A Sample of an Extension to the Monitors Control Panel

The following program shows code that defines an extension to the Monitors control panel. The next program after that shows the resources for this extension in Rez format.

In response to the startup message, the Monitors extension shown in the first program checks the value of the item parameter to determine whether the user is a superuser. If the user is not a superuser, the Monitors extension uses the default values for the 'RECT' resource shown in the second program This rectangle ends just before the dividing line, so that the superuser controls are not displayed. If the user is a superuser, the **SetUpData** function in the first program extends the rectangle in the 'RECT' resource so that the rectangle includes all of the controls in the 'DITL' resource.

The code for the startup message allocates memory for the use of the Monitors extension and returns a handle to this memory as the function result. For all subsequent messages, the Monitors control panel passes the previous function result to the extension in the monitorValue parameter. In order to preserve the handle to this memory, the first program example sets the monitor function equal to the monitorValue parameter for all messages that the function does not process.

```
// Sample of an extension to the Monitors control panel
// Assuming inclusion of MacHeaders

// constants used in code
#define textItem    1    // StaticText item in cdev
#define lineItem    2    // separation line
#define downItem    3    // Down Arrow user item
#define upItem      4    // Up Arrow user item
#define countItem   6    // frame for count
#define brightItem      9    // radio button "filter"
#define lessItem    10   // radio button "aliasing"
#define slotCount   6    // a reasonable value
#define initMsg     1    // initialization
#define okMsg   2    // user clicked OK button
#define cancelMsg       3    // user clicked Cancel button
#define hitMsg      4    // user clicked control in Options dialog box
#define nulMsg      5    // periodic event
#define updateMsg       6    // update event
#define activateMsg     7    // not used
#define deactivateMsg       8    // not used
#define keyEvtMsg       9    // keyboard event
#define superMsg    10   // show superuser controls
#define normalMsg       11   // show only normal controls
#define startupMsg      12   // code has been loaded

// resource IDs
#define MemErrAlert 130     // alert to user: out of memory
#define deepAlert   131     // all other errors
#define dataRes     -4080// store data

// global typedefs
```

```
struct ScrnRecord {   // 'scrn' info for each screen
    short srDrvrHW; // spDrvrHW from Slot Manager
    short srSlot;   // slot # for the screen's video card
    long srDCtlDevBase;   // base address of card's memory
    short srMode; // sRsrcID for desired Mode
    short srFlagMask;// 0x77FE
    short srFlags; // active,main screen, B/W or color
    short srColorTable;    // resource ID of desired 'clut'
    short srGammaTable; // resource ID of desired 'gamma'
    Rect srRect;    // device's rectangle global coordinates
    short srCtlCount; // number of control calls
};
typedef struct ScrnRecord ScrnRecord;  // Record type for 'scrn' info
typedef ScrnRecord *ScrnRecordPtr;     // Pointer type for 'scrn' info
typedef ScrnRecordPtr *ScrnRecordHandle; // Handle type for 'scrn' info

struct ScrnRsrc { // complete 'scrn' resource
    short count;    // # of screens configured here
    ScrnRecord scrnRecs[slotCount];      // config for each one
};
typedef struct ScrnRsrc ScrnRsrc;   // record type for 'scrn' resource
typedef ScrnRsrc  *ScrnRsrcPtr;     // pointer type for 'scrn' resource
typedef ScrnRsrcPtr *ScrnRsrcHandle; // handle type for 'scrn' resource

struct MonitorData {  // local data for extension
    Boolean isSuperUser; // is the user a superuser?
    short filteringSetting;     // new filter setting
    short oldFiltering;     // previous filter setting
    short sleepTime;  // new sleep time
    short oldSleep;     // previous sleep time
};
typedef struct MonitorData MonitorData;     // record type for local data
typedef MonitorData *MonitorDataPtr;  // pointer type for local data
typedef MonitorDataPtr *MonitorDataHandle;    // handle type for local data

// some handy typedefs
typedef  RectPtr *RectHandle;     // handle to a rect structure
typedef  short *IntPtr;    // pointer to a short
typedef IntPtr *IntHandle;     // handle to a short

// Function prototypes
void     DrawMyRect   (WindowPtr, short);
OSErr    SetUpData  (short,MonitorDataHandle);
void     HandleHits (DialogPtr, short, short, MonitorDataHandle);
void     SaveNewValues     (MonitorDataHandle);
void     SetParamText (short);

// Extension entry point

//    ****************            Main           ******************

pascal long main(short message, short item, short numItems,
                        long monitorValue, DialogPtr mDialog, EventRecord theEvent,
                        short ScreenNum, ScrnRsrcHandle *screens, Boolean
                        *scrnChanged)
```

```
{
    short   itemType;
    Handle  dItem;
    Rect    box;
    MonitorDataHandle dataRecHand;
    OSErr   result;
    short   i;

    // set up handle to local data
    dataRecHand = (MonitorDataHandle) monitorValue;

    switch ( message ) {
    case startupMsg :  // time to check for superusers
        // first allocate memory to hold your local data
        dataRecHand = (MonitorDataHandle)NewHandle(sizeof(MonitorData));
        if ( dataRecHand ) {
            // initialize all fields
            result = SetUpData(item, dataRecHand);
            if ( result == noErr  ) {
                // this comes back in monitorValue
                return (long)dataRecHand;
            }
            else {
                return result;  // error should stop any further action
            }
        }
        else {   // display error message
            i = StopAlert(MemErrAlert, nil);
            return 255;
        }
        break;
    case initMsg :  // initialize cdev
        // set controls to their initial values and set the proc that
        // draws user items
        GetDItem(mDialog, numItems+lineItem,
                    &itemType, dItem,&box);
        if ( itemType == userItem )
            SetDItem(mDialog, numItems+lineItem, itemType,
                    &DrawMyRect, &box);
        GetDItem(mDialog, numItems+countItem, &itemType, dItem, &box);
        if ( itemType == userItem )
            SetDItem(mDialog, numItems+countItem, itemType,
                            &DrawMyRect, &box);
        SetParamText((**dataRecHand).sleepTime);
        if ( (**dataRecHand).isSuperUser  ) {
            if ( (**dataRecHand).oldFiltering == 0 )
                GetDItem(mDialog, numItems+lessItem, &itemType, dItem,
                            &box);
            if ( itemType == radCtrl + ctrlItem)
                SetCtlValue((ControlHandle)dItem,1);
            else
                GetDItem(mDialog, numItems+brightItem, &itemType, dItem,
                            &box);
            if ( itemType == radCtrl + ctrlItem )
                SetCtlValue((ControlHandle)dItem,1);
        }
```

```
            break;
        case hitMsg :
            HandleHits(mDialog, item, numItems, dataRecHand);
            break;
        case okMsg :    // user wants to implement changes
            // execute any hardware changes here
            SaveNewValues(dataRecHand);
            DisposeHandle(dataRecHand);  // release memory
            break;
        case cancelMsg:     // user does not want to save changes
            // make sure no changes are made permanent}
            DisposeHandle(dataRecHand);  // release memory
            break;
        }
        return monitorValue; // return handle to local data
    }   // MonExtend

    //    ****************         DrawMyRect        ******************

    // The following procedure is used both to frame the minutes-to-sleep box
    // and to draw the line separating the superuser controls from the
    // other controls.
    void  DrawMyRect(WindowPtr theWindow, short itemNo)
    {
        short itemType;
        Handle dItem;
        Rect box;

        GetDItem(theWindow, itemNo, &itemType, dItem, &box);
        FrameRect(&box);
    }

    //    ****************         SetUpData        ******************

    // The following procedure sets up the data storage for the monitor
    OSErr  SetUpData(short superUser,  MonitorDataHandle storage)
    {
        Handle filterType, sleepyTime;
        short i;
        OSErr result;
        Handle rHandle;

        result = noErr;
        HLock((Handle)storage);
        sleepyTime = GetResource('SLEP',dataRes);
        if ( sleepyTime ) {
            (**storage).oldSleep = **(IntHandle)sleepyTime;     // get old value
            (**storage).sleepTime = (**storage).oldSleep;
            ReleaseResource(sleepyTime);    // get rid of the resource
            if ( superUser ) {
                (**storage).isSuperUser = TRUE;
                filterType = GetResource('INTE',dataRes);
                if ( filterType ) {
                    // get old value
                    (**storage).oldFiltering = **(IntHandle)filterType;
                    (**storage).filteringSetting = (**storage).oldFiltering;
                    ReleaseResource(filterType);// get rid of the resource

                    // if the user is a superuser, change the rect to display more
                    // controls
                    rHandle = GetResource('RECT',-4096);
                    if ( rHandle )
                        (**(RectHandle)rHandle).top = -160;
                    else
                        result = 255;
                }
                else
                    result = 255;
            }
            else
                result = 255;

            // flag any memory errors
            if ( result == 255 ) {
                DisposHandle((Handle)storage);
                // tell the user there's a problem
                i = StopAlert(deepAlert, nil);
            }
        }
        HUnlock((Handle)storage);

        // nonzero result should stop any further action
        return result;
    }


    //    ****************      SaveNewValues     ******************

    // This procedure will save the current settings in resources
    void SaveNewValues(MonitorDataHandle dataRecHand)
    {
        Handle resHandle;

        if ( (**dataRecHand).sleepTime != (**dataRecHand).oldSleep ) {
            resHandle = GetResource('SLEP',dataRes);
            if ( resHandle ) {
                // set sleep time
                **(IntHandle)resHandle = (**dataRecHand).sleepTime;
                ChangedResource(resHandle);
                WriteResource(resHandle);
            }
        }
        // settings only for superusers
        if ( (**dataRecHand).isSuperUser )
            if ( (**dataRecHand).filteringSetting
                                    != (**dataRecHand).oldFiltering ) {
            resHandle = GetResource('INTE', dataRes);
            if ( resHandle ) {
                // set superuser controls
                **(IntHandle)resHandle = (**dataRecHand).filteringSetting;
                ChangedResource(resHandle);
                WriteResource(resHandle);
            }
        }
    }

    //    ****************          HandleHits          ******************

    // This procedure will mouse clicks on items in a dialogue
    void  HandleHits(DialogPtr mDialog, short whichItem, short numItems,
                MonitorDataHandle dataHand)
    {
        short itemType;
        Handle dItem;
        Rect box;

        HLock((Handle)dataHand);

        switch ( whichItem - numItems ) {
        case upItem :
            // get the text box
            GetDItem(mDialog,numItems+countItem,&itemType, dItem, &box);
            (**dataHand).sleepTime = ((**dataHand).sleepTime + 1) % 26;
            SetParamText((**dataHand).sleepTime);
            InvalRect(&box);
            break;
        case downItem:
            GetDItem(mDialog,numItems+countItem,&itemType, dItem,&box);
            (**dataHand).sleepTime = (**dataHand).sleepTime - 1;
            if ( (**dataHand).sleepTime < 0 )
                (**dataHand).sleepTime = 25;
            SetParamText((**dataHand).sleepTime);
            InvalRect(&box);
            break;
        case brightItem:
            GetDItem(mDialog,whichItem ,&itemType, dItem,&box);
            if ( itemType == radCtrl + ctrlItem )
                SetCtlValue((ControlHandle)dItem,1);
            GetDItem(mDialog,numItems+lessItem,&itemType, dItem,&box);
            if ( itemType == radCtrl + ctrlItem )
                SetCtlValue((ControlHandle)dItem,0);
            (**dataHand).filteringSetting = 1;
            break;
        case lessItem:
            GetDItem(mDialog,numItems+brightItem ,&itemType, dItem, &box);
            if ( itemType == radCtrl + ctrlItem )
                SetCtlValue((ControlHandle)dItem,0);
            GetDItem(mDialog, whichItem, &itemType, dItem, &box);
            if ( itemType == radCtrl + ctrlItem )
                SetCtlValue((ControlHandle)dItem,1);
            (**dataHand).filteringSetting = 0;
            break;
        }
        HUnlock((Handle)dataHand);
    }

    //    ****************          SetParamText          ******************
    // This procedure sets the text to change the static text items in
    // the dialogues
    void  SetParamText (short Sleep)
    {
        Str255 countStr;

        NumToString(Sleep, countStr);
        if  (Sleep < 10 ) {
            countStr[0] = countStr[0] + 1;
            countStr[2] = countStr[1];
            countStr[1] = '0';
        }
        ParamText(countStr,"\p","\p","\p");
    }
```

The following (i.e. the second example) shows, in Rez format, the resources that are used by the Monitors control panel extension shown in the first example, above.

```
// Resources that are used by the Monitors control panel extension

#include "Types.r"
#include "Pict.r"
#include "SysTypes.r"

type 'kcah' as 'STR ';

type 'card' as 'STR ';

type 'sysz' { unsigned hex longint; };

type 'RECT'
{
    rect;
};

// used to keep track of filter type
type 'INTE'
{
    integer;
};

// used to maintain setting of sleep interval
type 'SLEP'
{
    integer;
};

resource 'sysz' (0, purgeable) {
    $1000     // about 64 KB needed in system heap
};
```

```
resource 'vers' (1) {
    0x01, 0x00, release, 0x00,
    verUS,
    "1.00",
    "1.00, Copyright © 1990 Apple Computer, Inc."
};

resource 'kcah' (0, purgeable) {
    "Monitors Extension Sample"
};

resource 'BNDL' (128, purgeable) {
    'kcah',
    0,
    {
    'ICN#', {0, 128},
    'FREF', {0, 128}
    }
};

resource 'ICN#' (128, purgeable) {
    {   // array: 2 elements
        // [1]
        $"0000 0000 07FF FFE0 0800 0010 09FF FF90"
        $"0A00 0050 0A00 0050 0AF3 CE50 0A8A 2950"
        $"0A8A 2950 0AF3 CE50 0A82 4950 0A82 2950"
        $"0A82 2E50 0A00 0050 0A00 0050 0A00 0050"
        $"09FF FF90 0800 0010 0FFF FFF0 0800 0010"
        $"0800 0010 0800 0010 0800 7F10 0800 0010"
        $"0800 0010 0800 0010 0800 0010 07FF FFE0"
        $"0400 0020 0400 0020 0400 0020 07FF FFE0",
        //[2]
        $"0000 0000 07FF FFE0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 07FF FFE0"
        $"07FF FFE0 07FF FFE0 07FF FFE0 07FF FFE0"
    }
};

data 'ICON' (-4096, purgeable) {
    $"0000 0000 07FF FFE0 0800 0010 09FF FF90"
    $"0A00 0050 0A00 0050 0AF3 CE50 0A8A 2950"
    $"0A8A 2950 0AF3 CE50 0A82 4950 0A82 2950"
    $"0A82 2E50 0A00 0050 0A00 0050 0A00 0050"
    $"09FF FF90 0800 0010 0FFF FFF0 0800 0010"
    $"0800 0010 0800 0010 0800 7F10 0800 0010"
    $"0800 0010 0800 0010 0800 0010 07FF FFE0"
    $"0400 0020 0400 0020 0400 0020 07FF FFE0"
    $"0000 0000 07FF FFE0 0FFF FFF0 0FFF FFF0"
    $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
```

```
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 0FFF FFF0"
        $"0FFF FFF0 0FFF FFF0 0FFF FFF0 07FF FFE0"
        $"07FF FFE0 07FF FFE0 07FF FFE0 07FF FFE0"
    };

    resource 'DITL' (-4096, purgeable) {
        {   // array DITLarray: 10 elements
            // [1]
            {10, 161, 73, 301},
            StaticText {
                disabled,
                "Monitor Extension Example \n Screen saver"
                "\n© 1990 Apple Computer, Inc."
            },
            // [2]
            {80, 1, 81, 319},
            UserItem {
                enabled
            },
            // [3]
            {50, 79, 59, 88},
            UserItem {
                enabled
            },
            // [4]
            {39, 79, 49, 88},
            UserItem {
                enabled
            },
            // [5]
            {43, 58, 58, 78},
            StaticText {
                enabled,
                "^0"
            },
            // [6]
            {41, 55, 57, 75},
            UserItem {
                enabled
            },
            // [7]
            {89, 128, 106, 236},
            StaticText {
                enabled,
                "Filter Type"
            },
            // [8]
            {21, 25, 38, 162},
            StaticText {
                enabled,
                "Minutes before sleep"
            },
            // [9]
            {112, 102, 131, 249},
            RadioButton {
                enabled,
                "Zirconian Filtration"
            },
            // [10]
            {132, 102, 151, 244},
            RadioButton {
                enabled,
                "Anti-Aliasing Filter"
            },
            // [11]
            {40, 79, 58, 90},
            Picture {
                enabled,
                -4080
            }
        }
    };

    resource 'FREF' (128, purgeable) {
        'cdev',
        0,
        ""
    };

    resource 'RECT' (-4096, purgeable)
    {
        {-80,0,0,320}
    };

    // The 'card' resources ensure that this cdev is
    //   called when Options is pressed and the following
    //   cards are being used:
    //
    //   Macintosh Display Card
    //   Macintosh Display Card 8•24 GC


    resource 'card' (-4080, purgeable)
    {
        "Macintosh Display Card"
    };
    resource 'card' (-4079, purgeable)
    {
        "Macintosh Display Card 8•24 GC"
    };

    // The 'STR#' resource is used if for some reason you want to display
    //   a name for the card that is different from the one in the
    //   sResource of the board.

    resource 'STR#' (-4096, purgeable)
    {
        {   "Macintosh Display Card";
            "Macintosh Display Card 8•24";
```

```
		"Macintosh Display Card 8•24 GC";
		"Macintosh Display Card 8•24 GC 'The Accelerated'"};
};

	(Continued)
resource 'ALRT' (130, purgeable) {
	{50, 30, 190, 400},
	130,
	{	// array: 4 elements
		// [1]
		OK, visible, sound1,
		// [2]
		OK, visible, sound1,
		// [3]
		OK, visible, sound1,
		// [4]
		OK, visible, sound1
	}
};

resource 'ALRT' (131, purgeable) {
	{50, 30, 190, 400},
	131,
	{	// array: 4 elements
		// [1]
		OK, visible, sound1,
		// [2]
		OK, visible, sound1,
		// [3]
		OK, visible, sound1,
		// [4]
		OK, visible, sound1
	}
};

resource 'DITL' (130, purgeable) {
	{	// array DITLarray: 2 elements
		// [1]
		{90, 267, 110, 337},
		Button {
			enabled,
			"OK"
		},
		// [2]
		{10, 60, 70, 350},
		StaticText {
			disabled,
			"There is not enough memory"
		}
	}
};

resource 'DITL' (131, purgeable) {
	{	// array DITLarray: 2 elements
		// [1]
		{90, 267, 110, 337},
            Button {
                enabled,
                "OK"
            },
            // [2]
            {10, 60, 70, 350},
            StaticText {
                disabled,
                "An error occurred. \nI cannot display"
                " the options. "
            }
        }
};

// used to keep the setting of the filter type controls
//resource 'INTE' (-4080, purgeable) {
  5
};

// used to keep the setting of the time to sleep
//resource 'SLEP' (-4080, purgeable) {
  5
};

// PICT used to display the arrows to increase/decrease the sleep time
resource 'PICT' (-4080) {
    {134, 272, 152, 283},
    VersionOne {
        {   // array OpCodes: 2 elements
            // [1]
            clipRgn {
                {-30000, -30000, 30000, 30000},
                $""
            },
            // [2]
            bitsRect {
                2,
                {254, 352, 272, 368},
                {254, 352, 272, 363},
                {134, 272, 152, 283},
                srcCopy,
                $"3F80 4040 8420 8E20 9F20 BFA0 8E20 8E20"
                $"8020 8020 8E20 8E20 BFA0 9F20 8E20 8420"
                $"4040 3F80"
            }
        }
    }
};
```