $ 7

p unused, p=pad(E)

9 ' .$ $ 194)' *) 9
)$ ') $ * *) !

pad(E) = (4 - (E mod 4)) mod 4

Common Types

D, 8 *;::

K ,)' D, 8 * ' ')' * $
* ;:: M) * !

98 *;::

K ($ ' $
!

E,8 (C7 #(=J0

<,A=:<7 #(=J0

EB897 & +)

,A8&7 &

,A8 R7 R

,A8J07 J0

#(=&7 &)

#(= R7 R)

#(=J07 J0)

8, 9 8(7 #(=J0

9 9A8#D(7 #(=J0

```
INPUTCLASS
 0 KeyClass
 1 ButtonClass
 2 ValuatorClass
 3 FeedbackClass
 4 ProximityClass
 5 FocusClass
 6 OtherClass
```

```
INPUTCLASS
 0 KbdFeedbackClass
 1 PtrFeedbackClass
 2 StringFeedbackClass
 3 IntegerFeedbackClass
 4 LedFeedbackClass
 5 BellFeedbackClass
```

```
INPUTINFO
 0 KEYINFO
 1 BUTTONINFO
 2 VALUATORINFO
```

```
DEVICEMODE
 0 Relative
 1 Absolute
```

```
PROXIMITYSTATE
 0 InProximity
 1 OutOfProximity
```

```
BOOL
 0 False
 1 True
```

C9B B 7 #(=J0

C9B#:=97 #(=&

E@88:A7 #(=&

```
SETofKEYBUTMASK
 #x0001 Shift
 #x0002 Lock
 #x0004 Control
 #x0008 Mod1
 #x0010 Mod2
 #x0020 Mod3
 #x0040 Mod4
 #x0080 Mod5
 #x0100 Button1
 #x0200 Button2
 #x0400 Button3
 #x0800 Button4
 #x1000 Button5
 #xe000 unused but must be zero
```

```
SETofKEYMASK
 encodings are the same as for SETofKEYBUTMASK, except with
 #xff00  unused but must be zero
```

8 ,A &7 D, 8 *#(=&

```
STR
 1 n  length of name in bytes
 n STRING8  name
```

```
Errors
```

```
Request
 1 0  Error
 1 1  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
```

```
 1 CARD8  major opcode
 21   unused



Value
 1 0  Error
 1 2  code
 2 CARD16  sequence number
 4 <32-bits>  bad value
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Window
 1 0  Error
 1 3  code
 2 CARD16  sequence number
 4 CARD32  bad resource id
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Match
 1 0  Error
 1 8  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Access
 1 0  Error
 1 10  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Alloc
```

```
 1 0  Error
 1 11  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Name
 1 0  Error
 1 15  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Device
 1 0  Error
 1 CARD8  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Event
 1 0  Error
 1 CARD8  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused



Mode
 1 0  Error
 1 CARD8  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused
```

```
Class
 1 0  Error
 1 CARD8  code
 2 CARD16  sequence number
 4   unused
 2 CARD16  minor opcode
 1 CARD8  major opcode
 21   unused
```

C

C9B#:=9 +) 1 0TR4!

C9B B +) U. """"""" + + $ * !

8 ' * C9B B +) $$.
;!

E@88:A +))' !

G)

```
GetExtensionVersion
 1 CARD8  input extension opcode
 1 1  GetExtensionVersion opcode
 2 2+(n+p)/4  request length
 2 n  length of name
 2   unused
 n STRING8  name
 p   unused, p=pad(n)
```

```
 =>
 1 1  Reply
 1 1  GetExtensionVersion opcode
 2 CARD16  sequence number
 4 0  reply length
 2 CARD16  major version
 2 CARD16  minor version
 1 BOOL  present
 19   unused
```

```
ListInputDevices
```

```
1 CARD8  input extension opcode
1 2  ListInputDevices opcode
2 1  request length



=>
1 1   Reply
1 2   ListInputDevices opcode
2 CARD16   sequence number
4 (n+p)/4   reply length
1 CARD8   number of input devices
23    unused
n LISTofDEVICEINFO info for each input device
p    unused, p=pad(n)



 DEVICEINFO
4 CARD32  device type
1 CARD8  device id
1 CARD8  number of input classes this device reports
1 CARD8  device use
  0 IsXPointer
  1 IsXKeyboard
  2 IsXExtensionDevice
1   unused
n LISTofINPUTINFO   input info for each input class
m STR   name
p    unused, p=pad(m)



 INPUTINFO KEYINFO or BUTTONINFO or VALUATORINFO



 KEYINFO
1 0  class id
1 8  length
1 KEYCODE  minimum keycode
1 KEYCODE  maximum keycode
2 CARD16  number of keys
2   unused



BUTTONINFO
 1 1  class id
 1 4  length
```

```
 2 CARD16  number of buttons



VALUATORINFO
 1 2  class id
 1 8+12n  length
 1 n  number of axes
 1 SETofDEVICEMODE   mode
 4 CARD32  size of motion buffer
 12n LISTofAXISINFO  valuator limits



AXISINFO
 4 CARD32  resolution
 4 CARD32  minimum value
 4 CARD32  maximum value



OpenDevice
 1 CARD8  input extension opcode
 1 3  OpenDevice opcode
 2 2  request length
 1 CARD8  device id
 3   unused



 =>
 1 1   Reply
 1 3   OpenDevice opcode
 2 CARD16   sequence number
 4 (n+p)/4   reply length
 1 CARD8   number of input classes
 23    unused
 n LISTofINPUTCLASSINFO input class information
 p    unused, p=pad(n)



  INPUTCLASSINFO
 1 CARD8  input class id
  0 KEY
  1 BUTTON
  2 VALUATOR
  3 FEEDBACK
  4 PROXIMITY
  5 FOCUS
```

```
  6 OTHER
 1 CARD8  event type base code for this class



CloseDevice
 1 CARD8  input extension opcode
 1 4  CloseDevice opcode
 2 2  request length
 1 CARD8  device id
 3   unused



SetDeviceMode
 1 CARD8  input extension opcode
 1 5  SetDeviceMode opcode
 2 2  request length
 1 CARD8  device id
 1 CARD8  mode
 2   unused



 =>
 1 1   Reply
 1 5   SetDeviceMode opcode
 2 CARD16   sequence number
 4 0   reply length
 1 CARD8   status
  0  Success
   1 AlreadyGrabbed
  3 + first_error DeviceBusy
 23    unused



SelectExtensionEvent
 1 CARD8   input extension opcode
 1 6   SelectExtensionEvent opcode
 2 3+n    request length
 4 Window   event window
 2 CARD16   count
 2    unused
 4n LISTofEVENTCLASS desired events



GetSelectedExtensionEvents
 1 CARD8  input extension opcode
```

```
 1 7  GetSelectedExtensionEvents opcode
 2 2  request length
 4 Window  event window



 =>
 1 1  Reply
 1 7  GetSelecteExtensionEvents opcode
 2 CARD16  sequence number
 4 n + m    reply length
 2 n       this client count
 2 m       all clients count
 20   unused
 4n LISTofEVENTCLASS this client list
 4m LISTofEVENTCLASS all clients list



ChangeDeviceDontPropagateList
 1 CARD8   input extension opcode
 1 8   ChangeDeviceDontPropagateList opcode
 2 3+n    request length
 4 Window   event window
 2 n   count of events
 1 mode
  0 AddToList
  1 DeleteFromList
 1    unused
 4n LISTofEVENTCLASS desired events



GetDeviceDontPropagateList
 1 CARD8  input extension opcode
 1 9  GetDeviceDontPropagateList opcode
 2 2  request length
 4 Window  event window



 =>
 1 1   Reply
 1 9   GetDeviceDontPropagateList opcode
 2 CARD16   sequence number
 4 n     reply length
 2 n   count of events
 22    unused
 4n LISTofEVENTCLASS don't propagate list
```

```
GetDeviceMotionEvents
 1 CARD8  input extension opcode
 1 10  GetDeviceMotionEvents opcode
 2 4  request length
 4 TIMESTAMP  start
  0 CurrentTime
 4 TIMESTAMP  stop
  0 CurrentTime
 1 CARD8  device id
 3   unused



 =>
 1  1  Reply
 1  10 GetDeviceMotionEvents opcode
 2  CARD16 sequence number
 4  (m+1)n reply length
 4  n  number of DEVICETIMECOORDs in events
 1  m  number of valuators per event
 1  CARD8  mode of the device
   0 Absolute
   1 Relative
 18    unused
 (4m+4)n LISTofDEVICETIMECOORD events



  DEVICETIMECOORD
 4 TIMESTAMP  time
 4m LISTofINT32  valuators



ChangeKeyboardDevice
 1 CARD8  input extension opcode
 1 11  ChangeKeyboardDevice opcode
 2 2  request length
 1 CARD8  device id
 3   unused



 =>
 1 1  Reply
 1 11  ChangeKeyboardDevice opcode
 2 CARD16  sequence number
 4 0  reply length
 1   status
```

```
  0 Success
  1 AlreadyGrabbed
  2 DeviceFrozen
 23   unused



ChangePointerDevice
 1 CARD8  input extension opcode
 1 12  ChangePointerDevice opcode
 2 2  request length
 1 CARD8  x-axis
 1 CARD8  y-axis
 1 CARD8  device id
 1   unused



 =>
 1 1  Reply
 1 12  ChangePointerDevice opcode
 2 CARD16  sequence number
 4 0  reply length
 1   status
  0 Success
  1 AlreadyGrabbed
  2 DeviceFrozen
 23   unused



GrabDevice
 1 CARD8  input extension opcode
 1 13  GrabDevice opcode
 2 5+n   request length
 4 WINDOW  grab-window
 4 TIMESTAMP  time
  0 CurrentTime
 2 n  count of events
 1   this-device-mode
  0 Synchronous
  1 Asynchronous
 1   other-devices-mode
  0 Synchronous
  1 Asynchronous
 1 BOOL  owner-events
 1 CARD8  device id
 2   unused
 4n LISTofEVENTCLASS event list
```

```
 =>
 1 1  Reply
 1 13  GrabDevice opcode
 2 CARD16  sequence number
 4 0  reply length
 1   status
  0 Success
  1 AlreadyGrabbed
  2 InvalidTime
  3 NotViewable
  4 Frozen
 23   unused



UngrabDevice
 1 CARD8  input extension opcode
 1 14  UngrabDevice opcode
 2 3  request length
 4 TIMESTAMP  time
  0 CurrentTime
 1 CARD8  device id
 3   unused



GrabDeviceKey
 1 CARD8  input extension opcode
 1 15  GrabDeviceKey opcode
 2 5+n  request length
 4 WINDOW  grab-window
 2 n  count of events
 2 SETofKEYMASK  modifiers
  #x8000 AnyModifier
 1 CARD8  modifier device
  #x0FF UseXKeyboard
 1 CARD8  grabbed device
 1 KEYCODE   key
  0 AnyKey
 1   this-device-mode
  0 Synchronous
  1 Asynchronous
 1   other-devices-mode
  0 Synchronous
  1 Asynchronous
 1 BOOL  owner-events
 2   unused
 4n LISTofEVENTCLASS event list
```

```
UngrabDeviceKey
 1 CARD8  input extension opcode
 1 16  UngrabDeviceKey opcode
 2 4  request length
 4 WINDOW  grab-window
 2 SETofKEYMASK  modifiers
  #x8000 AnyModifier
 1 CARD8  modifier device
  #x0FF UseXKeyboard
 1 KEYCODE   key
  0 AnyKey
 1 CARD8  grabbed device
 3   unused




GrabDeviceButton
 1 CARD8  input extension opcode
 1 17  GrabDeviceButton opcode
 2 5+n  request length
 4 WINDOW  grab-window
 1 CARD8  grabbed device
 1 CARD8  modifier device
  #x0FF UseXKeyboard
 2 n  count of desired events
 2 SETofKEYMASK  modifiers
 1   this-device-mode
  0 Synchronous
  1 Asynchronous
 1   other-device-mode
  0 Synchronous
  1 Asynchronous
 1 BUTTON  button
  0 AnyButton
 1 BOOL  owner-events
  #x8000 AnyModifier
 2   unused
 4n LISTofEVENTCLASS event list




UngrabDeviceButton
 1 CARD8  input extension opcode
 1 18  UngrabDeviceButton opcode
 2 4  request length
 4 WINDOW  grab-window
 2 SETofKEYMASK  modifiers
  #x8000 AnyModifier
 1 CARD8  modifier device
  #x0FF UseXKeyboard
 1 BUTTON  button
```

```
  0 AnyButton
 1 CARD8  grabbed device
 3   unused



AllowDeviceEvents
 1 CARD8  input extension opcode
 1 19  AllowDeviceEvents opcode
 2 3  request length
 4 TIMESTAMP  time
  0 CurrentTime
 1   mode
  0 AsyncThisDevice
  1 SyncThisDevice
  2 ReplayThisDevice
  3 AsyncOtherDevices
  4 AsyncAll
  5 SyncAll
 1 CARD8  device id
 2   unused



GetDeviceFocus
 1 CARD8  input extension opcode
 1 20  GetDeviceFocus opcode
 2 2  request length
 1 CARD8  device
 3   unused



 =>
 1 1  Reply
 1 20  GetDeviceFocus opcode
 2 CARD16  sequence number
 4 0  reply length
 4 WINDOW  focus
  0 None
  1 PointerRoot
  3 FollowKeyboard
 4 TIMESTAMP focus time
 1   revert-to
  0 None
  1 PointerRoot
  2 Parent
  3 FollowKeyboard
 15   unused
```

```
SetDeviceFocus
 1 CARD8  input extension opcode
 1 21  SetDeviceFocus opcode
 2 4  request length
 4 WINDOW  focus
  0 None
  1 PointerRoot
  3 FollowKeyboard
 4 TIMESTAMP  time
  0 CurrentTime
 1   revert-to
  0 None
  1 PointerRoot
  2 Parent
  3 FollowKeyboard
 1 CARD8  device
 2   unused



GetFeedbackControl
 1 CARD8  input extension opcode
 1 22  GetFeedbackControl opcode
 2 2  request length
 1 CARD8  device id
 3   unused



 =>
 1 1  Reply
 1 22  GetFeedbackControl opcode
 2 CARD16  sequence number
 4 m/4   reply length
 2 n  number of feedbacks supported
 22   unused
 m LISTofFEEDBACKSTATE feedbacks



  FEEDBACKSTATE KBDFEEDBACKSTATE, PTRFEEDBACKSTATE, INTEGERFEEDBACKSTATE,
  STRINGFEEDBACKSTATE, BELLFEEDBACKSTATE, or LEDFEEDBACKSTATE



  KBDFEEDBACKSTATE
 1 0  feedback class id
 1 CARD8  id of this feedback
 2 20  length
 2 CARD16  pitch
```

```
2 CARD16  duration
4 CARD32  led_mask
4 CARD32  led_values
1       global_auto_repeat
 0 Off
 1 On
1 CARD8  click
1 CARD8  percent
1       unused
32 LISTofCARD8 auto_repeats



 PTRFEEDBACKSTATE
1 0  feedback class id
1 CARD8  id of this feedback
2 12  length
2   unused
2 CARD16  acceleration-numerator
2 CARD16  acceleration-denominator
2 CARD16  threshold



 INTEGERFEEDBACKSTATE
1 0  feedback class id
1 CARD8  id of this feedback
2 16  length
4 CARD32  resolution
4 INT32  minimum value
4 INT32  maximum value



 STRINGFEEDBACKSTATE
1 1  feedback class id
1 CARD8  id of this feedback
2 4n+8  length
2 CARD16  max_symbols
2 n        number of keysyms supported
4n LISTofKEYSYM  key symbols supported



 BELLFEEDBACKSTATE
1 1  feedback class id
1 CARD8  id of this feedback
2 12  length
1 CARD8  percent
3   unused
2 CARD16  pitch
```

```
 2 CARD16  duration



  LEDFEEDBACKSTATE
 1 1  feedback class id
 1 CARD8  id of this feedback
 2 12  length
 4 CARD32  led_mask
 4 BITMASK  led_values
  #x0001 On
  #x0002 Off



ChangeFeedbackControl
 1 CARD8  input extension opcode
 1 23  ChangeFeedbackControl opcode
 2 3+n/4  request length
 4 BITMASK  value-mask (has n bits set to 1)
  #x0001 keyclick-percent
  #x0002 bell-percent
  #x0004 bell-pitch
  #x0008 bell-duration
  #x0010 led
  #x0020 led-mode
  #x0040 key
  #x0080 auto-repeat-mode
  #x0001 string
  #x0001 integer
  #x0001 acceleration-numerator
  #x0002 acceleration-denominator
  #x0004 acceleration-threshold
 1 CARD8  device id
 1 CARD8  feedback class id
 2   unused
 n FEEDBACKCLASS



  FEEDBACKCLASS KBDFEEDBACKCTL, PTRFEEDBACKCTL, INTEGERFEEDBACKCTL,
  STRINGFEEDBACKCTL, BELLFEEDBACKCTL, or LEDFEEDBACKCTL



  KBDFEEDBACKCTL
 1 0  feedback class id
 1 CARD8  id of this feedback
 2 20  length
 1 KEYCODE  key
 1   auto-repeat-mode
```

```
 0 Off
 1 On
 2 Default
1 INT8  key-click-percent
1 INT8  bell-percent
2 INT16  bell-pitch
2 INT16  bell-duration
4 CARD32  led_mask
4 CARD32  led_values



 PTRFEEDBACKCTL
1 1  feedback class id
1 CARD8  id of this feedback
2 12  length
2   unused
2 INT16  numerator
2 INT16  denominator
2 INT16  threshold



 STRINGCTL
1 2  feedback class id
1 CARD8  id of this feedback
2 4n+8  length
2   unused
2 n  number of keysyms to display
4n LISTofKEYSYM  list of key symbols to display



 INTEGERCTL
1 3  feedback class id
1 CARD8  id of this feedback
2 8  length
4 INT32  integer to display



 LEDCTL
1 4  feedback class id
1 CARD8  id of this feedback
2 12  length
4 CARD32  led_mask
4 BITMASK  led_values
 #x0001 On
 #x0002 Off
```

```
 BELLCTL
 1 5  feedback class id
 1 CARD8  id of this feedback
 2 8  length
 1 INT8  percent
 3   unused
 2 INT16  pitch
 2 INT16  duration



GetDeviceKeyMapping
 1 CARD8  input extension opcode
 1 24  GetDeviceKeyMapping opcode
 2 2  request length
 1 CARD8  device
 1 KEYCODE  first-keycode
 1 CARD8  count
 1   unused



 =>
 1 1  Reply
 1 24  GetDeviceKeyMapping opcode
 2 CARD16  sequence number
 4 nm  reply length (m = count field from the request)
 1 n  keysyms-per-keycode
 23   unused
 4nm LISTofKEYSYM  keysyms



ChangeDeviceKeyMapping
 1 CARD8  input extension opcode
 1 25  ChangeDeviceKeyMapping opcode
 2 2+nm  request length
 1 CARD8  device
 1 KEYCODE  first-keycode
 1 m  keysyms-per-keycode
 1 n  keycode-count
 4nm LISTofKEYSYM  keysyms



GetDeviceModifierMapping
 1 CARD8  input extension opcode
 1 26  GetDeviceModifierMapping opcode
```

```
 2 2  request length
 1 CARD8  device
 3   unused



 =>
 1 1  Reply
 1 26  GetDeviceModifierMapping opcode
 2 CARD16  sequence number
 4 2n  reply length
 1 n  keycodes-per-modifier
 23   unused
 8n LISTofKEYCODE  keycodes



SetDeviceModifierMapping
 1 CARD8  input extension opcode
 1 27  SetDeviceModifier opcode
 2 2+2n  request length
 1 CARD8  device
 1 n  keycodes-per-modifier
 2   unused
 8n LISTofKEYCODE  keycodes



 =>
 1 1  Reply
 1 27  SetDeviceModifierMapping opcode
 2 CARD16  sequence number
 4 0  reply length
 1   status
  0 Success
  1 Busy
  2 Failed
 23   unused



GetDeviceButtonMapping
 1 CARD8  input extension opcode
 1 28  GetDeviceButtonMapping opcode
 2 2  request length
 1 CARD8  device
 3   unused
```

```
 =>
 1 1  Reply
 1 28  GetDeviceButtonMapping opcode
 2 CARD16  sequence number
 4 (n+p)/4  reply length
 1 n  number of elements in map list
 23   unused
 n LISTofCARD8   map
 p   unused, p=pad(n)



SetDeviceButtonMapping
 1 CARD8  input extension opcode
 1 29  SetDeviceButtonMapping opcode
 2 2+(n+p)/4  request length
 1 CARD8  device
 1 n  length of map
 2   unused
 n LISTofCARD8   map
 p   unused, p=pad(n)



 =>
 1 1  Reply
 1 29  SetDeviceButtonMapping opcode
 2 CARD16  sequence number
 4 0  reply length
 1   status
  0 Success
  1 Busy
 23   unused



QueryDeviceState
 1 CARD8  input extension opcode
 1 30  QueryDeviceState opcode
 2 2  request length
 1 CARD8  device
 3   unused



 =>
 1 1   Reply
 1 30   QueryDeviceState opcode
 2 CARD16   sequence number
 4 m/4   reply length
 1 n         number of input classes
```

```
23    unused
m LISTofINPUTSTATE
```

INPUTSTATE KEYSTATE or BUTTONSTATE or VALUATORSTATE

```
KEYSTATE
1 CARD8  key input class id
1 36  length
1 CARD8  num_keys
1   unused
32 LISTofCARD8  status of keys
```

```
BUTTONSTATE
1 CARD8  button input class id
1 36  length
1 CARD8  num_buttons
1   unused
32 LISTofCARD8   status of buttons
```

```
VALUATORSTATE
1 CARD8  valuator input class id
1 4n + 4  length
1 n  number of valuators
1   mode
 #x01 DeviceMode (0 = Relative, 1 = Absolute)
 #x02 ProximityState (0 = InProximity, 1 = OutOfProximity)
4n LISTofCARD32  status of valuators
```

```
SendExtensionEvent
1 CARD8   input extension opcode
1 31   SendExtensionEvent opcode
2 4 + 8n + m  request length
4 WINDOW   destination
1 CARD8   device
1 BOOL   propagate
2 CARD16   eventclass count
1 CARD8   num_events
3    unused
32n LISTofEVENTS  events to send
4m LISTofEVENTCLASS desired events
```

```
DeviceBell
 1 CARD8  input extension opcode
 1 32  DeviceBell opcode
 2 2  request length
 1 CARD8  device id
 1 CARD8  feedback id
 1 CARD8  feedback class
 1 INT8  percent



SetDeviceValuators
 1 CARD8  input extension opcode
 1 33  SetDeviceValuators opcode
 2 2 + n  request length
 1 CARD8  device id
 1 CARD8  first valuator
 1 n      number of valuators
 1   unused
 4n LISTofINT32 valuator values to set



 =>
 1 1   Reply
 1 33   SetDeviceValuators opcode
 2 CARD16   sequence number
 4 0   reply length
 1 CARD8   status
  0 Success
  1 AlreadyGrabbed
 23    unused



GetDeviceControl
 1 CARD8  input extension opcode
 1 34  GetDeviceControl opcode
 2 2  request length
 2 CARD16  device control type
 1 CARD8  device id
 1   unused



 =>
 1 1   Reply
 1 34   GetDeviceControl opcode
```

```
 2 CARD16   sequence number
 4 n/4   reply length
 1 CARD8   status
  0   Success
  1   AlreadyGrabbed
  3 + first_error  DeviceBusy
 23    unused
 n    DEVICESTATE
```

```
DEVICESTATE   DEVICERESOLUTIONSTATE
```

```
DEVICERESOLUTIONSTATE
  2 0  control type
  2 8 + 12n  length
  4 n      num_valuators
  4n LISTOfCARD32 resolution values
  4n LISTOfCARD32 resolution min_values
  4n LISTOfCARD32 resolution max_values
```

```
ChangeDeviceControl
  1 CARD8  input extension opcode
  1 35  ChangeDeviceControl opcode
  2 2+n/4  request length
  2 CARD16  control type
  1 CARD8  device id
  1   unused
  n   DEVICECONTROL
```

```
DEVICECONTROL  DEVICERESOLUTIONCTL
```

```
DEVICERESOLUTIONCTL
  2 1  control type
  2 8 + 4n  length
  1 CARD8  first_valuator
  1 n      num_valuators
  2   unused
  4n LISTOfCARD32 resolution values
```

```
 =>
  1 1   Reply
  1 35   ChangeDeviceControl opcode
  2 CARD16   sequence number
  4 0   reply length
  1 CARD8   status
   0   Success
   1   AlreadyGrabbed
   3 + first_error  DeviceBusy
  23    unused
```

9+

```
=+ C      =+ C      =+ E)      =+ E)
 .' ,   .' :)   =+    A * +  '   *     F
 ' =+   )   + !=+    A * +      *
 ' =+   )   + !
```

```
DeviceValuator
 1 CARD8  code
 1 CARD8  device id
 2 CARD16  sequence number
 2 SETofKEYBUTMASK  state
 1 n  number of valuators this device reports
 1 m  number of first valuator in this event
 24 LISTofINT32 valuators
```

```
DeviceKeyPress
 1 CARD8  code
 1 KEYCODE  detail
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
  0 None
 2 INT16  root-x
 2 INT16  root-y
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK  state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow
```

```
DeviceKeyRelease
 1 CARD8  code
 1 KEYCODE  detail
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
  0 None
 2 INT16  root-x
 2 INT16  root-y
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK   state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow



DeviceButtonPress
 1 CARD8  code
 1 BUTTON  detail
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
  0 None
 2 INT16  root-x
 2 INT16  root-y
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK  state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow



DeviceButtonRelease
 1 CARD8  code
 1 BUTTON  detail
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
  0 None
 2 INT16  root-x
 2 INT16  root-y
```

```
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK  state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow



DeviceMotionNotify
 1 CARD8  code
 1   detail
  0 Normal
  1 Hint
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
   0 None
 2 INT16  root-x
 2 INT16  root-y
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK  state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow



DeviceFocusIn
 1 CARD8  code
 1   detail
  0 Ancestor
  1 Virtual
  2 Inferior
  3 Nonlinear
  4 NonlinearVirtual
  5 Pointer
  6 PointerRoot
  7 None
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  event
 1   mode
  0 Normal
  1 Grab
  2 Ungrab
  3 WhileGrabbed
 1 CARD8  device id
 18   unused
```

```
DeviceFocusOut
 1 CARD8  code
 1   detail
  0 Ancestor
  1 Virtual
  2 Inferior
  3 Nonlinear
  4 NonlinearVirtual
  5 Pointer
  6 PointerRoot
  7 None
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  event
 1   mode
  0 Normal
  1 Grab
  2 Ungrab
  3 WhileGrabbed
 1 CARD8  device id
 18   unused



ProximityIn
 1 CARD8  code
 1   unused
 2 CARD16  sequence number
 4 TIMESTAMP  time
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
  0 None
 2 INT16  root-x
 2 INT16  root-y
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK   state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow



ProximityOut
 1 CARD8  code
 1   unused
 2 CARD16  sequence number
 4 TIMESTAMP  time
```

```
 4 WINDOW  root
 4 WINDOW  event
 4 WINDOW  child
  0 None
 2 INT16  root-x
 2 INT16  root-y
 2 INT16  event-x
 2 INT16  event-y
 2 SETofKEYBUTMASK   state
 1 BOOL  same-screen
 1 CARD8  device id
  #x80  MORE_EVENTS follow
```

=+ A * + ' " * F =+ C
A * 5 F ' =+) + !

```
DeviceStateNotify
 1 CARD8  code
 1 CARD8  device id
  #x80  MORE_EVENTS follow
 2 CARD16  sequence number
 4 TIMESTAMP  time
 1 CARD8  num_keys
 1 CARD8  num_buttons
 1 CARD8  num_valuators
  1 CARD8  valuator mode and input classes reported
   #x01 reporting keys
   #x02 reporting buttons
   #x04 reporting valuators
   #x40 device mode (0 = Relative, 1 = Absolute)
   #x80 proximity state (0 = InProximity, 1 = OutOfProximity)
 4 LISTofCARD8  first 32 keys (if reported)
 4 LISTofCARD8  first 32 buttons (if reported)
 12 LISTofCARD32  first 3 valuators (if reported)


DeviceKeyStateNotify
 1 CARD8  code
 1 CARD8  device id
  #x80  MORE_EVENTS follow
 2 CARD16  sequence number
 28 LISTofCARD8  state of keys 33-255


DeviceButtonStateNotify
 1 CARD8  code
 1 CARD8  device id
  #x80  MORE_EVENTS follow
```

```
 2 CARD16  sequence number
 28 LISTofCARD8  state of buttons 33-255



DeviceValuator
 1 CARD8  code
 1 CARD8  device id
 2 CARD16  sequence number
 2 SETofKEYBUTMASK  state
 1 n  number of valuators this device reports
 1 n  number of first valuator in this event
 24 LISTofINT32 valuators



DeviceMappingNotify
 1 CARD8  code
 1 CARD8  device id
 2 CARD16  sequence number
 1   request
  0 MappingModifier
  1 MappingKeyboard
  2 MappingPointer
 1 KEYCODE  first-keycode
 1 CARD8  count
 1   unused
 4 TIMESTAMP  time
 20   unused



ChangeDeviceNotify
 1 CARD8  code
 1 CARD8  id of device specified on change request
 2 CARD16  sequence number
 4 TIMESTAMP  time
 1   request
  0 NewPointer
  1 NewKeyboard
 23   unused
```