#35 Writing external area procedures

Written by Todd Carper December 12, 1987
Modified February 1, 1988

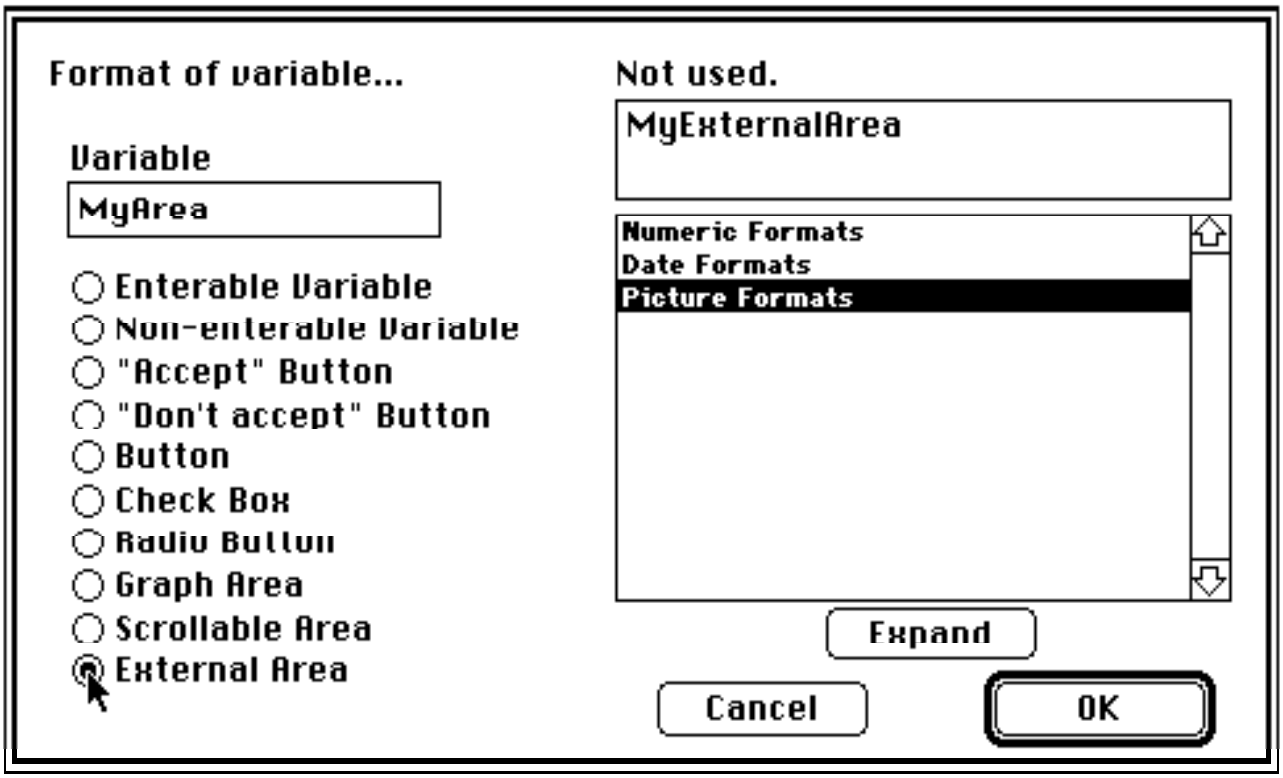
This document discusses writing external area procedures

This document discusses writing external area procedures. It is recommended that you read pages 237-248 of the Command Reference first. It is also recommended that you write an external procedure before attempting to write an external area procedure. All examples used in this document are written using MPW Pascal.

External areas are usually made of at least two procedures. The first procedure is the external area procedure, the second procedure is an external procedure that accesses information provided by the external area procedure.

When specifying the external area in your layout, follow these steps:
1) Be sure that the external area procedure has been installed with 4D External Mover.
2) Open the layout.
3) Draw a variable space the size you want to be used by the external area.
4) Give the variable a name which means something to you, ex. MyArea.
5) Click the External Area radio button.
6) In the box where formatting is usually supplied, (it is labeled as Not Used), type in the name of the external area procedure.
7) Close the format window.

The variable name you specified, (MyArea in this case), is of type LongInt and will contain the handle to the external area. Thus if you want to access information stored in the data structure defined in the external area, pass the variable name to another external procedure and then you can manipulate the information stored and used by the external area.



This example will create an external area that will be used as if it were an invisible button. In other words, in a layout we will specify the external area and then put a picture over the external area. When using the layout, we want to know if a user clicks on the picture. We can check for this because, unknown to the user, there is an external area procedure behind the picture monitoring all activity in that area.

The first procedure will be the external area procedure. The second procedure will be used to find out if the user has clicked the mouse in the external area.

Following is the program listing for the ***AreaButton*** external area procedure.

```
Program AreaButton;

Uses
        Memtypes;

Const
        InitEvt         = 16;
        DeInitEvt       = 17;
        CursorEvt       = 18;
        ScrollEvt       = 25;
        SelectReq       = 20;
        ActivNote       = 21;           {These two parameters are reserved for}
        DeActivNote     = 22;           {future compatibility}
```

Program listing for the AreaButton external area procedure

```
Type
        MyData =        record
                         Click:boolean;
                        end;

        MyDataPtr       = ^Mydata;
        MyDataHndl      = ^MyDataPtr;

{R-}
{D+}

Var
        MyEvent         :       EventRecord;
        MyRect :        AreaRect;
        MyName          :       str255;
        MyHndl :        MyDataHndl;

procedure thisisit(
        var AreaEvent   :       EventRecord;
        var AreaRect    :       Rect;
        var AreaName    :       str255;
        var AreaHndl    :       MyDataHndl);

begin

        case AreaEvent.what of

                InitEvt:        {enable the external area.  Do whatever setup is needed}
                begin
                        AreaHndl:=MyDataHndl(NewHandle(sizeof(MyData)));
                        AreaHndl^^.click:=False;
                end;

                DeInitEvt:      {disable the external area.  Cleanup after yourself!!}
                begin
                        DisposHandle(handle(AreaHndl));
                end;

                CursorEvt:      {we're in the area, do whatever needs be done}
                begin
                        SetCursor(GetCursor(crossCursor)^^);
                end;

                MouseDown:      {user has clicked the mouse in the area.  Do whatever}
                begin
                        AreaHndl^^.click:=True;         {set True so we can test for it later}
                end;
```

```
UpdateEvt:
begin
        {toolbox has set UpdateEvt true, redraw area if necessary}
end;
```

Program listing for the AreaButton external area procedure

```
                ScrollEvt:
                begin
                        {screen is being scrolled, call your scrolling sequence}
                end;

                SelectReq:
                begin
                        {
                        4D asks if you want to be selected for keyboard input.
                        If you want to be selected for keyboard input then

                        AreaEvent.message:=101;

                        now your area will receive keys entered from the keyboard.
                        Which means that you will also have to check for KeyDown in the above
                        case statement.
                        }
                end;

        end; {end case}
end; {end procedure thisisit}

{Main Block}
{Just used to compile properly}

Begin
        thisisit(MyEvent, MyRect, MyName, MyHndl);
end.
```

Following is the program listing for the external procedure ***GetButton***. This external procedure is to be used with the external area procedure ***AreaButton***.

```
Program GetButton;

Uses
        Memtypes;

Type
        MyData =        record
                         Click:boolean;
                        end;

        MyDataPtr       = ^Mydata;
        MyDataHndl      = ^MyDataPtr;

{$R-}
{D+}
```

```
Var
        MyHndl :        MyDataHndl;
        MyClick:        Integer;

procedure thisisit(
        var AreaHndl    :       MyDataHndl;     {this is the handle to the external area data
                                                structure}
        var Clicked     :       Integer);
```

Program listing for the GetButton external procedure

```
begin
        Clicked:=0;
        if(AreaHndl^^.Click) then
        begin
                AreaHndl^^.Click:=False;        {Set to False for the next check}
                Clicked:=1;                     {Set Clicked =1 to return to 4D}
        end;
end;


{Main Block}
{Just used to compile}

Begin
        thisisit(MyHndl, MyClick);
end.
```

Compile and install both procedures using 4D External Mover. When installing the external area procedure AreaButton, do not specify any parameters in the parameter list. When installing the external procedure GetButton, specify the parameters to be LongInt and Integer, respectively. LongInt is used because this is the parameter containing the handle to the data structure defined by the external area procedure.

In 4D, open a layout. Draw a variable area. Specify as an External Area. Give it the name MyButton. In the Not Used portion of the format dialog enter the external area procedure name, AreaButton. Close the format window.

To discover if the user has clicked in the area, have the following statement in the During phase of the layout procedure.

GetButton(MyButton;Clicked)

Now you can check the variable Clicked. If it equals 1 then the user clicked in the area, if it equals 0 then the user did not click in the area.

Drawing in an external area.

This next example will actually do something in the external area. When in the area we will allow the user to draw a filled box. This example will not take care of passing the picture back to 4D, nor will it perform a redraw when necessary, nor will it mask properly when you draw over an existing picture. These items are left to you to design for your specific application.

Following is the program listing for the external area procedure ***DrawArea***.

```
Program DrawArea;

Uses
        Memtypes, QuickDraw;
```

```
Const
        InitEvt         = 16;
        DeInitEvt       = 17;
        CursorEvt       = 18;
        ScrollEvt       = 25;
        SelectReq       = 20;
        ActivNote       = 21;           {These two parameters are reserved for}
```

Program listing for the DrawArea external area procedure

```
        DeActivNote     = 22;           {future compatibility}

Type

        MyData =        record
                                thePat:Pattern;  {The pattern to draw with and fill}
                                theMode:Integer;        {The transfer mode}
                                theSize:Point;          {width,height of pen}
                                theItem:Integer; {square, filled square}
                        end;

        MyDataPtr       = ^Mydata;
        MyDataHndl      = ^MyDataPtr;


{R-}
{D+}

Var
        MyEvent         :       EventRecord;
        MyRect :        Rect;
        MyName          :       str255;
        MyHndl :        MyDataHndl;

procedure thisisit(
        var AreaEvent   :       EventRecord;
        var AreaRect    :       Rect;
        var AreaName    :       str255;
        var AreaHndl    :       MyDataHndl);

Var
        LastPen, StartPen, EndPen       :       Point;

begin

        case AreaEvent.what of

                InitEvt:        {enable the external area.  Do whatever setup is needed}
                begin
                        AreaHndl:=MyDataHndl(NewHandle(sizeof(MyData)));
                        AreaHndl^^.theSize.h:=1;
                        AreaHndl^^.theSize.v:=1;
                        AreaHndl^^.theMode:=patCopy;
                        AreaHndl^^.thePat:=Black;
                        AreaHndl^^.theItem:=1;
                end;

                DeInitEvt:      {disable the external area.  Cleanup after yourself!!}
```

```
begin
        DisposHandle(handle(AreaHndl));
end;

CursorEvt:      {we're in the area, do whatever needs be done}
begin
        SetCursor(GetCursor(crossCursor)^^);    {change cursor to a cross}
end;
```

Program listing for the DrawArea external area procedure

```
MouseDown:    {user has clicked the mouse in the area.}
begin
        ShowPen;                {Show the pen}
        PenNormal;              {Set default draw settings}
        PenSize(AreaHndl^^.theSize.h,AreaHndl^^.theSize.v);    {Set pensize}
        PenMode(AreaHndl^^.theMode);          {Set the penmode}
        GetMouse(StartPen);                   {Get mouse location}
        GetMouse(LastPen);                    {Stores where mouse was}
        While(StillDown) do
        begin
                GetMouse(EndPen);             {Get where the mouse is now}

{This section of code is used to determine which way the mouse has traveled}
                If ((EndPen.h<>LastPen.h) or (EndPen.v<>LastPen.v)) then
                begin
                        if(StartPen.v<LastPen.v) then
                        begin
                                r.bottom:=LastPen.v;
                                r.top:=StartPen.v;
                        end
                        else
                        begin
                                r.bottom:=Startpen.v;
                                r.top:=LastPen.v;
                        end;

                        if(StartPen.h<LastPen.h) then
                        begin
                                r.left:=Startpen.h;
                                r.right:=LastPen.h;
                        end
                        else
                        begin
                                r.left:=LastPen.h;
                                r.right:=StartPen.h;
                        end;

                        r.topLeft.h:=r.left;
                        r.topLeft.v:=r.top;
                        r.botRight.h:=r.right;
                        r.botRight.v:=r.bottom;
                        PenPat(White); {to erase the existing square}

                        case AreaHndl^^.theItem of
                                2:      FrameRect(r);
                                3:      EraseRect(r);
                        end;
```

```
MoveTo(EndPen.h,EndPen.v);
PenNormal;
PenSize(AreaHndl^^.theSize.h,AreaHndl^^.theSize.v);
PenMode(AreaHndl^^.theMode);
if(StartPen.v<EndPen.v) then
begin
        r.bottom:=Endpen.v;
r.top:=StartPen.v;
```

Program listing for the DrawArea external area procedure

```
                                        end
                                        else
                                        begin
                                                r.bottom:=Startpen.v;
                                                r.top:=EndPen.v;
                                        end;
                                        if(StartPen.h<EndPen.h) then
                                        begin
                                                r.left:=Startpen.h;
                                                r.right:=EndPen.h;
                                        end
                                        else
                                        begin
                                                r.left:=EndPen.h;
                                                r.right:=StartPen.h;
                                        end;
                                        r.topLeft.h:=r.left;
                                        r.topLeft.v:=r.top;
                                        r.botRight.h:=r.right;
                                        r.botRight.v:=r.bottom;

                                        case AreaHndl^^.theItem of
                                                2:      FrameRect(r);
                                                3:      begin
                                                                FillRect(r,AreaHndl^^.ThePat);
                                                                FrameRect(r);
                                                        end;
                                        end;
                                        MoveTo(EndPen.h,EndPen.v);
                                        LastPen:=EndPen;
                                end;
                        end;
                end;

                UpdateEvt:
                begin
                        {toolbox has set UpdateEvt true, redraw area if necessary}
                end;

                ScrollEvt:
                begin
                        {screen is being scrolled, call your scrolling sequence}
                end;

                SelectReq:
                begin
                end;
        end; {end case}
```

```
end; {end procedure thisisit}

{Main Block}
{Just used to compile}
Begin
        thisisit(MyEvent,MyRect,MyName,MyHndl);
end.
```

Following is the program listing for the external procedure ***SetDrawType***. This external procedure is to be used with the external area procedure ***DrawArea***.

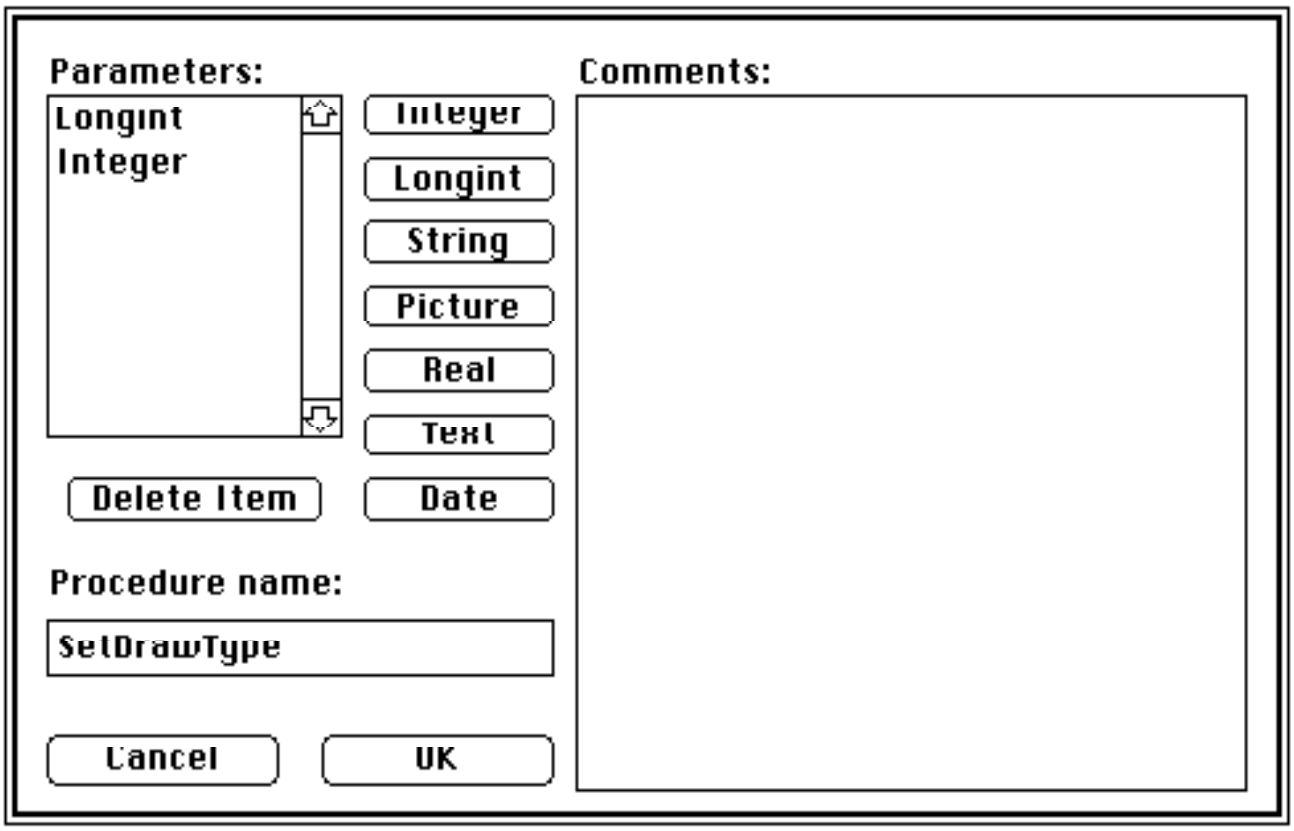
```
Program SetDrawType;

Uses
        Memtypes, QuickDraw;

Type

        MyData =        record
                                thePat:Pattern;
                                theMode:Integer;
                                theSize:Point; {width,height}
                                theItem:Integer; {square, filled square}
                        end;

        MyDataPtr       = ^Mydata;
        MyDataHndl      = ^MyDataPtr;

{R-}
{D+}

Var
        MyHndl :        MyDataHndl;
        MyType:         Integer;

procedure thisisit(
        var AreaHndl    :       MyDataHndl;
        var theType     :       Integer);

begin

        AreaHndl^^.theItem:=theType;
end;

{Main Block}
{Just used to compile}

Begin
        thisisit(MyHndl, MyType);
end.
```

Following is the program listing for the external procedure ***SetFillPat***. This external procedure is to be used with the external area procedure ***DrawArea***.

```
Program SetFillPat;

Uses
        Memtypes, QuickDraw;

Type

        MyData =        record
                                thePat:Pattern;
                                theMode:Integer;
                                theSize:Point; {width,height}
                                theItem:Integer; {square, filled square}
                        end;
        MyDataPtr       = ^Mydata;
        MyDataHndl      = ^MyDataPtr;

{R-}
{D+}

Var
        MyHndl :        MyDataHndl;
        MyPat           :       Integer;

procedure thisisit(
        var AreaHndl    :       MyDataHndl;
        var theFill     :       Integer);

begin
        GetIndPattern(AreaHndl^^.thePat, sysPatListID, theFill); {Use the standard pattern list}
end;

{Main Block}
{Just used to compile}

Begin
        thisisit(MyHndl, MyPat);
end.
```

Now compile and link all programs. When installing with 4D External Mover specify parameters as follows:

DrawArea	***SetDrawType***	***SetFillPat***
NONE	LongInt	LongInt
	Integer	Integer



In the layout, draw a variable field, specify as an External Area. Enter the external procedure name, 'DrawArea', in the Not Used portion of the dialog.

In the **Before** section of the layout procedure, call ***SetFillPat***(*numvar*) to set the initial fill pattern, (valid fillpatterns are 1-38. See Inside Mac I-474). Also call ***SetDrawType***(*numvar*) to set the initial drawing type, where valid draw types are 2, (regular box), and 3, (filled box).

In your layout you can also include some buttons which you can check for in the **During** phase to allow the user to change the pattern and the type of box being drawn. When appropriate, call the external procedures, ***SetDrawtype*** and ***SetFillPat***, with the desired values.