Rafe's TP Collection

Release 1.0

By Rafe Aldridge

Street Farm,
Dereham Road,
Garvestone,
Norfolk.
NR9 4QT
ENGLAND

E-Mail
until July 1994: rca @ ib.rl.ac.uk
then from September 1994 to July 1995: rcaldrid @ genvax.glam.ac.uk
(if this doesn't work check the username with postmaster @ genvax.glam.ac.uk)

Welcome, Disclaimer etc.

Thanks for taking the time to look at my collection of stuff for Turbo Pascal Versions 6 and 7. Hopefully, you will find it useful and will learn something from it. Please feel free to write to me with any ideas or comments at the address shown at the front of this document.

This collection is distributed as shareware as it's taken me a long time to compile this collection and a large proportion of my student loan has gone to Borland in order for it to happen. So if you find any part of Rafe's TP Collection useful then please become a registered user by sending 10 Pounds Sterling (plus shipping which is free in the UK, £1 for Europe and £2 for the rest of the world) to the address given at the front of this document. In return you will receive the LATEST FULL source code to ALL the units as well anything new. When you do register please try and use the registration form on the next page. This is also duplicated in REGISTER.FRM so you can either edit it in your favourite word processor or print out a copy with `COPY REGISTER.FRM PRN`. Apologies for the copyright message which will appear if you use the units but I want to encourage you to register (registered units don't have it).

Please feel free to make copies of this collection, pass it onto friends, upload it to BBSs, FTP it across the globe, whatever, so long as it stays complete. NOTE to vendors: Please drop me a note when adding the collection to your library. I would like to hear from you.

Finally, just to cover myself, I must say: THIS SOFTWARE AND MANUAL ARE DISTRIBUTED "AS IS" AND WITHOUT ANY WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. BECAUSE OF THE VARIOUS HARDWARE AND SOFTWARE ENVIRONMENTS THAT THIS SOFTWARE CAN BE PUT, NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. GOOD DATA PROCESSING PROCEDURE DICTATES THAT ANY PROGRAM BE THOROUGHLY TESTED WITH NON-CRITICAL DATA BEFORE RELYING ON IT. THE USER MUST ASSUME THE ENTIRE RISK OF USING THE UNITS OR SAMPLE CODE. ANY LIABILITY OF THE SELLER WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT OR REFUND OF THE PURCHASE PRICE.

Thanks again.

Rafe Aldridge.

Everything in Rafe's TP Collection is © 1993 by Rafe Aldridge.

Registration Form

Send to:
Rafe Aldridge,
Street Farm,
Dereham Road,
Garvestone,
Norfolk.
NR9 4QT
England

Please make cheques payable to Rafe Aldridge and write your card number on the reverse.

I would like to register Rafe's TP Collection Version 1.0. My details are as below and I enclose a cheque for £10 (pounds sterling) plus shipping (free if in UK, £1 for Europe and £2 for the rest of the world).

Name	
Company	
Address	
Town	
City	
Postcode	
Country	
Telephone (optional)	
E-Mail Address	
Disk Size Required	o 3.5" LD o 3.5" HD o 5.25" HD
I got Rafe's TP Collection from:	
Which is a:	o BBS o FTP Site o Mail Server o Shareware Vendor

Archive Contents

The following files will be found in the archive:

Filename	*Description*
BLANK.PAS	Example program that acts as a screen blanker.
BASEN.TP?	A unit for converting between various number bases
BASEN_D.PAS	Example program for the above
CLICK.PAS	Example program demonstrating the writing of simple TSRs
DESIGN.EXE	Program to aid the design of VGA characters
DISKSUM.PAS	Example program showing a recursive disk search
DJDUMP.PAS	Example procedure for dumping graphics to the HP DeskJet
GCLOCKS.TP?	A unit for displaying analogue / digital clocks in graphics modes (requires GRAPH)
GRAFIX.TP?	A unit for to supplement the graph unit (including faster plot routine)
JOKE.PAS	Example program demonstrating avoiding re-entrancy in TSRs
MATHS.TP?	A unit of maths functions not included as standard with TP/BP
MATRICES.TP?	A unit for manipulating small (5x5) matrices
MDA.TP?	A unit for controlling a MDA (monochrome display adaptor)
MOUSE.TP?	A unit for using a mouse in your programs
MOUSE_D.PAS	Example program for the above
NO_MD_RD.ASM	Assembler program demonstrating a simple TSR
SCREEN.TP?	A unit with some usefull text based routines
VGA.TP?	A unit to aid programming in 320x200x256 VGA graphics mode
VGA_D.PAS	Example program for the above
MANUAL.TXT	The manual in plain text format
MANUAL.PS	The manual typeset in PostScript format
READ.ME	Last minute additions and the ususal stuff
RAFETP.DIZ	Description for vendors
INSTALL6.BAT	A batch file to set up Rafe-TP for TP6
INSTALL7.BAT	A batch file to set up Rafe-TP for TP/BP7
REGISTER.FRM	Registration Form

The Programs

Blank.pas

This demonstrates the writing of simple TSRs in Turbo Pascal. It hooks both the keyboard and timer intterupts and acts as a screen saver.

Click.pas

This demonstrates the writing of very simple TSRs in Turbo Pascal. Basically it hooks the keyboard interrupt and makes a click every time a key is pressed. This program also demonstrates how to keep data in the code segment.

Design.exe

This is a simple program to help design characters for use with the *def_char* procedure from the *screen* unit.

Disksum.pas

This is an example of recursion in Turbo Pascal by means of a recursive disk search

DJDump.pas

Although not a complete program this example procedure shows how to dump graphics to the HP DeskJet. It also includes a small piece of assembler to read pixels of a graphics screen without the need of the Graph unit and BGI. NOTE: it was written for use with the 640x480x16 VGA graphics mode.

Joke.pas

Another example of simple TSRs that also demonstrates how to avoid re-entrancy.

No_md_rd.asm

Assembler program demonstrating a simple TSR. Well, seeing as TASM comes with Turbo Pascal Professional and Borland Pascal I thought I'd include this for the adventurous.

The Units

All the units were compiled with the switches {$D-,L-,R-} except for the *Maths* unit which was compiled with {$D-,E+,L-,N+,R-}. Hopefully these will be okay for most purposes otherwise register and you can compile them with whatever switches you desire.

Below is an alphabetical list of all the procedures found in the various units, supplied as part of the Rafe-TP collection, in the style of the Turbo Pascal manuals.

The following is a sample table.

Sample procedure - Unit it occupies

Function	What it does
Declaration	How it is declared in the unit
Result Type	If a function this gives the type of result
Remarks	General information
Restrictions	Restrictions or points to watch out for
Example	An example if applicable

Add_Matrix procedure - Matrices

Function	Adds two matrices.
Declaration	Add_Matrix (M1,M2 : Matrix; var M : Matrix);
Remarks	Returns in matrix M the sum of M1 and M2.

AFace procedure - GClocks

Function	Displays an analogue clockface.
Declaration	AFace (X,Y,L : Integer);
Remarks	Displays dots for an analogue clockface. The center of the clockface is X,Y and the radius of the clockface is L.
Restrictions	Must be in graphics mode and be using the Graph unit
Example	See AHands Procedure.

AHands procedure - GClocks

Function	Displays a analogue clockface.
Declaration	AHands (X,Y,L : Integer; H,M : Word);
Remarks	Displays the hands of an analogue clock. The time is given by H:M and the centre of the clock face is X,Y and the length of the minute hand is given by L.
Restrictions	Must be in graphics mode and be using the Graph unit
Example	

```
Uses GClock;
       Var H,M,S,SS,OH,OM : word;
Begin
       { set up graphics mode here }
       Repeat
          AFace(50,50,28);
          GetTime(H,M,S,SS);
          If M<>OM then
             Begin
                SetColor(Black);
                AHands(50,50,25,OH,OM);
                SetColor(White);
                AHands(50,50,25,H,M);
                OM:=M;
                OH:=H;
             End;
       Until False;
```

```
End;
```

Bin function - Basen

Function Converts a decimal value into a binary string.
Declaration Bin (Length : Byte; Dec_Val : Word; LZero : Boolean);
Result Type String
Remarks Converts Dec_Val into a string of Length representing the binary equivalent. If LZero is true then leading zeros are suppressed. Length is between 1 and 16.
Example

```
Uses Basen;
Begin
       Writeln (Bin(4,5,True));{ 101 }
       Writeln (Bin(4,5,False)); { 0101 }
End.
```

BinByte function - Basen

Function Converts a binary number into a byte.
Declaration BinByte (S : String);
Result Type Byte
Remarks Converts a binary number stored in string S into the decimal equivelent. Note that the length of S must be 8 characters - otherwise the value returned will be 0.
Example

```
Uses Basen;
Begin
       Writeln (BinByte ('10101010'));
End.
```

Blink_Disable - Mda

Function Disables blinking on the MDA Screen
Declaration Blink_Disable;

Blink_Enable - Mda

Function Enables blinking on the MDA Screen
Declaration Blink_Enable;

Box procedure - Screen

Function Draws a shadowed box on the text screen.
Declaration Box (X1,Y1,X2,Y2 : Byte; C : Char; T : String);
Remarks Draws a box in double lines using character C for the shadow. The top of the box can contain a title, T. X1,Y1 is the top left hand corner of the box while X2,Y2 is the bottom right corner.

Clear_Box procedure - Screen

Function Clears part of thescreen.
Declaration Clear_Box (C : Byte; var H : Hidden_Area);
Remarks Clears a 20x10 box in the centre of the screen coloured C. The previous contents are stored in H ready to be replaced by *Restore_Box*.
Example See *Restore_Box*.

CloseVGA procedure - VGA

Function Close the VGA graphics mode.
Declaration CloseVGA;
Remarks This closes the VGA graphics mode and returns to 80x25x16 text mode.

Cosh function - Maths

Function Returns the Hyp Cosine of the argument.
Declaration Cosh (X : Real);

Result Type Real
Remarks X is a real type expression. The result is Hyp Cosine of X.
Example

```
Uses Maths;
Begin
       Writeln (Cosh(2));  { 3.7621 }
       Writeln (Cosh(-4)); { 27.308 }
End.
```

Create_Matrix procedure - Matrices

Function Creates a matrix
Declaration Create_Matrix (R,C : Byte; var M : Matrix);
Remarks Creates an empty matrix, M, with R rows and C columns.

CursorOn procedure - Screen

Function Turns the text cursor on and sets the start and stop scan lines.
Declaration CursorOn (H,L : Byte);
Remarks Turns the text cursor on. Cursor starts at line L and finishes at line H. L and H are between 0 and 7 for 25 line mode and between 0 and 15 for 43 or 50 line mode. If L and H are both set greater than 15 then the default cursor is set.
Example

```
Uses Text;
Begin
       CursorOff;
       Writeln ('Return to Show Cursor');
       Readln;
       CursorOn (0,7);
       Writeln ('Return to Reset Cursor');
       Readln;
       CursorOn (16,16);
End.
```

CursorOff procedure - Screen

Function Turns the text Cursor off.
Declaration CursorOff;
Remarks Use CursorOn to retrieve cursor.
Example See CursorOn.

D2R function - Maths

Function Converts degrees to degrees radians.
Declaration D2R (X : Real);
Result Type Real
Remarks X is a real type expression representing an angle in degrees. The result represents the angle in degrees radians.
Example

```
Uses Maths;
Begin
       Writeln (D2R(90));{ 1.57 }
End.
```

Def_Char procedure - Screen

Function Redefines a VGA text character.
Declaration Def_Char (CT_Ofs, CT_Seg, N, C : Word);
Remarks A VGA character is 8x16 pixels and can therefore be described in a 16 byte array where each byte in binary represents which pixels will be lit on that line. Def_Char can be used to redefine a VGA character from such an array. CT_Seg:CT_Ofs points to the array (which can contain more than one set of 16 Bytes). N represents the number of characters to redefine and C is the starting character. The program DESIGN.EXE, supplied with this collection

can be used to define these characters.

Example

```
Uses Screen;
       Const New_Char : array [1..16] of byte =
          (1,2,4,8,16,32,64,128,255,128,64,32,16,8,4,2,1);
Begin
       Def_Char (Ofs (New_Char), Seg (New_Char, 1, 128);
       Write ('Character 128 has been redefined ');
       Write (#128);
       Writeln;
       Writeln ('Use the DOS command MODE CO80 to reset the
character set');
End.
```

Display_Matrix procedure - Matrices

Function Used to display a matrix.

Declaration Display_Matrix (X, Y : Byte; M : Matrix);

Remarks Outputs the matrix M to the screen at the position given by X,Y.

DClock procedure - GClocks

Function Displays a digital clock on the screen.

Declaration DClock (X,Y : Integer; H,M,S : Word);

Remarks Displays the time in the format HH:MM:SS in the current colour and character size. The current justification is also used. X,Y is the position of the clock.

Restrictions Must be in graphics mode and be using the Graph unit.

Example

```
Uses GClocks;
       Var H,M,S,SS,OH,OM,OS : word;
Begin
       { set up graphics mode here }
       Repeat
          Begin
            GetTime(H,M,S,SS);
            If S<>OS then
               Begin
                  SetColor(Black);
                  DClock(50,50,OH,OM,OS);
                  SetColor(White);
                  DClock(50,50,H,M,S);
                  OS:=S;
                  OM:=M;
                  OH:=H;
                End; {if}
          End; {repeat}
       Until False;
End;
```

DCos function - Maths

Function Returns the cosine of the argument.

Declaration DCos (X : Real);

Result Type Real

Remarks X is a real type expression. The result is the cosine of X. Where X is assumed to be an angle in degrees.

Example

```
Uses Maths;
Begin
       Writeln (DCos(90)); { 0 }
End;
```

DSin function - Maths

Function Returns the sine of the argument.

Declaration DSin (X : Real);
Result Type Real
Remarks X is a real type expression. The result is the sine of X. Where X is assumed to be an angle in degrees.
Example

```
Uses Maths;
Begin
       Writeln (DSin(90)); { 1 }
End.
```

DTan function - Maths

Function Returns the tangent of the argument.
Declaration DTan (X : Real);
Result Type Real
Remarks X is a real type expression. The result is the tangent of X. Where X is assumed to be an angle in degrees.
Example

```
Uses Maths;
Begin
       Writeln (DTan(180)); { 0 }
End.
```

Empty_Elements constant - Matrices

Function An empty matrix (see Matrix type)
Declaration const empty_elements : elements = (
(0,0,0,0,0),
(0,0,0,0,0),
(0,0,0,0,0),
(0,0,0,0,0),
(0,0,0,0,0));

Empty_Matrix procedure - Matrices

Function Empties a matrix.
Declaration Empty_Matrix (var M : Matrix);
Remarks Returns M empty.

ESetRes procedure - Grafix

Function Puts a value into the VGA ESetRes register.
Declaration ESetRes (P : Byte);
Remarks The byte P will be placed into the register.

Factorial function - Maths

Function Returns the factorial of the argument.
Declaration Factorial (X : Integer);
Result Type Real
Remarks X is an integer type expression. The result is the factorial of X otherwise known as X!.
Example

```
Uses Maths;
Begin
       Writeln (Factorial(5)); { 125 }
End.
```

FastPlot procedure - Grafix

Function Plots a pixel onto an EGA/VGA screen.
Declaration FastPlot (X,Y : Word; C : Byte);
Remarks Plots a pixel at X,Y in colour C. This routine contains no error checking to see whether the

screen is in graphics mode. It is at least three times faster than the Graph PutPixel procedure.

Restrictions Must be in graphics mode.

FlyBack function - VGA

Function Waits for a screen flyback.

Declaration FlyBack;

Remarks This waits for a screen flyback to occur.

GetButton function - Mouse

Function Returns status of the mouse buttons.

Declaration GetButton;

Result Type Byte

Remarks The returned byte contains the status of the mouse buttons. If a button is down then it's corresponding bit is set. Bit 0 represents the left button, bit 2 the middle button and bit 1 the right button.

Restrictions If the code between different calls to GetButton is small then the results may conflict. If this does occur then use a DELAY statement from the CRT unit to pad out your code. 250mS should be ample.

Example

```
Uses Mouse;
Begin
       if not MReset then Halt(0);
       Writeln ('Press left mouse button');
       repeat until GetButton=1;
       Writeln ('Press right mouse button');
       repeat until GetButton=2;
End.
```

GetButtonPress procedure - Mouse

Function Retrieve information about a button press.

Declaration GetButtonPress (Button : Byte; var BPN,X,Y : Word;);

Remarks Button is equal to 0 for the left button, 2 for the middle button and 1 for the right button. BPN is the number of times that Button has been pressed since the last time that this procedure was called for that Button. X and Y contain the coordinates of the last Button press.

Example

```
Uses Mouse;
       Var BPN,X,Y : Word;
Begin
       if not MReset then Halt(0);
       Writeln ('Press left button a few times and then press the
right button.');
       GetButtonPress (0,BPN,X,Y);
       repeat until GetButton=2;
       GetButtonPress (0,BPN,X,Y);
       Writeln ('You pressed the left button ',BPN,' times.');
End.
```

GetButtonRelease procedure - Mouse

Function Retrieve information about a button release.

Declaration GetButtonRelease (Button : Byte; var BRN,X,Y : Word;);

Remarks Button is equal to 0 for the left button, 2 for the middle button and 1 for the right button. BRN is the number of times that Button has been released since the last time that this procedure was called for that Button. X and Y contain the co-ordinates of the last Button release.

Example See GetButtonPress

GetMotion procedure - Mouse

Function Reads and returns the value of the mouse motion counters.

Declaration GetMotion (var X,Y : Integer);
Remarks The values X and Y are the net mouse displacement since the last call to this procedure. They are measured in mickeys, where 1 mickey equals 1/200". A positive number indicates movement to the right or downwards.
Example

```
Uses Mouse;
       Var X,Y : Integer;
Begin
       if not MReset then Halt(0);
       repeat
         GetMotion (X,Y);
         if X<0 then Writeln ('Right');
         if X>0 then Writeln ('Left');
         if Y<0 then Writeln ('Up');
         if Y>0 then Writeln ('Down');
        until GetButton>0;
End.
```

GetPos procedure - Mouse

Function Get current posistion of mouse.
Declaration GetPos (var X,Y : Word);
Remarks GetPos returns in X and Y the current coordinate of the mouse pointer, it is always measured in pixels.
Example

```
Uses Crt,Mouse;
       Var X,Y : Word;
Begin
       if not MReset then Halt(0);
       repeat
         ClrScr;
         GetPos (X,Y);
         Writeln ('X: ',X,'Y:',Y,'');
       until GetButton>0;
End.
```

Hex function - Basen

Function Converts a decimal value into a hexidecimal string.
Declaration Hex (Length : Byte; Dec_Val : Word; LZero : Boolean);
Result Type String
Remarks Converts Dec_Val into a string of Length representing the Hexidecimal equivalent. If LZero is True then leading zeros are suppressed.
Restrictions Length must be between 1 and 4.
Example

```
Uses Basen;
Begin
       Writeln (Hex(4,1234,True));{ 4D2 }
       Writeln (Hex(4,1234,False)); { 04D2 }
End.
```

HexArray constant - Basen

Function Constant array containing the Hexidecimal numbers.
Declaration HexArray : Array [0..15] of Char = ('0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F');

Hidden Area type - Screen

Function Used with *Restore_Box* and *Clear_Box*.
Declaration Type Hidden_Area = array [1..280] of Word;

HLine procedure - VGA

Function Draws a horizontal line in mode 13h.

Declaration HLine (L,X,Y : Word; Col : Byte);
Remarks Draws a horizontal line of length L starting at X,Y and in colour Col in mode 13h (320x200x256 VGA graphics mode).

Input_Matrix procedure - Matrices

Function Used to input a matrix.
Declaration Input_Matrix (X,Y : Byte; M : Matrix);
Remarks Allows the user to enter the elements of a matrix. The resultant matrix is stored in M and is input with the top left corner given by X,Y.
Example

```
Uses CRT, Matrices;
       Var M, NM : Matrix;
Begin
       ClrScr;
       Create_Matrix ( 3,3,M ); { create a 3x3 matrix }
       Input_Matrix (1,1,M); { get some elements }
       Add_Matrix ( M,M,NM ); { add matrix to itself }
       Display_Matrix (10,10,M); { display it }
End.
```

IsIt function - Mouse

Function Determines whether or not the pointer is in a given area.
Declaration IsIt (X1,Y1,X2,Y2 : Word);
Result Type Boolean
Remarks X1 and Y1 are the coordinates of the top left hand corner of a rectangular region of the screen. X2 and Y2 are the coords of the bottom right hand corner. IsIt returns true if the mouse pointer lies in this region.
Example

```
Uses Mouse;
Begin
       if not MReset then Halt(0);
       SetPointer (True);
       repeat
          if IsIt(0,0,640,8) then
            Writeln ('Mouse Pointer on line 1');
          end; { if }
        until GetButton>0;
End.
```

IsVGA function - VGA

Function Checks to see if the display adaptor is VGA.
Declaration IsVGA;
Result Type Boolean
Remarks This returns true if the display adaptor is a VGA compatible one.

KeyPress function - Screen

Function Checks to see if a keypress has occured.
Declaration KeyPress;
Result Type Boolean
Remarks This returns true if a key has been pressed. Unlike the CRT KeyPressed extended keys are also detected.

Matrix type - Matrices

Function Defines a type for use in Matrices
Declaration

```
Type    Elements = array [1..5,1..5] of integer;
Type    Matrix = record
                   Cols : Byte;
                   Rows : Byte;
```

```
                    Element   : Elements;
                End;
```

MDA_Clear - Mda

Function Clears the MDA Screen
Declaration MDA_Clear;
Remarks A sort of ClrScr for the MDA.

MDA constants - Mda

Function Defines constants for the various screen attributes. B stands for Bold.
Declaration

```
Const   Invisible = $0000;
        Underline = $0100;
        Normal = $0700;
        BUnderline = $0900;
        Bold = $0F00;
        Reverse = $7000;
        Blink = $8700;
        BBlink = $8F00;
        BReverse = $F000;
```

MDA_Style variable - Mda

Function Dictates the style used with *MDA_Write* and *MDA_Wrint*. It can be assigned any of the values given in MDA constants.
Declaration var MDA_Style : word;

MDA_Wrint procedure - Mda

Function Writes an integer on an MDA Screen
Declaration MDA_Wrint (X,Y,L : Byte; I : Integer);
Remarks Writes the integer, I, (including Bytes and Words) on the MDA screen at the position X,Y with length L. The style is given by the MDA_Style variable.
Example See MDA_Write.

MDA_Write procedure - Mda

Function Writes a string on an MDA Screen
Declaration MDA_Write (X,Y : Byte; S : String);
Remarks Writes the string, S, on the MDA screen at the position X,Y. The style is given by the MDA_Style variable.
Example

```
Uses Mda;
Begin
        MDA_Clear;
        MDA_Style:=Reverse; { use reverse video }
        MDA_Wrint (10,10,5,1000); { write 1000 on the MDA screen }
        MDA_Style=BUnderline; { use bold and underline }
        MDA_Write (20,20,'Bold and Underlined!');
        ReadLn;
End.
```

MReset function - Mouse

Function Resets mouse.
Declaration MReset;
Result Type Boolean
Remarks This should always be called at the start of a program that uses Mouse. It will return True if a valid mouse driver is found. This also does the following:

- The pointer is centralised and hidden.
- The Display page is set to 0 and the default shapes and styles for the pointer are

employed.

- Limits are set to be the entire screen.

Example

```
Uses Mouse;
Begin
       if not MReset then
         Writeln ('No Mouse Driver!');
       else
         Writeln ('Mouse Driver Found!);
End.
```

Multiply_Matrix procedure - Matrices

Function Multiplies two matrices.
Declaration Multiply_Matrix (M1, M2 : Matrix; var M : Matrix);
Remarks Returns in matrix M the result of M1*M2.

Oct function - Basen

Function Converts a decimal value into an octal string.
Declaration Oct (Length : Byte; Dec_Val : Word; LZero : Boolean);
Result Type String
Remarks Converts Dec_Val into a string of Length representing the Octal Equivalent. If LZero is True then leading zeros are suppressed.
Restrictions Length must be between 1 and 6.
Example

```
Uses Basen;
Begin
       Writeln (Oct(4,20,True)); { 24 }
       Writeln (Oct(4,20,False)); { 0024 }
End.
```

Offset procedure - Grafix

Function Puts a value into the VGA Offset register.
Declaration Offset (P : Byte);
Remarks The byte P will be placed into the register.

OpenVGA procedure - VGA

Function Switches to mode 13h
Declaration OpenVGA;
Remarks Makes mode 13h the current mode (320x200x256 VGA graphics mode). Check that the card is a VGA with *IsVGA* before using this procedure.

PelPan procedure - Grafix

Function Puts a value into the VGA PelPan register.
Declaration PelPan (P : Byte);
Remarks The byte P will be placed into the register.

Power function - Maths

Function Returns arg1 to the power of arg2.
Declaration Power (Arg1 : Integer; Arg2 : Real);
Result Type Real
Remarks Returns Arg1 to the power of Arg2 i.e. $Arg1^{Arg2}$.
Example

```
Uses Maths;
Begin
       Writeln ('5^3=',Power(5,3)); { 125 }
End;
```

PreRow procedure - Grafix

Function Puts a value into the VGA PreRow register.
Declaration PreRow (P : Byte);
Remarks The byte P will be placed into the register.

PutPixel procedure - VGA

Function Colours a pixel in mode 13h.
Declaration PutPixel (X,Y : Word; Col : Byte);
Remarks Sets the pixel at X,Y to colour Col in mode 13h (320x200x256 VGA graphics mode).

R2D function - Maths

Function Converts degrees radians to degrees.
Declaration R2D (X : Real);
Result Type Real
Remarks X is a real type expression representing an angle in degrees radians. The result represents the angle in degrees.
Example

```
Uses Maths;
Begin
       Writeln (R2D(1.57));{ 90 }
End.
```

Replicate procedure - Screen

Function Outputs multiple copies of a character.
Declaration Replicate (N : Word; C : Char);
Remarks Writes N copies of C starting at the current cursor location and in the current screen colours.

Restore_Box procedure - Screen

Function Replaces a hidden area.
Declaration Restore_Box (H : Hidden_Area);
Remarks H represents a 20x10 box removed from the screen. This procedure simply replaces it.
Example

```
Uses Screen;
       Var H : Hidden_Area;
Begin
       Clear_Box (10,H);
       Repeat Until Keypress;
       Restore_Box (H);
End.
```

RTan function - Maths

Function Returns the tangent of the argument.
Declaration RTan (X : Real);
Result Type Real
Remarks X is a real type expression. The result is the tangent of X. Where X is assumed to be an angle in degrees radians.
Example

```
Uses Maths;
Begin
       Writeln (RTan(PI)); { 0 }
End.
```

ScreenOff procedure - Grafix

Function Turns an EGA/VGA screen off.
Declaration ScreenOff ;
Remarks Turns the screen off. Does not have to be in graphics mode.

ScreenOn procedure - Grafix

Function Fucntion Turns an EGA/VGA screen on.
Declaration ScreenOn;
Remarks Turns the screen on. Does not have to be in graphics mode.

SearchString function - Screen

Function Checks to see if a substring exists in a string.
Declaration SearchString (S,SS : String);
Result Type Boolean
Remarks Returns True if SS exists in S. Lower and Upper case are different. Use UpString to convert both strings to uppercase if no distinction between case is required.
e.g. SearchString (UpString(s),UpString (ss));

SetActivePage - Grafix

Function Alters the active page.
Declaration SetActivePage (P : Byte);
Remarks Sets the active page to be P. No checking is done to see if this is a valid page or not.

SetGPointer procedure - Mouse

Function Defines the shape, colour and hotspot of the pointer in graphics modes.
Declaration SetGPointer (X,Y,OS,SE : Word);
Remarks X and Y are the coordinates of the hotspot relative to the top left corner of the pointer (0,0). SE:OS points to a pointer image buffer which is 64 bytes long. The first 32 bytes contain an ANDed bitmask whilst the second 32 bytes contain a XORed bitmask. The pointer is 16x16 pixels. The layout of the buffer is a 64 byte array. When arranged in a 32 row by 2 coloumn layout the first coloumn represents the right half of the pointer and the second coloumn represents the left half (I know it's backwards but I never wrote it!). The AND and XOR bits do the following.

AND bit	*XOR bit*	*resultant pixel colour*
0	0	background
0	1	XOR background
1	0	foreground (see thro')
1	1	XOR foreground

Example

```
{ A 16x16 hollow box would be defined as a
  mixture of the following:-
     $80 in binary  $01 in binary
     10000000       00000001
     11111111       11111111
     $FF in binary  $FF in binary
 The hotspot is in the top left corner.}
Const
       Box : Array [1..63] of Byte = (
          $FF,$FF,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
          $01,$80,
```

```
            $01,$80,
            $FF,$FF,
            $FF,$FF,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $01,$80,
            $FF,$FF);
Begin
       { set up graphics display here }
       If not MReset then Halt (0);
       SetPointer (True);
       SetGPointer (1,1,Ofs(Box),Seg(Box));
       repeat until GetButton>0;
End.
```

SetLimits procedure - Mouse

Function Limits the mouse pointer display area.

Declaration SetLimits (X1,Y1,X2,Y2 : Word);

Remarks Sets the limits for the mouse pointer by assigning an area where the pointer is restricted to. The mouse pointer will be moved (if necessary) to fit in the area. The area is a rectangle with X1,Y1 being the coordinate of the top left corner and X2,Y2 being the coordinate of the bottom right corner.

Example

```
Uses Mouse;
Begin
       if not MReset then Halt (0);
       SetPointer (True);
       Writeln ('Mouse restricted to top half of the screen.');
       SetLimits (0,100,632,100);
       repeat until GetButton>0;
End.
```

SetMode procedure - Grafix

Function Puts the screen into a different mode.

Declaration SetMode (P : Byte);

Remarks Sets the screen mode to that of P. Note that this procedure does not check to see if P is a valid mode.

SetNoLimits procedure - Mouse

Function Defines an exclusion area for the pointer.

Declaration SetNoLimits (X1,Y1,X2,Y2 : Word);

Remarks X1,Y1 are the coordinates of the top left corner of a rectangle. X2,Y2 are the coordinates of the bottom right hand corner. The mouse pointer will not be displayed whilst in this area. The exclusion area is replaced with another call to SetNoLimits or cancelled with a call to SetPointer (True) or MReset.

Example

```
Uses Mouse;
Begin
       if not MReset then Halt(0);
       SetPointer (True);
```

```
        Writeln ('Mouse pointer will not be shown in center of
screen');
        SetNoLimits (220,50,420,150);
        repeat until GetButton>0;
        SetPointer (True);
End.
```

SetPointer procedure - Mouse

Function Hides and shows mouse pointer.

Declaration SetPointer (Point : Boolean);

Remarks If point is True then the mouse pointer is displayed otherwise if point is False then the pointer is hidden.

Example

```
Uses Crt,Mouse;
Begin
        if not MReset then Halt(0);
        Writeln ('Press a button to show pointer');
        repeat until GetButton>0;
        Delay (250);{ see GetButton for this }
        SetPointer (True);
        Writeln ('Press a button to exit');
        repeat until GetButton>0;
End.
```

SetPointerPage procedure - Mouse

Function Selects the display page for the mouse pointer.

Declaration SetPointerPage (Page : Byte);

Remarks Page is the page number where the pointer will be displayed. Valid page numbers depend on the current display mode.

SetPos procedure - Mouse

Function Set posistion of mouse pointer.

Declaration SetPos (X,Y : Word);

Remarks Moves the mouse pointer to coordinate X,Y. The posistion is adjusted (if necessary) to lie within the boundry defined by SetLimits.

Example

```
Uses Mouse;
        Var X,Y : Word;
Begin
        if not MReset then Halt(0);
        SetPointer (True);
        Write ('Enter X coord: ');
        Readln (X);
        Writeln;
        Write ('Enter Y coord: ');
        Readln (Y);
        SetPos (X,Y);
End.
```

SetRes procedure - Grafix

Function Puts a value into the VGA SetRes register.

Declaration SetRes (P : Byte);

Remarks The byte P will be placed into the register.

SetTPointer procedure - Mouse

Function Defines the shape and attributes of the mouse pointer in text modes.

Declaration SetTPointer (ANDM, XORM : Word; SH : Boolean);

Remarks If SH is True then the hardware cursor is used.This cursor is defined by CursorOn or is the default cursor. If SH is False then ANDM is the AND mask value and XORM is the XOR

mask value. The following table shows how to define these masks.

Bits	Significance
0-7	Character Code
8-10	Foreground Colour
11	Intensity (or upper block of colours)
12-14	BackGround Colour
15	Blink (or upper block of colours)

The default values are ANDM = $77FF and XORM = $7700.

All the mouse routines work with pixels even in text mode where we would use character spaces. Conversion is easy as in 80x25 text mode the screen is 640x200 pixels. This obviously means that each character is 8x8 pixels. So to get from characters to pixels multiply by 8 and to get from pixels to characters divide by 8. That's not quite right because of the posistion of the hotspots in text mode. The character posistions start at 1,1 whilst pixels start at 0,0. So you have to take one away from your character locations before you multiply. Likewise when converting from pixels to characters you have to add one to each of the locations after division. As a simple example say we wanted to restrict the mouse to a box on the screen where the top left coord is 5,10 and the bottom right coord is 30,20. We simply take one from each of these and multiply by 8.

4* 8 = 32
9* 8 = 72
29 * 8 = 232
19 * 8 = 152

So our call to SetLimits will look like this:

SetLimit (32,72,232,152);

In 43x80 or 50x80 mode each character is only 8x8 but the screen dimensions are 640x344 and 640x400 respectivley.

Example

```
Uses Mouse;
Begin
       if not MReset then Halt(0);
       Writeln ('New Style Pointer!');
       SetTPointer ($7777,$FFFF,False);
       repeat until GetButton>0;
End.
```

Sinh function - Maths

Function Returns the Hyp sine of the argument.
Declaration Sinh (X : Real);
Result Type Real
Remarks X is a real type expression. The result is Hyp sine of X.
Example

```
Uses Maths;
Begin
       Writeln (Sinh(2));{ 3.6268 }
       Writeln (Sinh(-4)); { -27.29 }
End.
```

StartAddr procedure - Grafix

Function Puts a value into the VGA Start Address register.
Declaration StartAddr (P : Byte);

Remarks The byte P will be placed into the register.

Subtract_Matrix procedure - Matrices

Function Subtracts two matrices.
Declaration Subtract_Matrix (M1,M2 : Matrix; var M : Matrix);
Remarks Returns in matrix M the result of M1 - M2.

Tanh function - Maths

Function Returns the Hyp Tangent of the argument.
Declaration Tanh (X : Real);
Result Type Real
Remarks X is a real type expression. The result is Hyp tangent of X.
Example

```
Uses Maths;
Begin
       Writeln (Tanh(2));{ 0.964}
       Writeln (Tanh(-4)); { -0.999 }
End.
```

UpString function - Screen

Function Converts a string to uppercase.
Declaration UpString (S : String);
Result Type String
Remarks Returns S in uppercase. Non alphabetic characters are left alone.
Example

```
Uses Screen;
Begin
       Writeln (UpString ('rafe's tp collection'));
End.
```

VLine procedure - VGA

Function Draws a vertical line in mode 13h.
Declaration VLine (L,X,Y : Word; Col : Byte);
Remarks Draws a vertical line of length L starting at X,Y and in colour Col in mode 13h (320x200x256 VGA graphics mode).

VGAOff procedure - VGA

Function Turns off the VGA screen.
Declaration VGAOff;
Remarks Turns off the VGA screen.

VGAOn procedure - VGA

Function Turns on the VGA screen.
Declaration VGAOn;
Remarks Turns on a VGA screen having been switched off with VGAOff.

Wallpaper procedure - Screen

Function Wallpapers the screen.
Declaration Wallpaper (C : Char);
Remarks Fills the screen with character C in the current screen colours.

Wrte procedure - Screen

Function Faster version of WRITE.
Declaration Wrte (X,Y:Byte; Var S:String; F,G:Byte);
Remarks Writes a the string S directly into video RAM at the screen posistion denoted by X,Y. F is

the foreground color whilst B is the background color. NOTE: this procedure causes snow on a CGA adaptor.