X11 Input Extension Porting Document

George Sachs, Hewlett-Packard

X11 Input Extension Porting Document

© !" # $ % &

& ' & % (% % ' % '& (' % ! ' (%
% % % & "" !" # $ % & $
' ' " (' ((& % '&) % % * *
! ' + & " % ! , % '& " % (% % ('& % (
' -

. '&

& % ((((! % % % '
& (" / 0 (! 12 % " (! ! ' "'% ! ' " &
' & % (& ' " % ' ' " %3 "" ((! % &
! & (! (' % % ' - ("" ! % 4

, % & "" "'% % "" ' " (
(!

, 5 67,89:5) #:6);5; *9) * 8), 6<, 89::9=,> 67 9=> ?)=; 5 #:5 6:)@#A)5;)= A<;)=
B<, =6, A)@),5; ,6 , 5 89::9=,)5 67 @5: 9=,9B)A),> 7),=5 76: 9 #9:,) <A9: #<:#6 5 9=;
=6=)=7:)= 5@5=,)= =6 5 5=, 9AA , 5 6= 6:,)<@ B5 A)9BA5 76: 9=> A9)@ ;9@9 5 6:
6, 5: A)9B)A),> 8 5, 5:)= 9= 9 ,)6= 67 6=,:9 , ,6:, 6: 6, 5:8) 5 9:))= 7:6@ 6<, 67
6:)= 6==5 ,)6= 8), , 5 67,89:5 6: , 5 < 5 6: 6, 5: ;59A)=)= , 5 67,89:5

5+ % & ('& "" ' % % !
& " ' % " (! ! ' ! ' C (& '&

8 % ! & % & $ (, 6 '

Table of Contents

) ' 5+ # ; '&
) " C 5+ ;
'&& ("" D'
) " C "" % 7 &)) ' E
) " C "" % 7 &) 9 % ; N
;)) ' ") " C : ' L
) " C , ; = & 9 % ,
" 5+ ;
)& " & ; % : '
9%%6) ' ;
6) ' ;
") ' ;
; @ %
; ""
;
? %; E
) ' 5+ 5 E
; ? 5 N
; B' 5 J
; @ 5 J
; # + & 5 L

Chapter 1. X11 Input Extension Porting Document

, % '& % % % ( ) ' 5+

@ ((' " % % ' + % % & " &
% % % '"% D' , (' " & " & %
' "" % ' % + 3
3+ ' , + '"% (' " % % " ' +
% " ' % (' % D' % $ % '
& % , " ' + D' & (% %
% (' % " " C % % '
(& % " % ; % % " (
% ' % (& ' % %! % ((&

, (" C ' + % & " ' % (
% % % % % ("" ! 8 &'" " ' %
% (! % "" '
% $ % " (& " & % %) " ' & "
& % % ! "" ' % ! "" % %'
" C % $ (" ((, "
" $ % % %' " C % '
% " ! D' % " % 5 (& " &
' % ' +

) ' + % ' & E +5 !
% ' ! %% " (& &' (
' + % D' &'" " +5 % &
% % (& ' + % , +5 & %
" " 5 ' + " 9 " (% '&
% (& % (' +

Initializing Extension Devices

5+ ' % " C % & & ' %
;) % % % (' "" % (& ;;
" C % 8 (' "" % % ! ! "" & " &
% ! "" % % ! & " & "" ' %
" " ! " C % ! ' " " D' %
%) & " ;; ! "" "" ' % ("
C !)) ' ' "" %

Summary of Calling Sequence

```
Device-Independent X        |  Device-Dependent X
--------------------        |  ------------------
                            |
InitInput -------------->   |  - do device-specific initialization
```

```
                                |
                                |  - call AddInputDevice  (deviceProc,AutoStart)
AddInputDevice                  |
  - creates DeviceIntRec        |
  - records deviceProc          |
  - adds new device to          |
    list of off_devices.        |
sets dev->startup=AutoStart     |
                                |  - call one of:
                                |    - RegisterPointerDevice (X pointer)
                                |      - processInputProc = ProcessPointerEvents
                                |    - RegisterKeyboardDevice (X keyboard)
                                |      - processInputProc = ProcessKeyboardEvents
                                |    - RegisterOtherDevice  (extension device)
                                |      - processInputProc = ProcessOtherEvents
                                |
                                |
InitAndStartDevices ----->      |  - calls deviceProc with parameters
                                |    (DEVICE_INIT, AutoStart)
sets dev->inited = return       |
  value from deviceProc         |
                                |
                                |  - in deviceProc, do one of:
                                |    - call InitPointerDeviceStruct (X pointer)
                                |    - call InitKeyboardDeviceStruct (X keybd)
                                |    - init extension device by calling some of:
                                |      - InitKeyClassDeviceStruct
                                |      - InitButtonClassDeviceStruct
                                |      - InitValuatorClassDeviceStruct
                                |      - InitValuatorAxisStruct
                                |      - InitFocusClassDeviceStruct
                                |      - InitProximityClassDeviceStruct
                                |      - InitKbdFeedbackClassDeviceStruct
                                |      - InitPtrFeedbackClassDeviceStruct
                                |      - InitLedFeedbackClassDeviceStruct
                                |      - InitStringFeedbackClassDeviceStruct
                                |      - InitIntegerFeedbackClassDeviceStruct
                                |      - InitBellFeedbackClassDeviceStruct
                                |    - init device name and type by:
                                |      - calling MakeAtom with one of the
                                |        predefined names
                                |      - calling AssignTypeAndName
                                |
                                |
for each device added           |
    by AddInputDevice,          |
    InitAndStartDevices         |
    calls EnableDevice if       |  - EnableDevice calls deviceProc with
    dev->startup &              |    (DEVICE_ON, AutoStart)
    dev->inited                 |
                                |
If deviceProc returns           |  - core devices are now enabled, extension
    Success, EnableDevice       |    devices are now available to be accessed
    move the device from        |    through the input extension protocol
```

```
    inputInfo.off_devices   |    requests.
    to inputInfo.devices    |
```

Initialization Called From InitInput

```
) ) '    (   ;;   '        "" % %'          '  ,   '
        " (  %     %  & " &        (     " C     % (
 ""   9%%) ';          %    " C   ;   ) :   ' ' (
  ' %    9%%) ';        %   %%   (     %'      "" %
   ;)  '   ) 9 %   ;    !   ' %        " % ,
%'   +   %    ( &    " C   (     ' %

)(   %        ' %          ;;   '"%     "" :    #
;          ;   ) :          " C   %

)(   %        ' %     $    % ;;   '"%    %  "" :    ?
  %;        " C   %        $   %

)(   %        ' %     +    %    ;;   '"%    %  "" :
  6  ;          ;   ) #   '  %   9%%) ';

9  & " ) ) ' & " &       !  " !


InitInput(argc,argv)
    {
    int i, numdevs, ReadInput();
    DeviceIntPtr dev;
    LocalDevice localdevs[LOCAL_MAX_DEVS];
    DeviceProc kbdproc, ptrproc, extproc;

    /*************************************************************
     * Open the appropriate input devices, determine which are
     * available, and choose an X pointer and X keyboard device
     * in some implementation-dependent manner.
     *************************************************************/

    open_input_devices (&numdevs, localdevs);

    /*************************************************************
     * Register a WakeupHandler to handle input when it is generated.
     *************************************************************/

    RegisterBlockAndWakeupHandlers (NoopDDA, ReadInput, NULL);

    /*************************************************************
     * Register the input devices with DIX.
     *************************************************************/

    for (i=0; i<numdevs; i++)
        {
        if (localdevs[i].use == IsXKeyboard)
            {
            dev = AddInputDevice (kbdproc, TRUE);
            RegisterKeyboardDevice (dev);
```

```
        }
    else if (localdevs[i].use == IsXPointer)
        {
        dev = AddInputDevice (ptrproc, TRUE);
        RegisterPointerDevice (dev);
        }
    else
        {
        dev = AddInputDevice (extproc, FALSE);
        RegisterOtherDevice (dev);
        }
    if (dev == NULL)
        FatalError ("Too many input devices.");
    dev->devicePrivate = (pointer) &localdevs[i];
    }
```

Initialization Called From InitAndStartDevices

9( )) ' ' %) 9 % ; ;) ' "" %
" ' %) "" % " ' ! % 9%
%) ' ; ! & % "' (;5) 5F)=), , $ %
" ' % % ! % ' %)(%
% " ' '"% "") # ; ' "
C)(% $ % % " ' '"% ""
) ? %; ' ' + % & ' ' &
($ ' "' % (% $ " (' '
&' "C % 5 % (% ;) "C (
' % " (' % % % ("" !

9 &" % " ' "" % (&) 9 % ; ! "!

```
Bool extproc (dev, mode)
    DeviceIntPtr dev;
    int mode;
    {
    LocalDevice *localdev = (LocalDevice *) dev->devicePrivate;

    switch (mode)
        {
        case DEVICE_INIT:
            if (strcmp(localdev->name, XI_TABLET) == 0)
                {
                /*************************************************
                 * This device reports proximity, has buttons,
                 * reports two axes of motion, and can be focused.
                 * It also supports the same feedbacks as the X pointer
                 * (acceleration and threshold can be set).
                 *************************************************/

                InitButtonClassDeviceStruct (dev, button_count, button_map);
                InitValuatorClassDeviceStruct (dev, localdev->n_axes,);
                    motionproc, MOTION_BUF_SIZE, Absolute);
                for (i=0; i<localdev->n_axes; i++)
```

```
                    InitValuatorAxisStruct (dev, i, min_val, max_val,
                        resolution);
                InitFocusClassDeviceStruct (dev);
                InitProximityClassDeviceStruct (dev);
                InitPtrFeedbackClassDeviceStruct (dev, p_controlproc);
                }
            else if (strcmp(localdev->name, XI_BUTTONBOX) == 0)
                {
                /**************************************************
                 * This device has keys and LEDs, and can be focused.
                 **************************************************/

                InitKeyClassDeviceStruct (dev, syms, modmap);
                InitFocusClassDeviceStruct (dev);
                InitLedFeedbackClassDeviceStruct (dev, ledcontrol);
                }
            else if (strcmp(localdev->name, XI_KNOBBOX) == 0)
                {
                /**************************************************
                 * This device reports motion.
                 * It can be focused.
                 **************************************************/

                InitValuatorClassDeviceStruct (dev, localdev->n_axes,);
                    motionproc, MOTION_BUF_SIZE, Absolute);
                for (i=0; i<localdev->n_axes; i++)
                    InitValuatorAxisStruct (dev, i, min_val, max_val,
                        resolution);
                InitFocusClassDeviceStruct (dev);
                }
            localdev->atom =
                MakeAtom(localdev->name, strlen(localdev->name), FALSE);
            AssignTypeAndName (dev, localdev->atom, localdev->name);
            break;
        case DEVICE_ON:
            AddEnabledDevice (localdev->file_ds);
            dev->on = TRUE;
            break;
        case DEVICE_OFF:
            dev->on = FALSE;
            RemoveEnabledDevice (localdev->file_ds);
            break;
        case DEVICE_CLOSE:
            break;
        }
    }

, %     " '     "" % !    & %   "'  ( ;5 ) 5F6=     ;)
 '   5  " ;    !     "" % ( & )  9 %   ;    8    "" % !
   & %     '"%  "" 9%%5  " %;      '              $
(    " "   ' ( &    %

G7  & )  9 %   ;     5  " ;      "" % (   "" %
*   %*  % *   ' *( "%     ;    ) :     ,:<5 ,  *   %*( "%
```

) 9 % ; "' ' % % ! "" % !
& % "' (;5) 5F)=), , * ' * ("% 9%%) ' ; "'
(% & / ' 2

8 (" C % '"% " $ (' (&
$ % % 6 ! & " "" 9%%) ' ;
($ % % ! ' "' D' " ,:<5 ! "
"" 9%%) ' ; (' + % ! ' "' D' "
79A 5)(% 5 " ; ! "" $ "" ' + % %'
" C) 6) ' ; ' '"% *
' * ("% ,:<5 ! "" % (' + % , ! "" ' #
6) ' ; "" 5 " ; (% ! " (%
6 ; D'

DIX Input Class Initialization Routines

;) ' % (% " C (% (% ' " , % (%
" 4

H ? " % $

H B' " % '

H "' " % & % " %

H # + & " % + & (&

H 7 ' " % (' %

H 7 % $ " % ' & $ % ((% $

;) ' % % " C % $ % '
" (;' " C) # ; ' "" % " C
%) ? %; ' "" % " C $ %
, % ' (+ ' % % ""
' & " (') % ;; " (" C
(' " ' % + % 9 % ('
% % ;) (& " C ("" !

InitKeyClassDeviceStruct

, (' % % "" % " C ? " : % '"%
"" % (+ % $) % %
% ($ & % & % (% %) '
79A 5 (? " : '"% "" % (& ($ &
% % & % ('"% "" %) & 4

```
Bool
InitKeyClassDeviceStruct(dev, pKeySyms, pModifiers)
    DeviceIntPtr dev;
    KeySymsPtr pKeySyms;
    CARD8 pModifiers[];
```

, ;)) ? %; ' "" ' ($
%) &' "" % + " " (+ % $

InitButtonClassDeviceStruct

, (' % % "" % " C B' " : % '"%
"" % (+ % ') % %
'& (' ' % % & (% ' %) '
79A 5 (B' " : '"% "" %) & 4

```
Bool
InitButtonClassDeviceStruct(dev, numButtons, map)
    register DeviceIntPtr dev;
    int numButtons;
    CARD8 *map;
```

, ;)) # ; ' "" ' (
) &' "" % +" " (+ % '

InitValuatorClassDeviceStruct

, (' % % "" % " C '" " : % '"%
"" % (+ % '") % '& (+
(& % % %% (& %' (
% C (& '((% & % /9 '" : " 2
(%) ' 79A 5 ('" " : '"% "" %)
& 4

```
Bool
InitValuatorClassDeviceStruct(dev, numAxes, motionProc, numMotionEvents, mode)
    DeviceIntPtr dev;
    int (*motionProc)();
    int numAxes;
    int numMotionEvents;
    int mode;
```

, ;)) # ; ' "" ' (
) &' "" % +" " (+ % &

InitValuatorAxisStruct

, (' % % " C 9+) (: % '"% "" % (
% + % '" , (9+) (: ""
%) '" " ; ' (' ' " C %

) '" 9+ ' '"% "" % (+ (& %
% 5 '"% % + '& / ! 2
& &'& '" (+ & + &'& '" (+ % '"
(% ' &)(% " & '"%
% & &'& % & + &'& '") '" 9+ '
("" ! & 4

```
InitValuatorAxisStruct(dev, axnum, minval, maxval, resolution)
    DeviceIntPtr dev;
    int axnum;
```

```
    int minval;
    int maxval;
    int resolution;
```

, ' "" %) # ; ' ( ) &'
"" % + " " (% + % &

InitFocusClassDeviceStruct

, (' % % "" % " C 7 ' " : % "" %
"" % (+ % (' %) % %
% ' 79A 5 ("" (") ("" ! & 4

```
Bool
InitFocusClassDeviceStruct(dev)
    DeviceIntPtr dev;
```

, ;)) ? %; ' "" ' ($
%) &' "" % + " " (+ % (' % 8
"" % (' % " (& " & % %

InitProximityClassDeviceStruct

, (' % % "" % " C # + & " : % "" %
"" % (+ "" % + &) %
% % ' 79A 5 ("" (") ("" !
& 4

```
Bool
InitProximityClassDeviceStruct(dev)
    DeviceIntPtr dev;
```

Initializing Feedbacks

InitKbdFeedbackClassDeviceStruct

, (' % % "" % " C ? %7 % $ " : % &
"" % (+ % ' & "" ((% $
$ % ') % %
%' '% "" % % " %') '
79A 5 ("" (" % ("" ! & 4

```
Bool
InitKbdFeedbackClassDeviceStruct(dev, bellProc, controlProc)
    DeviceIntPtr dev;
    void (*bellProc)();
    void (*controlProc)();
```

, ;)) ? %; ' "" ' ($
%) &' "" % + " " (+ % & (%
$ $ % & (% $ ' A5; % "" ' %

! ?%7 % $" ! B""7 % $" % A %7 % $"
(% $

InitPtrFeedbackClassDeviceStruct

, (' % % "" % "C # 7 % $" : %
'"% "" % (+ % "" ! (" %
"%) % % % % "
%') ' 79A 5 ("" (" % ("" ! & 4

```
Bool
InitPtrFeedbackClassDeviceStruct(dev, controlProc)
    DeviceIntPtr dev;
    void (*controlProc)();
```

, ;)) # ; ' "" ' (
) &' "" % +" " (+ % ' ("
% "%

InitLedFeedbackClassDeviceStruct

, (' % % "" % "C A %7 % $" : %
'"% "" % (+ % A5;) %
% % % " %') ' 79A 5 (""
(" % ("" ! & 4

```
Bool
InitLedFeedbackClassDeviceStruct(dev, controlProc)
    DeviceIntPtr dev;
    void (*controlProc)();
```

< E A5; (% $ ' % % % & &'" "
(% $ (&

InitBellFeedbackClassDeviceStruct

, (' % % "" % "C B""7 % $" : %
'"% "" % (+ % "") %
% % % " %') ' 79A 5 (""
(" % ("" ! & 4

```
Bool
InitBellFeedbackClassDeviceStruct(dev, bellProc, controlProc)
    DeviceIntPtr dev;
    void (*bellProc)();
    void (*controlProc)();
```

InitStringFeedbackClassDeviceStruct

, (' % % "" % "C 7 % $" : %
'"% "" % (+ % % " ' !
% " %) % % % % "
%') ' 79A 5 ("" (" % ("" ! & 4

```
Bool
InitStringFeedbackClassDeviceStruct(dev, controlProc, max_symbols,
 num_symbols_supported, symbols)
    DeviceIntPtr dev;
    void (*controlProc)();
    int max_symbols:
    int num_symbols_supported;
    KeySym *symbols;
```

InitIntegerFeedbackClassDeviceStruct

, (' % % "" % "C) 7 % $" : %
'"% "" %(+ % % " ' !
% " %) % % % % "
%') ' 79A 5 ("" (" % ("" ! & 4

```
Bool
InitIntegerFeedbackClassDeviceStruct(dev, controlProc)
    DeviceIntPtr dev;
    void (*controlProc)();
```

Initializing The Device Name And Type

, % & % "C % "" 9 , 9 %= & !
("" ! & 4

```
void
AssignTypeAndName(dev, type, name)
    DeviceIntPtr dev;
    Atom type;
    char *name;
```

, !"" "" (% & % & ! % ,
% % "" @ $ 9 & ! (& %(%(
' % @ $ 9 & ("" ! & 4

```
Atom
MakeAtom(name, len, makeit)
    char *name;
    int len;
    Bool makeit;
```

&! " % & % ! ' + ! "C % "'
(& $ '"% 79A 5I

Closing Extension Devices

, ; "; % % ;) % " ' %) ""
% " ' ((% % ! & % "' (;5) 5F677
, % " ' '"% "": & 5 "%; (&
$ (' (& %

; " ; "" % ' + ') "" % (&
") ' ; ' ! "" % # " ; ! " & $
" ; D')(; " ; "" % "% " "" % !
" " ' + % & % "" % " ;

Implementation-Dependent Routines

" ' + " D' & " & % %
; ('" ' % (% (% %
' (" + 3 3+ ' 3+ ' & & " & & "
' % ('" ' ! ' , ("" ! %
('

AddOtherInputDevices

9%%6) ' ; "" % (&# A) ' ; '" (A)
' ; " D') & % % & " & %
+ ' % ' " D' % % & " , & " &
& "C "" % ! ' ' & (
% & ' A) ' ; (' " "
% "C % 9%%6) ' ; "" % ;;
"C '" ' " " ' %

9 & " 9%%6) ' ; ' & " $ " $ ("" ! 4

```
void
AddOtherInputDevices ()
    {
    DeviceIntPtr dev;
    int i;

    for (i=0; i<MAX_DEVICES; i++)
        {
        if (!local_dev[i].initialized && available(local_dev[i]))
            {
            dev = (DeviceIntPtr) AddInputDevice (local_dev[i].deviceProc, TRUE);
            dev->public.devicePrivate = local_dev[i];
            RegisterOtherDevice (dev);
            dev->inited = ((*dev->deviceProc)(dev, DEVICE_INIT) == Success);
            }
        }
    }
```

, % ('" 9%%6) ' ; ' + ' %)("" ' +
% "C % ! ' "("""
'

OpenInputDevice

& & " & "" ' % ! "C %
% " & 6 & " & & " % $
% % %' "C % ' % " !

& " & $ 6 ; D' , ("
(& " &

)(" % " % % ')
&' $ $ ((& " %
% "" " ! " " & $ " ;
D'

, % ('" & " & % / '& "" ' % %
%' " C % $ 2

CloseInputDevice

& & " & & " ' % ! " " '
% D' " % & ") ' ; "" % (&
# " ; ! " & $ " ; " D'

, % ('" & " & % / '& "" ' % %
%' " C % $ 2

SetDeviceMode

& & " & ' ' % "'
" % " & , ; @ % " D' % %
"" ! ;; ' & % (' %

, % ('" & " & "! ' B %@)(& " &
% ' ' % " ("
& % "' (& % ('" & " & & " (
' %

SetDeviceValuators

& & " & ' ' % "" ! "'
" "' , ; "' " D' % % "" ! ;;
"' (' %

, % ('" & " & "! ' B %@)(& " &
% ' ' % "" ! "'
% ('" & " & & " (' %

ChangePointerDevice

, # ; " D' % % ! %
' % & & " & & (& (
% ' ' % ;) :
# ; "" % "" ! ' & " & & (&
! % , ' " (' + & " (
((& & ((%

, ;) : ' ' % % %
7 ' :)(% & % ! ! ' "
% '"% (' % # # ; ! "" (7 ' :
% ! %

)(& " & % "" ! " (' "% %
/! ! " ' ' + 2 '"% "") 7 ' "
; ' ("% %

, # ; " D' " "" ! " !
+ (! % ' % & ' % > %
)(+ % ((% ('" & " &
'"% % (

)(& " & ' ' % ! '" "
" ! % ' % '"% % ' ' %
B %; ' %

, % ('" & " & %

ChangeKeyboardDevice

, ? %; " D' % % ! %
' % $ % & & " & & (& (
$ % % ' ' % ;) :
? %; "" % "" ! ' & " & & (
& ! $ % %

, $ % % (' % % ;) : %
% 7 ':)(% & % ! $ %
% % '" 7 ': # ? %; !"" ""
(

)(& " & % ! " " (' "% $ %
/! ! & " " ' + % 2 '"% "";"
7 ' " ; ' (7 ':

)(& " & ' ' % ! $ "" ! %
' % $ % '"% $ % (% B %;
' %

, % ('" & " & %

Input Extension Events

5 % ' ' + " '
' ' % ((, (DeviceKeyPress
DeviceKeyRelease DeviceButtonPress DeviceButtonRelease DeviceDevice-
MotionNotify DeviceProximityIn DeviceProximityOut % DeviceValuator
,) % + " % (%
' + , '" "" ! "" % % ! + '
% % % ! " C %

, % ' ' % (% (% (" extensions/in-
clude/XIproto.h 6 ' + % % ;; % (%
(" extensions/include/XI.h

& % (% ' + & (&
% E +5 % ' ' , % (& "

;; &' ! & E ! , ("" ! %
(

Device Key Events

DeviceKeyPresss "" (& % Key-
Press % " ("" ! %% " (& 4

H % % % ((% %

H % F (& % (% %

H '&F "' '& ("' %

H (F "' ("' %

H "' ' "' J "' ("'

) % (& ' + " ! E !
&' % ;; , ((DeviceKeyPress
% % (K(#; "' K(#

, ("" ! % (& ! ! ! ! '"% " C %4

```
extern int DeviceKeyPress;
DeviceIntPtr dev;
xEvent xE[2];
CARD8 id, num_valuators;
INT16 x, y, pointerx, pointery;
Time timestamp;
deviceKeyButtonPointer *xev = (deviceKeyButtonPointer *) xE;
deviceValuator *xv;

xev->type = DeviceKeyPress;                /* defined by input extension */
xev->detail = keycode;              /* key pressed on this device */
xev->time = timestamp;              /* same as for core events    */
xev->rootX = pointerx;              /* x location of core pointer */
xev->rootY = pointery;              /* y location of core pointer */

/*****************************************************************/
/*                                                               */
/* The following field does not exist for core input events.     */
/* It contains the device id for the device that generated the   */
/* event, and also indicates whether more than one 32-byte wire  */
/* event is being sent.                                          */
/*                                                               */
/*****************************************************************/

xev->deviceid = dev->id | MORE_EVENTS;        /* sending more than 1*/

/*****************************************************************/
/* Fields in the second 32-byte wire event:                      */
/*****************************************************************/
```

```
xv = (deviceValuator *) ++xev;
xv->type = DeviceValuator;          /* event type of second event */
xv->deviceid = dev->id;             /* id of this device          */
xv->num_valuators = 0;              /* no valuators being sent    */
xv->device_state  = 0;              /* will be filled in by DIX   */
```

Device Button Events

```
DeviceButton        ""   ( &          %      '
    % "    & %%   " ( &                        DeviceKey
```

Device Motion Events

```
DeviceMotion        ""    ( &          %      &
      % "  %%   " "'    ( &    9 "   ! !
 D'  %          ( &    , ( "" !   % (  &    !  !
! !      '"%     " C %4
```

```
extern int DeviceMotionNotify;
DeviceIntPtr dev;
xEvent xE[2];
CARD8 id, num_valuators;
INT16 x, y, pointerx, pointery;
Time timestamp;
deviceKeyButtonPointer *xev = (deviceKeyButtonPointer *) xE;
deviceValuator *xv;

xev->type = DeviceMotionNotify;     /* defined by input extension */
xev->detail = keycode;              /* key pressed on this device */
xev->time = timestamp;              /* same as for core events    */
xev->rootX = pointerx;              /* x location of core pointer */
xev->rootY = pointery;              /* y location of core pointer */

/*****************************************************************/
/*                                                               */
/* The following field does not exist for core input events.     */
/* It contains the device id for the device that generated the   */
/* event, and also indicates whether more than one 32-byte wire  */
/* event is being sent.                                          */
/*                                                               */
/*****************************************************************/

xev->deviceid = dev->id | MORE_EVENTS;        /* sending more than 1*/

/*****************************************************************/
/* Fields in the second 32-byte wire event:                      */
/*****************************************************************/

xv = (deviceValuator *) ++xev;
xv->type = DeviceValuator;          /* event type of second event */
xv->deviceid = dev->id;             /* id of this device          */
```

```
    xv->num_valuators = 2;              /* 2 valuators being sent     */
    xv->first_valuator = 0;             /* first valuator being sent  */
    xv->device_state  = 0;              /* will be filled in by DIX   */
    xv->valuator0 = x;                  /* first axis of this device  */
    xv->valuator1 = y;                  /* second axis of this device */
```

```
<   + +       %    %    '''      )(   %
&    L +   %%   "    (;   @   = (  %;    '''
 '''%     !    (  F '''  ( "%      "
```

Device Proximity Events

```
 &   ' %          '''      "  ( &    '
 "   %  '     &       + &                          ProximityIn
   % !        %   " $    '''          (   '
  ' M (    &     "    + &  !    '(   (    ' %
ProximityOut          % !     '''  (   "      + &
(    ' %    '(

Proximity        "&     &  ( &     '      ,
      ProximityIn   ProximityOut  %      %  " ( &
```