PERQ Systems Corporation

ACCENT LISP

SYSTEM INTERFACE

September 21, 1984

This manual is for use with Lisp Version M2, Accent Release S5.

Accent Lisp System Interface

September 21, 1984

Table of Contents — Page

1. Introduction

This document contains information on the interaction of Accent Lisp and the various subsystems of the Accent operating system. Chapters 2 through 11 contain Lisp calls for system interface routines. Chapter 12 contains specific Lisp programming examples demonstrating how to use some of the facilities of the operating system.

For complete information about the operating system and its subsystems, refer to the Accent Programming Manual. In the Programming Manual, the *Theory of Operations* document gives information about the operation of the Accent kernel, overviews of the subsystems, and Pascal programming examples. The other documents in the Programming Manual describe the subsystems in detail.

2. Kernel Interface Routines

This chapter gives the Lisp calls for Kernel interface routines. Information on the Kernel and descriptions of the routines listed may be found in the *Kernel Interface* document in the Accent Programming Manual.

2.1. IPC-Related Non-Primitive Routines

Descriptions of the following routines are in section 2.3 of the *Kernel Interface* document in the Accent Programming Manual.

```
AllocatePort
(Remote_Port BackLog)

    Returns Values :
(GR NewPort)


SetBackLog
(Remote_Port BackLogPort BackLog)

    Returns Values :
(GR)


DeallocatePort
(Remote_Port OldPort Reason)

    Returns Values :
```

```
(GR)


GetPortIndexStatus
(Remote_Port PortIndex)

    Returns Values :
(GR Backlog NWaitingMsgs EWaitingMsgs PortRight PortType)


GetPortStatus
(Remote_Port PortRight)

    Returns Values :
(GR Backlog NWaitingMsgs EWaitingMsgs PortIndex PortType)
```

2.2. Process Management Routines

Descriptions of the following routines are in section 3.2 of the *Kernel Interface* document in the Accent Programming Manual.

```
Fork
(Remote_Port HisKernelPort HisDataPort Ports Port_Count)

    Returns Values :
(GR HisKernelPort HisDataPort Ports Port_Count)


CreateProcess
(Remote_Port)

    Returns Values :
```

```
(GR HisKernelPort HisDataPort)


Terminate
(Remote_Port Reason)

    Returns Values :
(GR)


SetDebugPort
(Remote_Port DebugPort)

    Returns Values :
(GR)


Status
(Remote_Port)

    Returns Values :
(GR NStats)


SetPriority
(Remote_Port Priority)

    Returns Values :
(GR)


SetLimit
(Remote_Port ReplyPort Limit)
```

Returns Values :

(GR)

Suspend

(Remote_Port)

Returns Values :

(GR)

Resume

(Remote_Port)

Returns Values :

(GR)

Examine

(Remote_Port RegOrStack Index)

Returns Values :

(GR Value)

Deposit

(Remote_Port RegOrStack Index Value)

Returns Values :

(GR)

SoftInterrupt

(Remote_Port NormOrEmerg EnOrDisable)

```
    Returns Values :
(GR EnOrDisable)
```

2.3. Virtual Memory Management Routines

Descriptions of the following routines are in section 4.2 of the *Kernel Interface* document in the Accent Programming Manual.

```
CreateSegment
(Remote_Port ImagSegPort SegmentKind InitialSize MaxSize Stable)

    Returns Values :
(GR Segment)


TruncateSegment
(Remote_Port Segment NewSize)

    Returns Values :
(GR)


DestroySegment
(Remote_Port Segment)

    Returns Values :
(GR)


ReadSegment
(Remote_Port Segment Offset NumPages)
```

Returns Values :

(GR Data Byte_Count)

WriteSegment

(Remote_Port Segment Offset Data Byte_Count)

Returns Values :

(GR)

InterceptSegmentCalls

(Remote_Port)

Returns Values :

(GR OldPermSegPort NewPermSegPort)

SetPagingSegment

(Remote_Port Segment)

Returns Values :

(GR)

AvailableVM

(Remote_Port)

Returns Values :

(GR NumBytes)

ValidateMemory

```
(Remote_Port Address NumBytes CreateMask)

    Returns Values :
(GR Address)


InvalidateMemory
(Remote_Port Address NumBytes)

    Returns Values :
(GR)


SetProtection
(Remote_Port Address NumBytes Protection)

    Returns Values :
(GR)


ReadProcessMemory
(Remote_Port Address NumBytes)

    Returns Values :
(GR Data Byte_Count)


WriteProcessMemory
(Remote_Port Address NumBytes Data Byte_Count)

    Returns Values :
(GR)
```

```
Touch
(Remote_Port Address)

    Returns Values :
(GR)
```

2.4. Disk Management Routines

Descriptions of the following routines are found in section 5.2 of the *Kernel Interface* document in the Accent Programming Manual.

```
GetDiskPartitions
(Remote_Port DevNum)

    Returns Values :
(GR DevName PartL PartL_Cnt)


PartMount
(Remote_Port PartName ExUse)

    Returns Values :
(GR RootId PartKind PartPort PartS PartE)


PartDisMount
(Remote_Port)

    Returns Values :
(GR)
```

```
DirectIO
(Remote_Port CmdBlk DataHdr Data)

    Returns Values :
(GR CmdBlk DataHdr Data)
```

2.5. Display Management Routines

Descriptions of the following routines are found in section 6.2 of the *Kernel Interface* document in the Accent Programming Manual.

```
CreateRectangle
(Remote_Port RectPort BaseAddr ScanWidth
 BaseX BaseY MaxX MaxY IsFont)

    Returns Values :
(GR)


DestroyRectangle
(Remote_Port RectPort)

    Returns Values :
(GR)


EnableRectangles
(Remote_Port RectList Enable)

    Returns Values :
(GR)
```

SetKernelWindow

(Remote_Port LeftX TopY Width Height Inverted)

Returns Values :

(GR)

GetRectangleParms

(Remote_Port RectPort)

Returns Values :

(GR BaseAddr ScanWidth BaseX BaseY MaxX MaxY IsFont)

3. File System Interface Routines

This chapter gives the Lisp calls for File System interface routines. Information on the file system and descriptions of the routines listed may be found in the *File System* document in the Accent Programming Manual.

3.1. File I/O Routines

Descriptions of the following routines are in section 4.1 of the *File System* document in the Accent Programming Manual.

```
SubReadFile
(Remote_Port APathName)

    Returns Values :
(GR Data Byte_Count)


SesReadFile
(Remote_Port APathName)

    Returns Values :
(GR APathName Data Byte_Count DataFormat CreationDate NameStatus)


SubWriteFile
(Remote_Port APathName Data Byte_Count DataFormat)

    Returns Values :
```

```
(GR APathName CreationDate)
```

3.2. File Header Manipulation Routines

Descriptions of the following routines are found in section 4.2 of the *File System* document in the Accent Programming Manual.

```
SesGetFileHeader
(Remote_Port APathName)

   Returns Values :
(GR FileHeader)


SesReadBoth
(Remote_Port APathName)

   Returns Values :
(GR APathName Data Byte_Count FileHeader NameStatus)
```

3.3. Name Server Routines

Descriptions of the following routines are found in section 4.3 of the *File System* document in the Accent Programming Manual.

```
SubLookUpName
(Remote_Port APathName)

   Returns Values :
(GR APathName EntryType EntryData NameStatus)
```

SubTestName

(Remote_Port APathName)

Returns Values :

(GR APathName EntryType NameStatus)

SubEnterName

(Remote_Port APathName EntryType EntryData)

Returns Values :

(GR APathName)

SubDeleteName

(Remote_Port APathName)

Returns Values :

(GR)

SubReName

(Remote_Port OldAPathName NewAPathName)

Returns Values :

(GR NewAPathName)

SesScanNames

(Remote_Port WildAPathName NameFlags EntryType)

Returns Values :

(GR DirectoryName EntryList Entry_Count)

4. Process Manager Interface Routines

This chapter gives the Lisp calls for Process Manager interface routines. Information on the Process Manager may be found in the *Process Manager* document in the Accent Programming Manual. Descriptions of the following routines are in section 12.2 of the *Process Manager* document.

```
ProcMgr_Version
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)


PMRegisterProcess
(Remote_Port HisKPort HisDPort ProgName
 HisWindow HisTypescript EMConn Parent)

   Returns Values :
(GR)


PMSetSignal
(Remote_Port ProcPort Signal Action)

   Returns Values :
(GR)
```

PMSetSignalPort

(Remote_Port ProcPort SignalPort)

Returns Values :

(GR)

PMSetDebugPort

(Remote_Port ProcPort DebugPort DebugSignalOnly)

Returns Values :

(GR)

PMSaveLoadTime

(Remote_Port ProcPort LoadTime)

Returns Values :

(GR)

PMGetWaitID

(Remote_Port ProcPort)

Returns Values :

(GR WaitID)

PMGetTimes

(Remote_Port ProcPort)

Returns Values :

(GR LoadTime RunTime ElapsedTime)

PMGetProcPorts

(Remote_Port ProcPort)

Returns Values :

(GR hisWindow hisTypescript hisEMConn)

PMTerminate

(Remote_Port ProcPort Reason)

Returns Values :

(GR)

PMDebugProcess

(Remote_Port ProcPort Reason)

Returns Values :

(GR)

PMAddCtlWindow

(Remote_Port CtlWindow NewCtlWindow)

Returns Values :

(GR)

PMRemoveCtlWindow

(Remote_Port CtlWindow)

Returns Values :

(GR)

PMChangeGroup

(Remote_Port ProcPort NewWindow)

Returns Values :

(GR)

PMGroupSignal

(Remote_Port CtlWindow Signal)

PMProcessSignal

(Remote_Port ProcPort Signal)

PMSuspend

(Remote_Port ProcID)

Returns Values :

(GR)

PMResume

(Remote_Port ProcID)

Returns Values :

(GR)

PMDebug

(Remote_Port ProcID)

Returns Values :

(GR)

PMKill

(Remote_Port ProcID)

Returns Values :

(GR)

PMSetPriority

(Remote_Port ProcID priority)

Returns Values :

(GR)

PMBroadcast

(Remote_Port s)

Returns Values :

(GR)

PMGetStatus

(Remote_Port ProcID)

Returns Values :

(GR Stats Stats_Cnt)

5. Window Manager Interface Routines

This chapter gives the Lisp calls for Window Manager interface routines. Information on the Window Manager and descriptions of the routines listed here may be found in the *Procedural Guide to the Window Manager* document in the Accent Programming Manual.

5.1. Version Number Routine

This routine is described in section 11.2 of the *Procedural Guide to the Window Manager* document in the Accent Programming Manual.

```
Sapph_Version
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)
```

5.2. Window and Viewport Routines

Descriptions of the following routines are found in section 11.3 of the *Procecural Guide to the Window Manager* document in the Accent Programming Manual.

```
CreateWindow
(Remote_Port fixedPosition leftx topy fixedSize width
 height hasTitle hasborder title progName hasIcon)
```

Returns Values :

(R_e_s_u_l_t leftx topy width height progName vp)

DeleteWindow

(Remote_Port)

ModifyWindow

(Remote_Port newleftx newtopy newouterwidth newouterheight newRank)

Returns Values :

NIL

RemoveWindow

(Remote_Port)

RestoreWindow

(Remote_Port)

IdentifyWindow

(Remote_Port)

MakeViewport

(Remote_Port x y w h rank memory courteous transparent)

Returns Values :

(R_e_s_u_l_t)

DestroyViewport

```
(Remote_Port)
```

GetVPRank

```
(Remote_Port)

    Returns Values :
(R_e_s_u_l_t)
```

ViewportState

```
(Remote_Port)

    Returns Values :
(curlx curty curwidth curheight curRank
 memory courteous transparent)
```

ModifyVP

```
(Remote_Port newlx newty newwidth newheight newrank wantVpChEx)

    Returns Values :
NIL
```

GetFullViewport

```
(Remote_Port)

    Returns Values :
(R_e_s_u_l_t)
```

ReserveScreen

(Remote_Port reserve)

Returns Values :

NIL

GetScreenParameters

(Remote_Port)

Returns Values :

(width height)

SetWindowTitle

(Remote_Port title)

GetFullWindow

(Remote_Port)

Returns Values :

(R_e_s_u_l_t)

SetWindowName

(Remote_Port progName)

Returns Values :

(progName)

FullWindowState

(Remote_Port)

```
    Returns Values :
(leftx topy outerwidth outerHeigh rank hasBorder hasTitle isListener
name title)


SetWindowProgress
(Remote_Port nestLevel value max)


GetWinNames
(Remote_Port)

    Returns Values :
(names numNames curListenIndex)


WinForName
(Remote_Port name)

    Returns Values :
(R_e_s_u_l_t)


WindowViewport
(Remote_Port)

    Returns Values :
(vp vpWidth vpHeight)


DefineFullSize
(Remote_Port exceptW)
```

```
ExpandWindow
(Remote_Port)


ShrinkWindow
(Remote_Port)


GetWinProcess
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)


WinForViewPort
(Remote_Port vp)

   Returns Values :
(R_e_s_u_l_t isouter)
```

5.3. Icon Routines

Descriptions of the following routines are in section 11.4 of the {Procedural Guide to the Window Manager} in the Accent Programming Manual.

```
SetWindowError
(Remote_Port error)


SetWindowRequest
```

```
(Remote_Port requesting)


SetWindowAttention

(Remote_Port attn)


CompactIcons

(Remote_Port)


IconAutoUpdate

(Remote_Port allowed)


GetIconViewport

(Remote_Port)

   Returns Values :

(iconvp width height)


DeAllocIconVP

(Remote_Port)


GetIconWindow

(Remote_Port)

   Returns Values :

(R_e_s_u_l_t)
```

5.4. Graphics Routines

Descriptions of the following routines are in section 11.5 of the *Procedural Interface to the Window Manager* in the Accent Programming Manual.

```
ViewROP
(Remote_Port funct dx dy width height srcVP sx sy)


ViewColorRect
(Remote_Port funct x y width height)


ViewScroll
(Remote_Port x y width height Xamt Yamt)


ViewLine
(Remote_Port funct x1 y1 x2 y2)


ViewString
(Remote_Port fontVP funct dx dy str firstCh lastch)

    Returns Values :
(dx dy lastch)


ViewChArray
(Remote_Port fontVP funct dx dy chars arSize firstCh lastch)

    Returns Values :
(dx dy lastch)
```

ViewChar

(Remote_Port fontVP funct dx dy ch)

Returns Values :

(dx dy)

ViewPutString

(Remote_Port fontVP funct dx dy str firstCh lastch)

ViewPutChArray

(Remote_Port fontVP funct dx dy chars arSize firstCh lastch)

ViewPutChar

(Remote_Port fontVP funct dx dy ch)

VPtoScreenCoords

(Remote_Port x y)

Returns Values :

(scrX scrY)

ScreenToVPCoords

(vp scrX scrY)

Returns Values :

(x y)

LoadFont

(Remote_Port fileName)

Returns Values :

(R_e_s_u_l_t)

FontSize

(Remote_Port)

Returns Values :

(name PointSize Rotation FaceCode maxWidth maxHeight xOrigin yOrigin
fixedWidth
fixedHeight)

FontCharWidthVector

(Remote_Port ch)

Returns Values :

(dx dy)

GetSysFont

(Remote_Port)

Returns Values :

(R_e_s_u_l_t)

FontStringWidthVector

(Remote_Port str firstCh lastch)

Returns Values :

(dx dy)

LoadVPPicture

(Remote_Port fileName width height)

Returns Values :

(R_e_s_u_l_t)

PutViewportBit

(Remote_Port x y value)

GetViewportBit

(Remote_Port x y)

Returns Values :

(R_e_s_u_l_t value)

PutViewportRectangle

(Remote_Port Funct x y width height Data arSize WordsAcross ux uy)

Returns Values :

NIL

GetViewportRectangle

(Remote_Port x y width height ux uy)

Returns Values :

(R_e_s_u_l_t Data arSize WordsAcross)

5.5. Emergency Message Routine

This routine is described in section 11.6 of the {Procedural Guide to the Window Manager} in the Accent Programming Manual.

```
EnableNotifyExceptions
(Remote_Port notifyPort changed exposed)
```

5.6. Cursor, Region, and Tracking Routines

Descriptions of the following routines are in section 11.7 of the *Procedural Interface to the Window Manager* in the Accent Programming Manual.

```
LoadVPCursors
(Remote_Port fileName)

    Returns Values :
(R_e_s_u_l_t numCursors)


DestroyVPCursors
(cursors)

    Returns Values :
NIL


ReserveCursor
(vp reserve)
```

Returns Values :

NIL

SetCursorPos

(vp x y)

SetRegionCursor

(Remote_Port regionNum cursorImage cursIndex cursFunc track)

Returns Values :

NIL

GetRegionCursor

(Remote_Port regionNum)

Returns Values :

(cursorImage cursIndex cursFunc track)

SetRegionParms

(Remote_Port regionNum absolute speed minx maxx miny maxy modx posx mody posy)

Returns Values :

NIL

GetRegionParms

(Remote_Port regionNum)

Returns Values :

```
(absolute speed minx maxx miny maxy modx posx mody posy)
```

```
PushRegion
(Remote_Port regionNum leftx topy width height)
```

```
ModifyRegion
(Remote_Port regionNum leftx topy width height)

    Returns Values :
NIL
```

```
DeleteRegion
(Remote_Port regionNum)

    Returns Values :
NIL
```

```
DestroyRegions
(Remote_Port)
```

5.7. Listener Routines

Descriptions of the following routines are in section 11.8 of the *Procedural Guide to the Window Manager* document in the Accent Programming Manual.

```
EnableWinListener
(Remote_Port abortPort keytranTab timeOutarg)
```

Returns Values :

NIL

SetListener

(Remote_Port)

Returns Values :

NIL

MakeWinListener

(Remote_Port)

Returns Values :

NIL

GetListenerWindow

(Remote_Port)

Returns Values :

(R_e_s_u_l_t)

EnableInput

(Remote_Port keytrantab timeoutarg)

Returns Values :

NIL

5.8. Keyboard and Mouse Routines

Descriptions of the following routines are in section 11.9 of the *Procedural Guide to the Window Manager* in the Accent Programming Manual.

```
GetEvent
(Remote_Port howWait)

   Returns Values :
(R_e_s_u_l_t)


FlushEvents
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)
```

6. Environment Manager Interface Routines

This chapter gives the Lisp calls for Environment Manager interface routines. Information on the Environment Manager may be found in the *Environment Manager* document in the Accent Programming Manual. The routines listed here are described in section 3.2 of the *Environment Manager* document.

```
GetEnvVariable
(Remote_Port Name SearchScope)

    Returns Values :
(Variable Variable_Cnt VarType ActualScope)


SetEnvVariable
(Remote_Port Name VarType VarScope Variable)

    Returns Values :
NIL


ResolveSearchList
(Remote_Port Name FirstOnly)

    Returns Values :
(Variable FirstDefined)
```

ScanEnvVariables

```
(Remote_Port SearchScope)

    Returns Values :
(EnvScanList EnvScanList_Cnt)
```

CopyEnvConnection

```
(Remote_Port OldConnection)

    Returns Values :
(NewConnection)
```

EnvDisconnect

```
(Remote_Port)

    Returns Values :
NIL
```

7. Network Server Interface Routines

This chapter gives the Lisp calls for Network Server interface routines. Information on the Network Server may be found in the *Network Server* document in the Accent Programming Manual. The routines listed here are described in chapter 4 of the *Network Server* document.

```
E10GetAdd
(Remote_Port)

    Returns Values :
(Addr)


E10SetFilter
(Remote_Port PacketPort Which)

    Returns Values :
(GR)


E10PortClear
(Remote_Port PacketPort)

    Returns Values :
(GR)


E10Send
```

```
(Remote_Port Buff NumBytes)

   Returns Values :
(GR)


Net_Version
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)
```

8. Name Server Interface Routines

This chapter gives the Lisp calls for Name Server interface routines. Information on the Name Server may be found in the *Name Server* document in the Accent Programming Manual. The routines listed here are described in sections 2.1-2.4 of the *Name Server* document.

```
CheckIn
(Remote_Port PortsName Signature PortsID)

    Returns Values :
(GR)


Lookup
(Remote_Port PortsName)

    Returns Values :
(GR PortsID)


CheckOut
(Remote_Port PortsName Signature)

    Returns Values :
(GR)


MsgPortStatus
(Remote_Port PortsID)
```

```
   Returns Values :
(GR GlobalPort Owner Receiver SrcID SeqNum NetWaiting
 NumQueued Blocked Locked RecvQueue DataOffset InSrcID InSeqNum)
```

9. Time Server Interface Routines

This chapter gives the Lisp calls for Time Server interface routines. Information on the Time Server may be found in the *Time Server* document in the Accent Programming Manual. The routines listed here are described in section 2.2 of the *Time Server* document.

```
SetDateTime
(Remote_Port ITime)

    Returns Values :
NIL


SetSystemZone
(Remote_Port TimeZone DSTWhenTimely)

    Returns Values :
NIL


GetDateTime
(Remote_Port)

    Returns Values :
(R_e_s_u_l_t)


GetUserTime
```

```
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)


GetStringTime
(Remote_Port TimeFormat)

   Returns Values :
(R_e_s_u_l_t)


T_IntToZone
(Remote_Port ITime TZone)

   Returns Values :
(R_e_s_u_l_t)


T_IntToUser
(Remote_Port ITime)

   Returns Values :
(R_e_s_u_l_t)


T_UserToInt
(Remote_Port UTime)

   Returns Values :
(R_e_s_u_l_t)
```

```
T_UserToString
(Remote_Port UTime TimeFormat)

   Returns Values :
(R_e_s_u_l_t)


T_IntToString
(Remote_Port ITime TimeFormat)

   Returns Values :
(R_e_s_u_l_t)


T_StringToUser
(Remote_Port STime Index)

   Returns Values :
(R_e_s_u_l_t Index WhatIFound)


T_StringToInt
(Remote_Port STime Index)

   Returns Values :
(R_e_s_u_l_t Index WhatIFound)


T_Never
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)
```

10. Typescript Manager Interface Routines

This chapter gives the Lisp calls for Typescript Manager interface routines. Information on the Typescript Manager may be found in the *Typescript Manager* document in the Accent Programming Manual. The routines listed here are described in section 3.2 of the *Typescript Manager* document.

```
STSOpen
(Remote_Port vp env)

   Returns Values :
(R_e_s_u_l_t)


STSOpenWindow
(Remote_Port w env)

   Returns Values :
(R_e_s_u_l_t)


STSFullOpen
(Remote_Port vp env fontName doWrap dispPages)

   Returns Values :
(R_e_s_u_l_t)


STSFullOpenWindow
```

```
(Remote_Port w env fontName doWrap dispPages)

   Returns Values :
(R_e_s_u_l_t)


STSGetChar
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)


STSGetString
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)


STSPutChar
(Remote_Port ch)


STSPutString
(Remote_Port s)


STSFlushInput
(Remote_Port)


STSFlushOutput
(Remote_Port)
```

Returns Values :

NIL

STSChangeEnv

(Remote_Port env)

STSGrabWindow

(Remote_Port kPort)

Returns Values :

(R_e_s_u_l_t)

11. IO System Interface Routines

This chapter gives the Lisp calls for IO System interface routines. Information on the IO System may be found in the *IO System* document in the Accent Programming Manual. The routines listed here are described in chapter 3 of the *IO System* document.

```
IO-Init
(user-port)


IO_Version
(Remote_Port)

   Returns Values :
(R_e_s_u_l_t)


OpenIO
(Remote_Port UserPort)

   Returns Values :
(IOPort)


CloseIO
(Remote_Port)

   Returns Values :
(GR)
```

SyncIO

(Remote_Port Command CmdBlk CmdBlk_Cnt DataBuf DataBuf DataBuf_Cnt DataTransferCnt TimeOut_Arg)

Returns Values :

(Status)

12. Programming Examples

This chapter contains several Lisp programming examples. The programs demonstrate how to use some of the facilities of the Accent operating system.

For a detailed overview of the Accent operating system and its subsystems, refer to the *Theory of Operations* document in the Accent Programming Manual. Pascal programs that are equivalent to those presented in this chapter are in chapter 5 of the *Theory of Operations* document.

12.1. Graphics

This section provides a simple graphics program. The program draws lines and performs rasterops.

```
;;; -*- Mode: Lisp; Package: User -*-
;;;
;;; This shows some simple uses of the window manager.
;;; It does all of its operations in the window
;;; from which it was started.  We ignore emergency
;;; messages.

(in-package 'user)

(use-package 'accintdefs)             ; GR values
(use-package 'sapphdefsdefs)          ; Enumerated types for
Sapphire.
(use-package 'viewptdefs)             ; These four get you the
```

```
window calls
(use-package 'viewptuser)              ;  and various associated
information.
(use-package 'sapphdefs)
(use-package 'sapphuser)

(defvar max-x 0 "Maximum x coordinate")
(defvar max-y 0 "Maximum y coordinate")
(defvar vp nil "Inner viewport of the user window")

(defun get-view-port-info ()
  "Get the info neede to draw in the current window.  Userwindow is
the window
  that we are currently running in.  From this we can get the
information about
  the inner viewport for the window (the viewport itself, the max x
coordinate,
  and max y coordinate) which are returned as values in that order."
  (multiple-value-setq (vp max-x max-y ) (windowviewport
*userwindow*)))

(defun clear ()
  "This clears the inner viewport of the userwindow, once
get-view-port-info
  has been called."
  (vpcolorrect vp :rectwhite 0 0 max-x max-y))

(defun wait-for-carriage-return ()
  "Set the window attention flag (!) so the
   user knows we are waiting for a line
   of input, then wait for a carriage return."
  (setwindowattention *userwindow* t)
  (read-line)
  (setwindowattention *userwindow* nil)
  nil)

(defun draw-nested-rectangles ()
  "Draw a set of equally spaced rectangles one inside the other."
  (do ((start-x 0 (+ start-x 4))
       (start-y 0 (+ start-y 4))
       (end-x max-x (- end-x 4))
       (end-y max-y (- end-y 4)))
      ((or (>= start-x end-x) (>= start-y end-y)))
    (vpline vp :drawline start-x start-y end-x start-y)
    (vpline vp :drawline start-x end-y end-x end-y)
    (vpline vp :drawline start-x start-y start-x end-y)
    (vpline vp :drawline end-x start-y end-x end-y)))

(defun raster-op-nested-rectangles ()
  (do ((start-x 0 (+ start-x 4))
       (start-y 0 (+ start-y 4))
       (end-x max-x (- end-x 4))
       (end-y max-y (- end-y 4)))
      ((or (>= start-x end-x) (>= start-y end-y)))
    (vprop vp :rnot start-x start-y (- end-x start-x) (- end-y
start-y) vp
        start-x start-y)))

(defun graphics1 ()
  (get-view-port-info)
  (clear)
  (format t "This is a simple graphics test program.
           It runs a set of tests. When the attention
           flag in the icon for this window is set,
           type <cr> to go on to the next display, or leave the
           program after the last display.")
  (wait-for-carriage-return)
  (clear)
```

```
(draw-nested-rectangles)
(wait-for-carriage-return)
(clear)
(raster-op-nested-rectangles)
(wait-for-carriage-return))
```

12.2. File System

The following is a program that will read a file and treat the contents of that file as integers.

```
;;; -*- Mode: Lisp; Package: User -*-
;;;
;;; This program opens a file and prints out the first nintegers
integers in a
;;; file, or tells why it can't.

(in-package 'user)

(defconstant nintegers 100 "How many integers to read")

(defvar eof-value (cons nil nil)
  "This is a unique consed object that we can use to be sure that we
are at
  the end of file.")

(defun file1 (file)
  "This program opens a file of integers
   (its argument) and prints out the first
   nintegers integers in a file, or tells
   why it can't."
  (with-open-file (fp file :direction :input)
    (do ((object)
         (i 0 (1+ i)))
        ((= i nintegers))
      ;; What we read may be anything: an integer,
      ;; some other object, or the end
      ;; of file, so examine it.
      (setq object (read fp nil eof-value))
      (cond ((integerp object) (format t "~S~%" object))
            ((eq object eof-value)
             (format t
               "Encountered end of file after reading only ~S
objects. ~%" i)
             (return))
            (t
             (format t "Object ~S is not an INTEGER, it is of type
~S. ~%"
                     object (type-of object))))))
```

12.3. Memory

This section contains an example program that performs memory allocation and deallocation. It makes use of two system routines, ValidateMemory and InValidateMemory.

```
;;; -*- Mode: Lisp; Package: User -*-
;;;
;;; This is a simple example of memory allocation and deallocation.
;;;

(in-package 'user)

(use-package 'accintdefs)               ; GR values
```

```
(use-package 'accintuser)              ; (in)validatememory


;;; Use %primitive for system hacking.
(import 'lisp::%primitive *package*)

;;; This routine is used to 0 the type bits in a
;;; lisp object, so that an integer memory address
;;; can be turned into an absolute memory address
;;; as expected by the 8bit-system-ref and 8bit-system-set
;;; instructions.
(import 'lisp::%sp-make-misc *package*)

(defconstant memory-size 1024 "Amount of memory (in bytes) we
validate.")

(defun memory1 ()
  (multiple-value-bind (gr addr) (validatememory kernelport 0
memory-size -1)
    (if (not (eql gr success))
        (error "Could not validate memory, GR was ~S" gr))
    ;; Change addr from a fixnum to an absolute pointer.
    (setq addr (%sp-make-misc addr))
    ;; Go through memory and set each byte to the low bits of its
index into
    ;; the valid area by using Lisp system-hacking instructions.
    (dotimes (i memory-size) (%primitive 8bit-system-set addr i i))
    ;; Print out the contents of the memory.
    (dotimes (i memory-size) (format t " ~S " (%primitive
8bit-system-ref addr i)))
    (setq gr (invalidatememory kernelport addr memory-size))
    (if (not (eql gr success))
        (error "Could not invalidate memory, GR was ~S" gr))))
```