VFD-STUDIO

Erhalt von Systeminformationen

Inhalt

Systeminformationen:

Mit den beschriebenen Methoden der PortIOVFD-Klasse ist es möglich beliebige Texte und Grafiken darzustellen.
Hauptsächlicher Verwendungszweck des VFD-Studios sollte jedoch nicht die Anzeige eigener Texte sei, sondern die Darstellung verschiedener Systemeigenschaften.

Das VFD-Studio kann eine Vielzahl an Systeminformationen ermitteln und anzeigen, die in drei Gruppen zusammenfassbar sind:

Informationen, die sich nicht ständig verändern:

- Rechner-Name im Netzwerk
- Benutzername
- Größe des Arbeitsspeichers
- Betriebsystem
- Zeitzone
- CPU-Taktrate
- CPU-Hersteller
- CPU-Bezeichnung
- CPU-Identifikation
- CPU-Klasse
- CPU-Model
- CPU-Familie
- CPU-Fähigkeiten:
  - MMX MultiMediaExtension
  - FPU FloatingPointUnit
  - ISSE2 Intel SSÉ2 Befehlssatz
  - PSN ProzessorSerialNumber
  - VME VirtualMode
  - ACC AutomaticClockControl
  - PAE PhysicalAdressExtension
  - MSR ModelSpecificRegister
  - ACPI AdvanceConfigurationPowerInterface
- Letzte DirectDraw-Anwendung
- Letzte Direct3D-Anwendung

Informationen, die sich ständig verändern können:

- Uhrzeit
- Datum
- Wochentag
- Uptime (Zeit seit Starten des Betriebsystemes)
- IP-Nummer im TCP/IP-Netzwerk
- Auflösung des aktuellen Videomodus
- Aktuelle Farbtiefe
- Bildwiederholfrequenz
- CPU-Auslastung
- Durchschnittliche CPU-Auslastung
- Freier Arbeitsspeicher
- Winamp Version
- Winamp Titel
- Winamp Länge des aktuellen Titels
- Winamp Abgespielte Zeit, bzw. aktuelle Position im Titel

Informationen, die grafisch angezeigt werden, bzw. in besonderer Form zusammengefasst:

- Uhrzeit (analog oder digital)
- Verlauf der CPU-Nutzung
- Verlauf der Speichernutzung
- Übersicht über Festplatten/Laufwerke
- Übersicht über installierte Drucker

Beteiligte Units und Klassen:

An der Beschaffung dieser Informationen sind folgende Klassen und Units beteiligt:

adCPUUsage.pas	-	ermittelt CPU-Auslastung
Drives.dll	-	ermittelt Größe von Festplatten/Laufwerken
TWinampControl	-	ermittelt alle Winamp-Informationen
Tacpuid	-	ermittelt CPU-Fähigkeiten
TSysInfo	-	ermittelt alle anderen Informationen

adCPUUsage, TWinampControl und Tacpuid sind nicht selbstprogrammiert sondern stammen aus fremder Quelle (siehe Datei Copyright.txt) und werden daher nur oberflächlich beschrieben.

adCPUUsage

Diese Unit von Alexey A. Dynnikov dient dem VFD-Studio der Ermittlung der aktuellen CPU-Auslastung.
Sie enthält im wesentlichen die drei Methoden

```
procedure CollectCPUData;
function  GetCPUUsage(Index: Integer): Double;
procedure ReleaseCPUData;
```

Um die CPU-Last zu ermitteln muss zuerst CollectCPUData aufgerufen werden. Danach kann die CPU-Auslastung mit GetCPUUsage ermittelt werden.
Das Ergebnis wird als Gleitkommazahl zwischen 0 und 1 übergeben. Um einen Prozentwert zu erhalten muss das Ergebnis also noch mit 100 multipliziert werden.
Die Prozedur ReleaseCPUDate schließlich stoppt die CPU-Last-Aufzeichnung und gibt gebrauchte Ressourcen wieder frei.

Drives.dll

Das VFD-Studio sollte auch die Größe der Laufwerke angeben.
Delphi kennt dazu die Windows-Methode GetDiskFreeSpaceEx.
Diese gibt die Größe in des Laufwerkes in Bytes an.
Bei großen Laufwerken (über 2GB Speicher) führte dies jedoch zu einem Problem, da das VFD-Studio mit Delphi3 geschrieben war und Delphi3 nur 32 Bit-Variablen kennt.
Bei einer gewöhnlichen Festplatte mit beispielsweise 60GB (= 64.424.509.440 Bytes) entspricht das Ergebnis bereits 36 Bit.
Es gibt mit Delphi3 wegen der Beschränkung auf 32 Bit keine Möglichkeit die Größe einer solchen Festplatte anzugeben, selbst wenn nur die Größe in MBytes interessiert.

Eine Möglichkeit wäre nun gewesen das ganze Projekt in eine neuere Delphi-Version zu übertragen, die 64 Bit-Variablen kennt. Erfahrungsgemäß produziert man dabei jedoch mehr Probleme, als man behebt.
Aber ein anderer Ausweg besteht darin die Byte-Laufwerksgröße von einem externen Programm zu ermitteln, das nicht auf 32 Bit beschränkt ist, und das Ergebnis in MByte als 32 Bit-Größe zurück an das VFD-Studio zu übergeben.

Genau dies ist die Aufgabe der Drives.dll.
Diese Library wurde mit Delphi7 geschrieben und legt die Bytegröße des Laufwerkes in einer 64 Bit-Variablen ab. Dann wird dieses Ergebnis noch durch 1.048.576 geteilt, um auf die MByte-Größe zu kommen, und als Funktionsergebnis zurückgegeben.

Es werden von dieser Library zwei Funktionen exportiert:
GetFreeDiskSpace und GetDiskSpace, die wie die Namen bereits verraten die Größe des Laufwerkes, bzw. den freien Speicherplatz ermitteln.
Damit das VFD-Studio diese Funktionen verwenden kann muss es sie importieren, was im Implementierungsteil der MainUnit erfolgt:

```
implementation

{$R *.DFM}


//Die Funktionen aus der Drives.dll einbinden:
function GetFreeDiskSpace(Drive: char): double; external 'Drives.dll';
function GetDiskSpace(Drive: char): double; external 'Drives.dll';
```

Der Quellcode der Library befindet sich in der Datei Drives.dpr.

TWinampControl

Diese Klasse kann das Programm Winamp fernsteuern und verschiedene Informationen über Winamp abfragen.
Die Instanz dieser Klasse im Hauptprogramm des VFD-Studio ist die Komponente „Winamp“.

TWinampControl arbeitet folgendermaßen:
Mit der Windows-Methode „FindWindow“ wird das Handle der Anwendung mit dem Klassennamen „Winamp v1.x“ gesucht.
An dieses Handle (funktioniert natürlich nur, wenn Winamp tatsächlich läuft) werden dann Botschaften (Messages) gesendet, die für den Datenaustausch zwischen Winamp und Komponente sorgen. Somit ist es möglich Steuerbefehle zum Fernsteuern von Winamp (wovon das VFD-Studio jedoch keinen Gebrauch macht) zu senden und Informationen, z.B. Name des aktuellen Liedes zu empfangen.

Der Quelltext dieser Klasse befindet sich in der Datei WinampControl.pas

Tacpuid

Diese Klasse ermittelt die Fähigkeiten der CPU.
Die Instanz dieser Klasse im VFD-Studio heißt „CPU“.

Diese Klasse von Dadang Sofyan ist sehr umfangreich und komplex. Die Funktionsweise der (größtenteils in Assembler) geschriebenen Methoden würde den Rahmen dieser Dokumentation bei weitem sprengen.
Daher kann hier nur erwähnt werden, dass die Features (Fähigkeiten) der CPU bereits zur Entwurfszeit ermittelt werden und in ein Property (FeatureList) der Komponente geschrieben werden.
Das Hauptprogramm (die MainUnit) liest aus dieser Property-Liste die Fähigkeiten heraus und gibt auf dem Display „ja“ aus, wenn das Feature unterstützt wird, bzw. „nein“ falls nicht.

Der Quelltext der Klasse Tacpuid befindet sich in der Datei ACPUID.pas

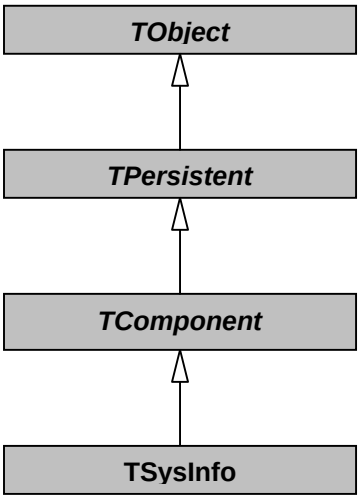
TSysInfo

Diese Klasse versammelt den größten Teil der Methoden zu Informationsbeschaffung, die teils mit fremder Unterstützung und teils selbst entwickelt wurden.

Der Sourcecode dieser Klasse befindet sich in dem Listing SysInfo.pas

Die Klasse TSysInfo ist wie TPortIOVFD abgeleitet von TComponent, welche wiederum von den Basisklassen TPersistent und TObject abgeleitet ist.

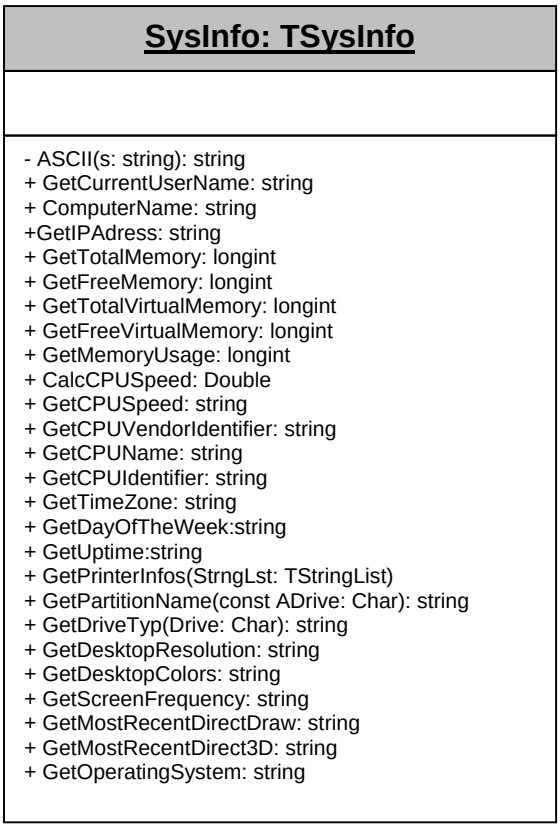
Klassenhierarchie:



TSysInfo wurde zwar als Komponente entworfen, hätte jedoch auch problemlos als DLL oder externe Unit entwickelt werden können.

Folgend nun die nähere Beschreibung dieser Klasse.

Übersicht - TSysInfo:



Vorwort zu TSysInfo:

Die Klasse TSysInfo enthält Methoden zur Ermittlung verschiedener Systeminformationen über Hardware, Speicher, Betriebssytem, usw.
Fast alle Methoden basieren auf bekannten Windowsmethoden und werden daher nur auf ihre Funktion hin beschrieben.
Für eine eingehendere Beschreibung der Windowsmethoden sei auf die „Win32 Developer Reference“ WIN32SDK.HLP von Delphi verwiesen.

Quellen:

Ein Teil der Methoden wurde mit Hilfe folgender Quellen erstellt oder von dort kopiert:

SwissDelphiCenter	www.swissdelphicenter.ch/de/index.php
Forum Delphi-Source	www.delphi-groups.de/YaBBSe/
Tory’s Delphi Pages	www.torry.net

Die Methoden der Klasse TSysInfo:

ASCII:

Dies ist die einzige nicht-public Funktion, die dazu dient String-Rückgabewerte der anderen Funktionen an den Displayzeichensatz anzupassen.

```
function TSysInfo.ASCII(s: string): string;  //ersetzt ö,ä,ü durch oe,ae,ue zwecks ASCII
Kompatiblität
var
   i: integer;
begin
     while pos('ö',s)<>0 do begin
      i:=pos('ö',s);
      delete(s,i,1);
      insert('oe',s,i);
     end;
     while pos('ä',s)<>0 do begin
      i:=pos('ä',s);
      delete(s,i,1);
      insert('ae',s,i);
     end;
     while pos('ü',s)<>0 do begin
      i:=pos('ü',s);
      delete(s,i,1);
      insert('ue',s,i);
     end;
     while pos('Ö',s)<>0 do begin
      i:=pos('Ö',s);
      delete(s,i,1);
      insert('Oe',s,i);
     end;
     while pos('Ä',s)<>0 do begin
      i:=pos('Ä',s);
      delete(s,i,1);
      insert('Ae',s,i);
     end;
     while pos('Ü',s)<>0 do begin
      i:=pos('Ü',s);
      delete(s,i,1);
      insert('Ue',s,i);
     end;
     result:=s;
end;
```

GetDesktopColors:

Diese Funktion ermittelt die gegenwärtige Farbtiefe und gibt das Ergebnis als String zurück. Beispiel: „24 Bit“

```
function TSysInfo.GetDesktopColors: string;
var
   DesktopDC: THandle;
begin
     DesktopDC:=GetDC(0);
     result:=inttostr(GetDeviceCaps(DesktopDC,BITSPIXEL))+ ' Bit';
     ReleaseDC(0,DesktopDC);
end;
```

Hinweis: die Funktion funktioniert nicht korrekt unter Windows 9x!

GetDesktopResolution:

Diese Funktion ermittelt die gegenwärtige Auflösung und gibt sie als String zurück. Beispiel: „1280x1024“.
Die Einheit ist in Pixeln angegeben.

```
function TSysInfo.GetDesktopResolution: string;
begin
     result:=inttostr(screen.width)+'x'+inttostr(screen.height);
end;
```

GetScreenFrequency:

Diese Funktion ermittelt die gegenwärtige Bildwiederholrate, d.h. die vertikale Frequenz der Bildschirmdarstellung.
Das Ergebnis wird als String zurückgegeben. Beispiel: „75 Hz“. Oder, falls die Bildwiederholrate nicht ermittelt werden konnte: „unbekannt“
Die Einheit ist in Hertz angegeben.

```
function TSysInfo.GetScreenFrequency: string; // nur ab WinNT!!!
var
   DesktopDC: THandle;
begin
     result:='unbekannt';
     DesktopDC:=GetDC(0);
     try
      result:=inttostr(GetDeviceCaps(DesktopDC,VREFRESH))+' Hz';
     except
     end;
     ReleaseDC(0,DesktopDC);
end;
```

Hinweis: Laut Win32 Referenz kann die Bildwiederholrate nur bei NT-basierten Windowsversionen ermittelt werden.

GetComputerName:

Diese Funktion ermittelt den Namen, den der PC im lokalen Netzwerk hat und gibt ihn als String zurück. Beispiel: „Arbeitsplatzrechner1“

```
function TSysInfo.ComputerName: string;
var
   size: DWord;
begin
     size:=Max_COMPUTERNAME_LENGTH+1;
     SetLength(Result, Size);
     if GetComputerName(PChar(Result), Size) then
        SetLength(Result, Size)
     else Result:='';
     result:=ASCII(result);
end;
```

GetCurrentUserName:

Diese Funktion ermittelt den Namen des gegenwärtig angemeldeten Benutzers und gibt ihn als String zurück. Beispiel: „Administrator“. Oder, falls unbekannt: „Unbekannter User“.

```
function TSysInfo.GetCurrentUserName: string;
var
   u: array[0..127] of Char;
   sz: DWord;
begin
     Result:='Unbekannter User';
     sz:=SizeOf(u);
     GetUserName(u,sz);
     Result:=u;
     result:=ASCII(result);
end;
```

GetIPAdress:

Diese Funktion ermittelt die aktuelle IP-Nummer des Rechners, falls er an ein Netzwerk angeschlossen ist und das TCP/IP-Protokoll verwendet.
Die IP-Nummer wird als String zurückgegeben. Beispiel: „127.0.0.1“

```
function TSysInfo.GetIPAdress: string;
var
   phoste: PHostEnt;
   Buffer: array[0..100] of Char;
   WSAData: TWSADATA;
begin
     result:='';
     if WSAStartup($0101, WSAData)<>0 then exit;
     GetHostName(Buffer, SizeOf(Buffer));
     phoste:=GetHostByName(buffer);
     if phoste = nil then result:='IP not found'
     else result:=StrPas(inet_ntoa(PInAddr(phoste^.h_addr_list^)^));
     WSACleanup;
end;
```

Hinweis zu Speicher-Funktionen:
Die Funktionen haben einen 32 bit Rückgabewert, d.h. Speicher der größer als 2GB ist wird nicht erfasst!

GetMemoryUsage:

Diese Funktion liefert die Größe des verwendeten Arbeitspeichers in Byte als 32 bit Longint zurück.

```
function TSysInfo.GetMemoryUsage: longint;
var
   memory: TMemoryStatus;
begin
     memory.dwLength:=SizeOf(memory);
     GlobalMemoryStatus(memory);
     result:=memory.dwMemoryLoad;
end;
```

GetTotalMemory:

Diese Funktion liefert die Größe des physikalischen Arbeitspeichers in Byte als 32 bit Longint zurück. (Der physikalische Arbeitsspeicher ist der Speicher der RAM-Chips).

```
function TSysInfo.GetTotalMemory: longint;
var
   memory: TMemoryStatus;
begin
     memory.dwLength:=SizeOf(memory);
     GlobalMemoryStatus(memory);
     result:=memory.dwTotalPhys;
end;
```

GetFreeVirtualMemory:

Diese Funktion liefert die Größe des freien virtuellen Arbeitspeichers in Byte als 32 bit Longint zurück. (Der virtuelle Arbeitsspeicher ist der auf der Festplatte ausgelagerte Speicher).

```
function TSysInfo.GetFreeVirtualMemory: longint;
var
   memory: TMemoryStatus;
begin
     memory.dwLength:=SizeOf(memory);
     GlobalMemoryStatus(memory);
     result:=memory.dwAvailVirtual;
end;
```

GetTotalVirtualMemory:

Diese Funktion liefert die Größe des virtuellen Arbeitspeichers in Byte als 32 bit Longint zurück.

```
function TSysInfo.GetTotalVirtualMemory: longint;
var
   memory: TMemoryStatus;
begin
     memory.dwLength:=SizeOf(memory);
     GlobalMemoryStatus(memory);
     result:=memory.dwTotalVirtual;
end;
```

GetFreeMemory:

Diese Funktion liefert die Größe des freien physikalischen Arbeitspeichers in Byte als 32 bit Longint zurück.

```
function TSysInfo.GetFreeMemory: longint;
var
   memory: TMemoryStatus;
begin
     memory.dwLength:=SizeOf(memory);
     GlobalMemoryStatus(memory);
     result:=memory.dwAvailPhys;
end;
```

CalcCPUSpeed:

Diese Funktion berechnet die Geschwindigkeit der CPU und gibt das Ergebnis als Fließkommazahl in MHz zurück. Funktioniert mit allen Windowsversionen.

```
function TSysInfo.CalcCPUSpeed: Double;
const
     TimeOfDelay=500;
var
   TimerHigh, TimerLow: DWord;
begin
     SetPriorityClass(GetCurrentProcess,REALTIME_PRIORITY_CLASS);
     SetThreadPriority(GetCurrentThread, THREAD_PRIORITY_TIME_CRITICAL);
     asm
     dw 310Fh
     mov TimerLow, eax
     mov TimerHigh, edx
     end;
     Sleep(TimeOfDelay);
     asm
     dw 310Fh
     sub eax, TimerLow
     sub edx, TimerHigh
     mov TimerLow, eax
     mov TimerHigh, edx
     end;
     SetPriorityClass(GetCurrentProcess,NORMAL_PRIORITY_CLASS);
     SetThreadPriority(GetCurrentThread, THREAD_PRIORITY_NORMAL);
     Result:=TimerLow/(1000.0*TimeOfDelay);
end;
```

GetCPUSpeed:

Diese Funktion ermittelt die CPU-Geschwindigkeit aus der Windows-Registry und gibt das Ergebnis als String zurück. Beispiel: „2800 MHz“
Achtung: Die CPU-Geschwindigkeit ist nicht bei allen Windowsversionen in der Registry abgespeichert!

```
function TSysInfo.GetCPUSpeed: string;
var
  Reg: TRegistry;
begin
  Result:='unbekannt';
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('Hardware\Description\System\CentralProcessor\0', False) then
    begin
      Result := IntToStr(Reg.ReadInteger('~MHz')) + ' MHz';
      Reg.CloseKey;
    end;
  except
  end;
  Reg.Free;
end;
```

GetCPUVendorIdentifier:

Diese Funktion ermittelt aus der Windows-Registry den Hersteller der CPU und gibt das Ergebnis als String zurück. Beispiel: „Genuine Intel“

```
function TSysInfo.GetCPUVendorIdentifier: string;
var
  Reg: TRegistry;
begin
  Result:='---';
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('Hardware\Description\System\CentralProcessor\0', False) then
    begin
      Result := Reg.ReadString('VendorIdentifier');
      Reg.CloseKey;
    end;
  except
  end;
  Reg.Free;
  result:=ASCII(result);
end;
```

GetCPUName:

Diese Funktion ermittelt aus der Windows-Registry den Namen der CPU und gibt das Ergebnis als String zurück. Beispiel: „Intel(R) Pentium(R) 4 CPU 2.80GHz“

```
function TsysInfo.GetCPUName: string;
var
  Reg: TRegistry;
  s: string;
begin
  Result:='---';
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('Hardware\Description\System\CentralProcessor\0', False) then
    begin
      s:=copy(Reg.ReadString('ProcessorNameString'),15,255);
      Result := s;Reg.ReadString('ProcessorNameString');
      Reg.CloseKey;
    end;
  except
  end;
  Reg.Free;
  result:=ASCII(result);
end;
```

GetCPUIdentifier:

Diese Funktion ermittelt aus der Windows-Registry die Identifikation der CPU und gibt das Ergebnis als String zurück. Beispiel: „x86 Family 15 Model 2 Stepping 9“

```
function TSysInfo.GetCPUIdentifier: string;
var
  Reg: TRegistry;
begin
  Result:='---';
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('Hardware\Description\System\CentralProcessor\0', False) then
    begin
      Result := Reg.ReadString('Identifier');
      Reg.CloseKey;
    end;
  except
  end;
  Reg.Free;
  result:=ASCII(result);
end;
```

GetPrinterInfos:

Diese Prozedur ermittelt aus der Windows-Registry die installierten Drucker und gibt das Ergebnis als Stringliste im Parameter ‚StrngLst' zurück.

```
procedure TSysInfo.GetPrinterInfos(StrngLst: TStringList);
var
   reg: TRegistry;       // Registry
   s: string;            // HilfsString
   Printers: TStrings;   // Liste mit PrinteSchlüsseln
   NrOfPrinters: Byte;   // Anzahl der Printer
   i: byte;              // Zähler
   Rs: TStrings;         // Ausgabe
begin
     { Initialisierung }
     Printers := TStringList.Create;
     Rs:= TStringList.Create;
     s:='';
     reg:=TRegistry.Create;
     try
      reg.RootKey:=HKEY_LOCAL_MACHINE;
      reg.OpenKey('System\CurrentControlSet\Control\Print\Printers',false);
      if reg.HasSubKeys= true then begin
        reg.GetKeyNames(Printers);
        NrOfPrinters:=Printers.count;
        reg.CloseKey;
        reg.free;
       for i:= 1 to NrOfPrinters do begin
        reg:=TRegistry.create;
        reg.RootKey:=HKEY_LOCAL_MACHINE;
        s:='System\CurrentControlSet\Control\Print\Printers'+'\'+Printers[i-1];
        reg.OpenKey(s,false);
        rs.add(reg.ReadString('Name'));
        rs.add('Anschluss: '+reg.ReadString('Port'));
        rs.add('Freigabename: '+reg.ReadString('Share Name'));
        rs.add('----------------------------------------');
        reg.CloseKey;
        reg.free;
       end;  // end for
     end; //end if HasSubKeys
     except   //end try
      Rs.add('---');
     end;
     strnglst.AddStrings(Rs);
     Printers.Free;
     Rs.free;
end;
```

Sind Drucker vorhanden, so enthält die Stringliste für jeden Drucker folgende Einträge:
„Name_des_Druckers"
„Anschluss: Drucker_Anschluss"
„Freigabename: Freigabename_des_Druckers"
„----------"

Die letzte Zeile dient der optischen Trennung der Informationen, falls mehr als ein Drucker installiert ist.

GetPartitionName:

Diese Funktion ermittelt den Laufwerksnamen einer Partition und gibt das Ergebnis als String zurück. Beispiel: „Win98Platte“

```
function TSysInfo.GetPartitionName(const ADrive: Char): String;
var
  tmp: Integer;
  buffer: array[0..19] of Char;
begin
  GetVolumeInformation(PChar(ADrive+':\'), @buffer[0], SizeOf(buffer), nil, tmp, tmp, nil, 0);
  Result := buffer;
end;
```

GetDriveTyp:

Diese Funktion ermittelt den Typ einer Partition und gibt das Ergebnis als String zurück. Beispiel: „Diskettenlaufwerk“

```
function TSysInfo.GetDriveTyp(Drive: Char): string;
var
   p: Array[0..2] of char;
begin
     strpcopy(p,Drive+':');
      case GetDriveType(p) of
       DRIVE_REMOVABLE : begin
                          if (Drive='A') or (Drive='B') then result:='Diskettenlaufwerk'
                          else result:='RamDisk';
                         end;
       DRIVE_FIXED : result:='Festplatte';
       DRIVE_REMOTE : result:='Netzlaufwerk';
       DRIVE_RAMDISK : result:='RamDisk';
       DRIVE_CDROM : result:='CD-ROM';
       1 : result:='nicht vorhanden';
       else result:='unbekannt';
      end; //case
end;
```

GetTimeZone:

Diese Funktion ermittelt aus der Windows-Registry die Zeitzone und gibt das Ergebnis als String zurück. Beispiel: „Westeuropäische Normalzeit“

```
function TSysInfo.GetTimeZone: string;
var
   reg: TRegistry;
begin
     Result:='---';
     reg:=TRegistry.Create;
     try
      reg.RootKey:=HKEY_LOCAL_MACHINE;
      reg.OpenKey('System\CurrentControlSet\Control\TimeZoneInformation',false);
      Result:=reg.ReadString('StandardName');
      finally
      reg.free;
      end;
      result:=ASCII(result);
end;
```

GetDayOfTheWeek:

Diese Funktion ermittelt den aktuellen Wochentag und gibt das Ergebnis als String zurück. Beispiel: „Montag“

```
function TSysInfo.GetDayOfTheWeek:string;
begin
     case DayOfWeek(now) of
      1: result:='Sonntag';
      2: result:='Montag';
      3: result:='Dienstag';
      4: result:='Mittwoch';
      5: result:='Donnerstag';
      6: result:='Freitag';
      7: result:='Samstag';
     end;
end;
```

GetUptime:

Diese Funktion ermittelt die Zeit seit starten des Betriebssystemes und gibt das Ergebnis als String im Uhrzeitformat zurück. Beispiel: „01:23:45“ oder: „2 Tage, 05:43:21“.

```
function TSysInfo.GetUptime:string;
var
   d,h,m,s: integer;
   uptime: longint;
   uptimedate: TDateTime;
begin
    uptime:=GetTickCount;
    uptime:=trunc(uptime/1000);
    s:=uptime mod 60;
    uptime:=uptime div 60;
    m:=uptime mod 60;
    uptime:=uptime div 60;
    h:=uptime mod 60;
    uptime:=uptime div 60;
    d:=uptime mod 24;
    uptimedate:=encodeTime(h,m,s,0);
    if d>0 then result:=inttostr(d)+' Tage, '+TimeToStr(uptimedate)
    else result:=TimeToStr(uptimedate);
end;
```

GetMostRecentDirectDraw:

Diese Funktion ermittelt aus der Windows-Registry das Programm, welches zuletzt DirectDraw verwendet hat und gibt das Ergebnis als String zurück. Beispiel: „opera.exe“.

```
function TSysInfo.GetMostRecentDirectDraw: string;
var
  Reg: TRegistry;
begin
  Result:='unbekannt';
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('Software\Microsoft\DirectDraw\MostRecentApplication', False) then
    begin
      Result := Reg.ReadString('Name');
      Reg.CloseKey;
    end;
  except
  end;
  Reg.Free;
  result:=ASCII(result);
end;
```

GetMostRecentDirect3D:

Diese Funktion ermittelt aus der Windows-Registry das Programm, welches zuletzt DirectDraw verwendet hat und gibt das Ergebnis als String zurück. Beispiel: „bsplayer.exe“.

```
function TSysInfo.GetMostRecentDirect3D: string;
var
  Reg: TRegistry;
begin
  Result:='unbekannt';
  Reg := TRegistry.Create;
  try
    Reg.RootKey := HKEY_LOCAL_MACHINE;
    if Reg.OpenKey('Software\Microsoft\Direct3D\MostRecentApplication', False) then
    begin
      Result := Reg.ReadString('Name');
      Reg.CloseKey;
    end;
  except
  end;
  Reg.Free;
  result:=ASCII(result);
end;
```

GetOperatingSystem:

Diese Funktion ermittelt den Namen des verwendeten Betriebssystemes und gibt das Ergebnis als String zurück. Beispiel: „Windows 2000“

```
function TSysInfo.GetOperatingSystem: String;
const
  { operating system (OS)constants }
  cOsUnknown = -1;
  cOsWin95 = 0;
  cOsWin98 = 1;
  cOsWin98SE = 2;
  cOsWinME = 3;
  cOsWinNT = 4;
  cOsWin2000 = 5;
  cOsXP = 6;
var
  osVerInfo: TOSVersionInfo;
  majorVer, minorVer: Integer;
begin
  Result := 'unbekannt';
  { set operating system type flag }
  osVerInfo.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  if GetVersionEx(osVerInfo) then
  begin
    majorVer := osVerInfo.dwMajorVersion;
    minorVer := osVerInfo.dwMinorVersion;
    case osVerInfo.dwPlatformId of
      VER_PLATFORM_WIN32_NT: { Windows NT/2000 }
        begin
          if majorVer <= 4 then
            Result := 'Windows NT'
          else if (majorVer = 5) and (minorVer = 0) then
            Result := 'Windows 2000'
          else if (majorVer = 5) and (minorVer = 1) then
            Result := 'Windows XP'
          else
            Result := 'unbekannt';
        end;
      VER_PLATFORM_WIN32_WINDOWS:  { Windows 9x/ME }
        begin
          if (majorVer = 4) and (minorVer = 0) then
            Result := 'Windows 95'
          else if (majorVer = 4) and (minorVer = 10) then
          begin
            if osVerInfo.szCSDVersion[1] = 'A' then
              Result := 'Windows 98 SE'
            else
              Result := 'Windows 98';
          end
          else if (majorVer = 4) and (minorVer = 90) then
            Result := 'Windows ME'
          else
            Result := 'unbekannt';
        end;
      else
        Result := 'unbekannt';
    end;
  end
  else
    Result := 'unbekannt';
end;
```