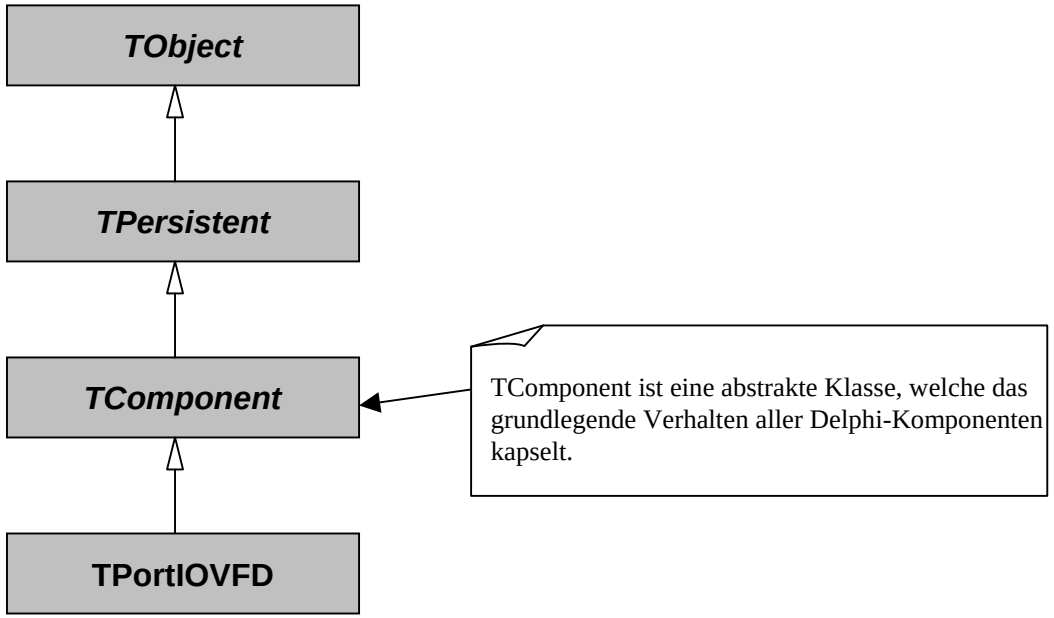
Die Klasse PortIOVFD

Die nichtvisuelle-Komponente „vfd“ des VFD-Studios ist eine Instanz der Klasse PortIOVFD und dient der Ansteuerung des Displays.

Neben Methoden zur Kommunikation mit dem Display und zu dessen Initialisierung beinhaltet sie u.a. Methoden zur Text-, Grafik- und Uhrzeitdarstellung und zum Abspielen von Animationen.

Klassenhierarchie:



Variablen, Objekte und Methoden im Überblick:

Die Bitmap- und Timer-Objekte werden zur Laufzeit erzeugt und sind Eigentum der Komponente *vfd* (Parent-Child-Beziehung).

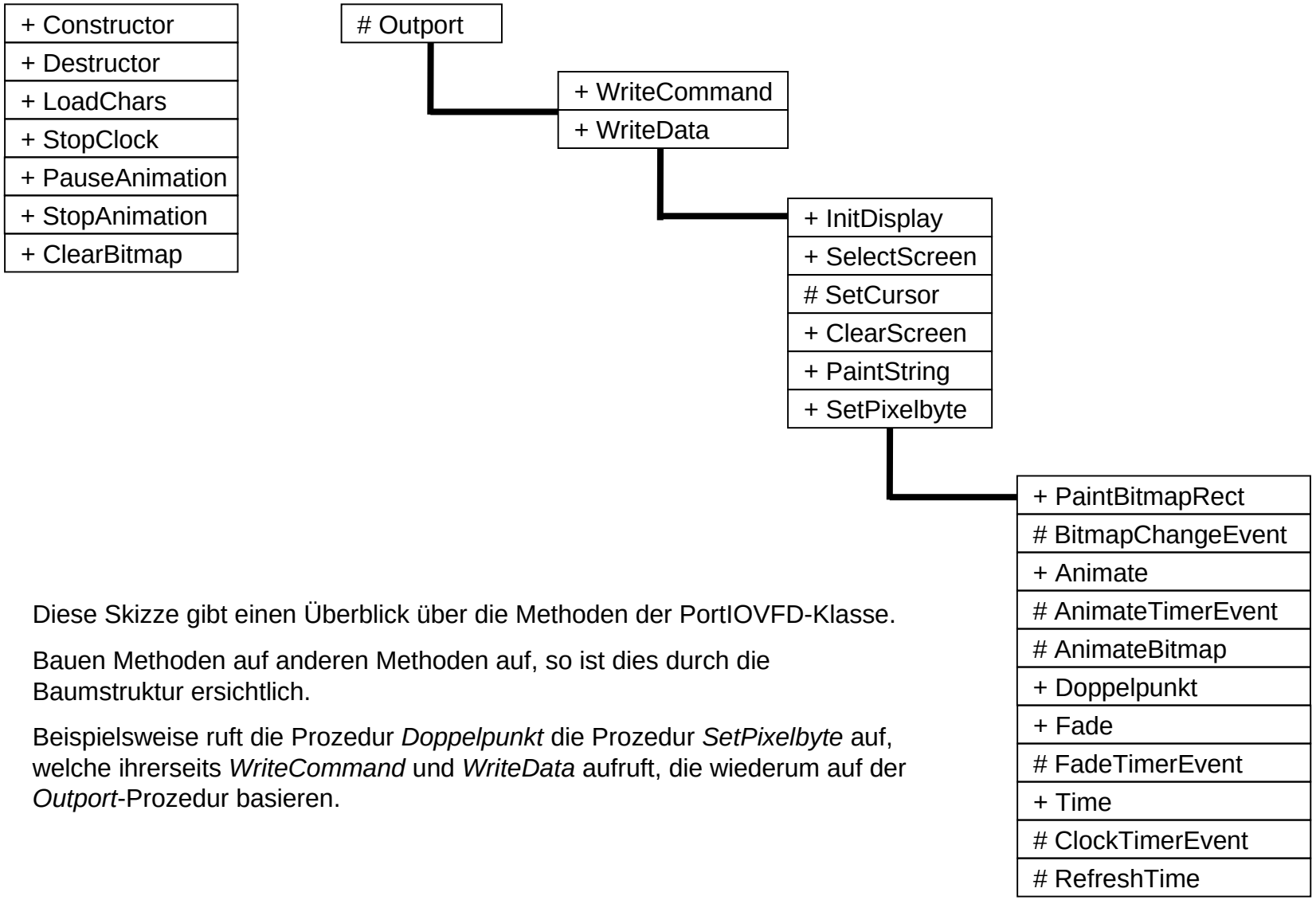
Die PortIOComponent jedoch ist im Besitz des Hauptfensters – es wird lediglich eine Referenz zu dieser Komponente hergestellt (vgl. Dokumentation).

vfd: TPortIOVFD

- BA: word
- DC: array[0..9, 0..191] of byte
- LightLevel: byte
- FOut: boolean
- framewidth: byte
- frame, frames, Xposit : word
- Bmap: TFileName
- Digital: boolean
- Xuhr: byte
- Bmp, ClockBitmap: TBitmap
- FadeTimer, AnimateTimer, ClockTimer: TTimer
+ PortIOComponent: TDLPortIOPortIOComponent
+ LPTPort: word
+ Bitmap: TBitmap

outport(Adresse: word, wert, byte)
Doppenpunkt()
LoadChars()
AnimateTimerEvent(Sender: TObject)
AnimateBitmap(bmp: Tbitmap; xPos,Frame, FrameWidth: word; Farbe: TColor);
FadeTimerEvent(Sender: TObject)
ClockTimerEvent(Sender: TObject)
RefreshTimer()
BitmapChangeEvent(Sender: TObject)
+ constructor Create(Aowner: TComponet)
+ destructor Destroy()
+ InitVFD()
+ WriteCommand(c: byte)
+ WriteData(d: byte)
+ SetCursor(C_low, C_high: byte)
+ ClearScreen(ScreenNr: byte)
+ SelectScreen(Screen: byte)
+ PaintString(s: string; col, row: byte)
+ PaintBitmapRect(Source: TBitmap; xDisplay,yDisplay,width,xOffset: byte; Farbe: TColor)
+ ClearBitmap()
+ SetPixelbyte(Pixelbyte,x,y: byte)
+ PauseAnimation()
+ StopAnimation()
+ Fade(off: boolean; Speed: word)
+ Time(analog: boolean; Xposition: byte)
+ stopClock()

Übersicht über die Methoden



Diese Skizze gibt einen Überblick über die Methoden der PortIOVFD-Klasse.

Bauen Methoden auf anderen Methoden auf, so ist dies durch die Baumstruktur ersichtlich.

Beispielsweise ruft die Prozedur *Doppelpunkt* die Prozedur *SetPixelbyte* auf, welche ihrerseits *WriteCommand* und *WriteData* aufruft, die wiederum auf der *Outport*-Prozedur basieren.

Notation

Im Folgendem die Ablaufdiagramme der Methoden dieser Klasse.

Da wegen der großen Anzahl globaler Variablen, Konstanten, Objekten und Methoden die Übersicht leicht verloren geht gelten dabei folgende Vereinbarungen zur Verbesserung der Lesbarkeit und Verständlichkeit:

Aufruf anderer Methoden:	Methodenname unterstrichen
Parameter der Methode:	„in Anführungszeichen“
Globale Variablen, Konstanten oder Objekte:	*kursiv*
Lokale Variablen der Methode:	**Fettschrift**

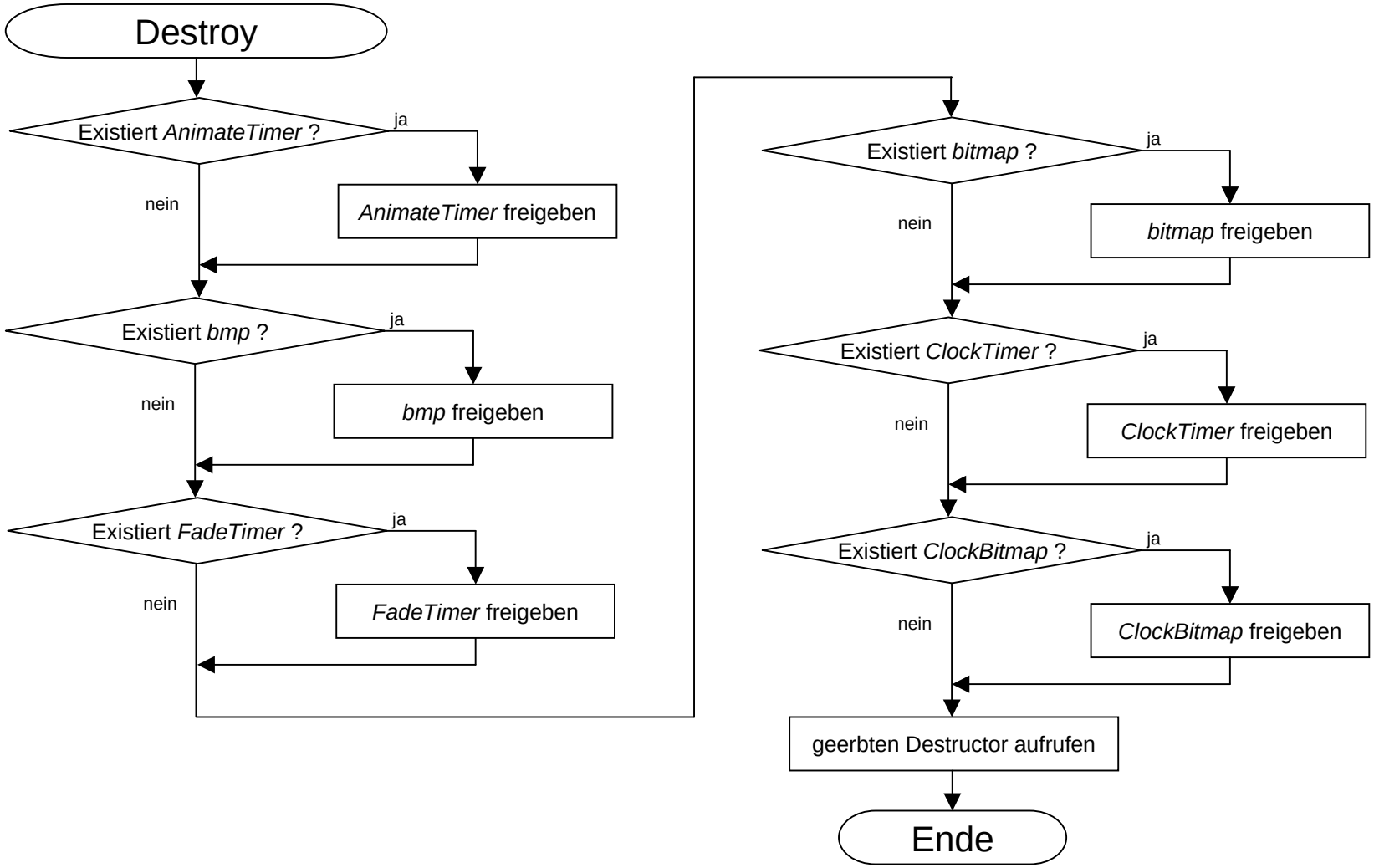
Komplexere Zusammenhänge zwischen Prozeduren und Objekten werden durch Anwendungsfälle in übersichtlicher Form veranschaulicht.
Für nähere Hintergründe sei jedoch auf die Dokumentation verwiesen.

Allgemeine Methoden

constructor TPortIOVFD.Create (AOwner: TComponent)

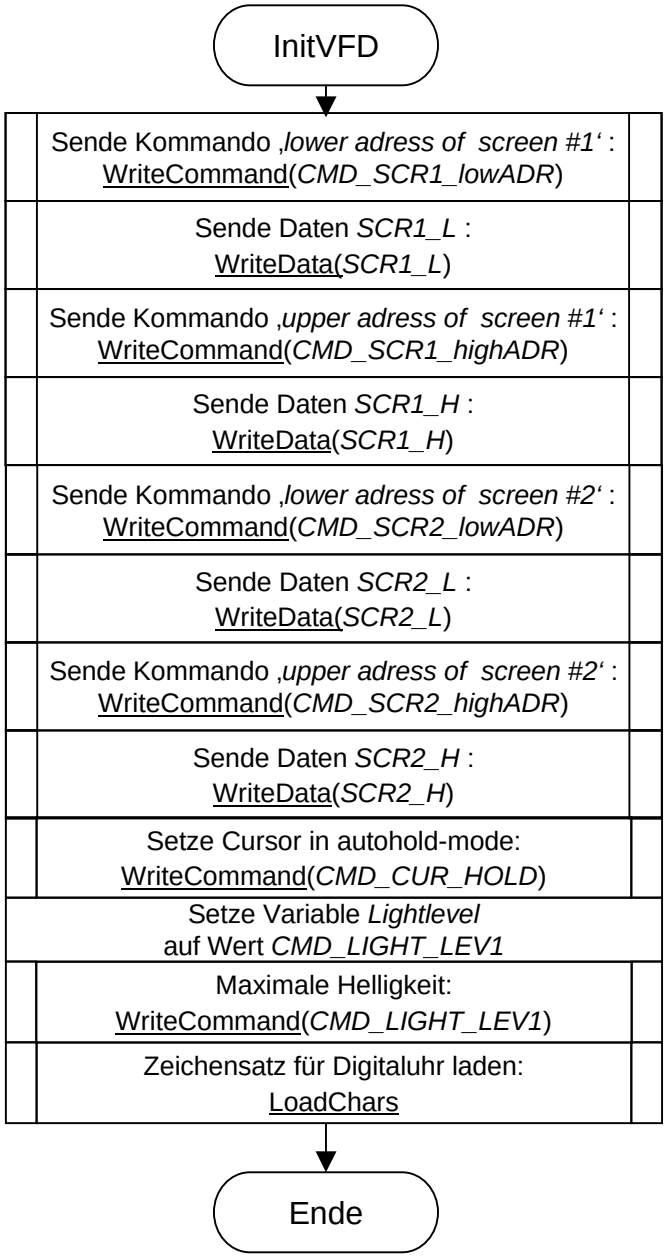
Nicht im Ablaufplan eingezeichnet:
Alle Objekterstellungen mit
try...except-Blöcken gesichert (vgl. Code)

destructor TPortIOVFD.Destroy



Prozedur TPortIOVFD.InitVFD

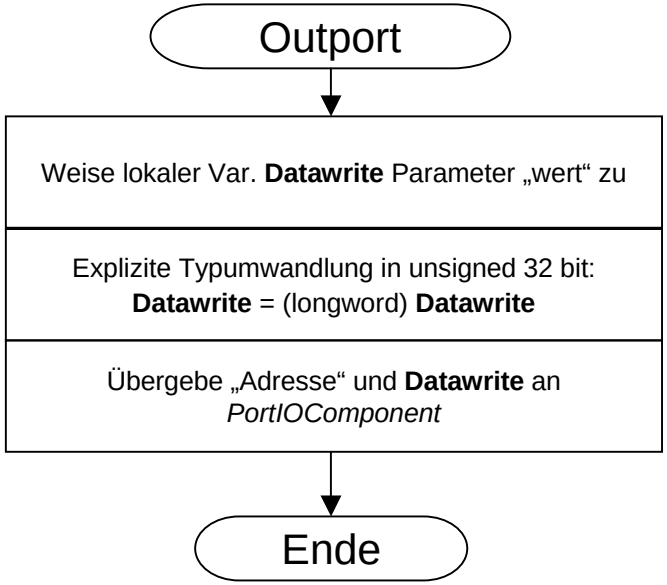
Diese Prozedur muss einmal aufgerufen werden, nachdem eine Instanz der Klasse erzeugt wurde. Es dürfen vor ihr keine anderen Methoden aufgerufen werden (ausgenommen Constructor und Destructor)!



Kommunikation mit dem Display-Controller

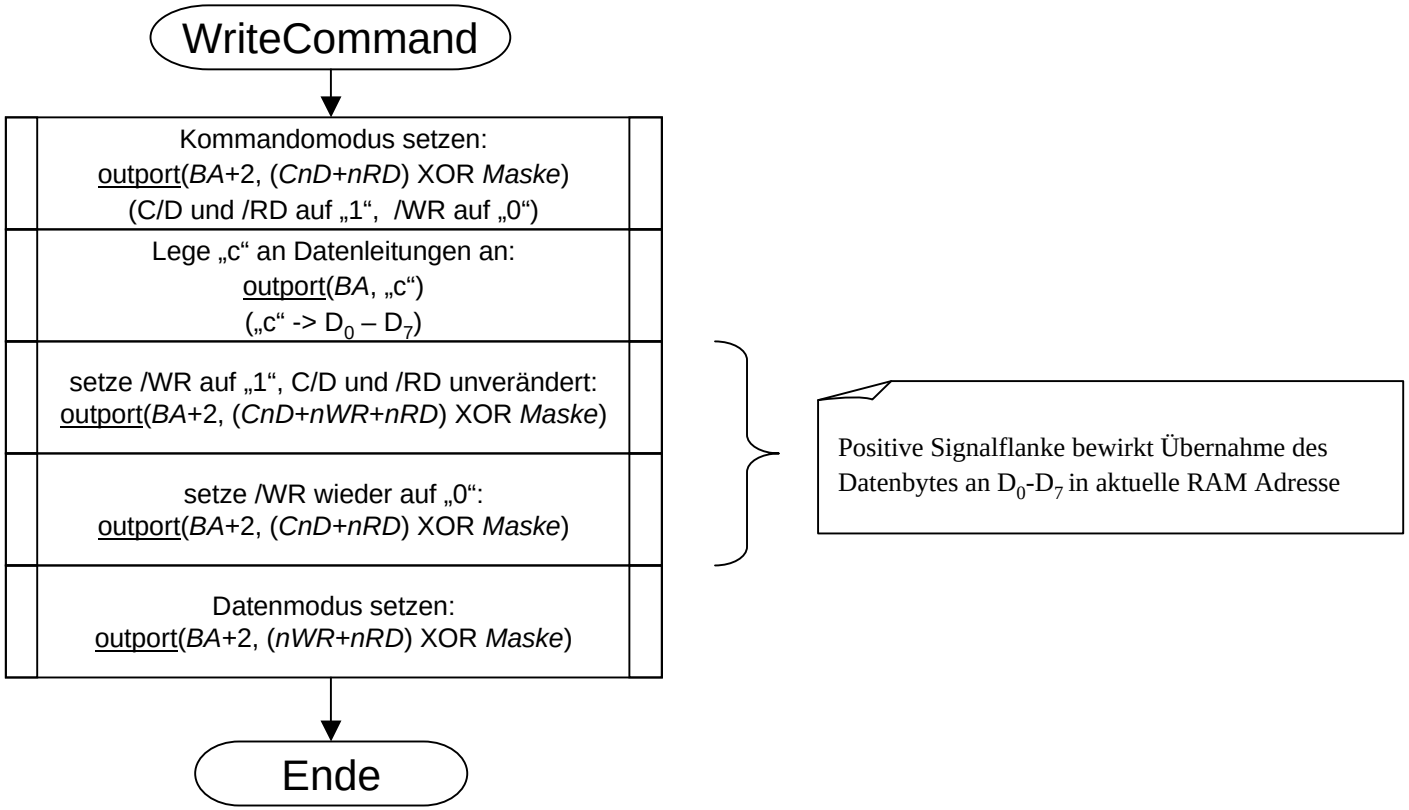
Prozedur TPortIOVFD.Outport (Adresse: word; wert: byte)

Sämtliche Kommunikation mit demDisplay läuft über diese Prozedur!



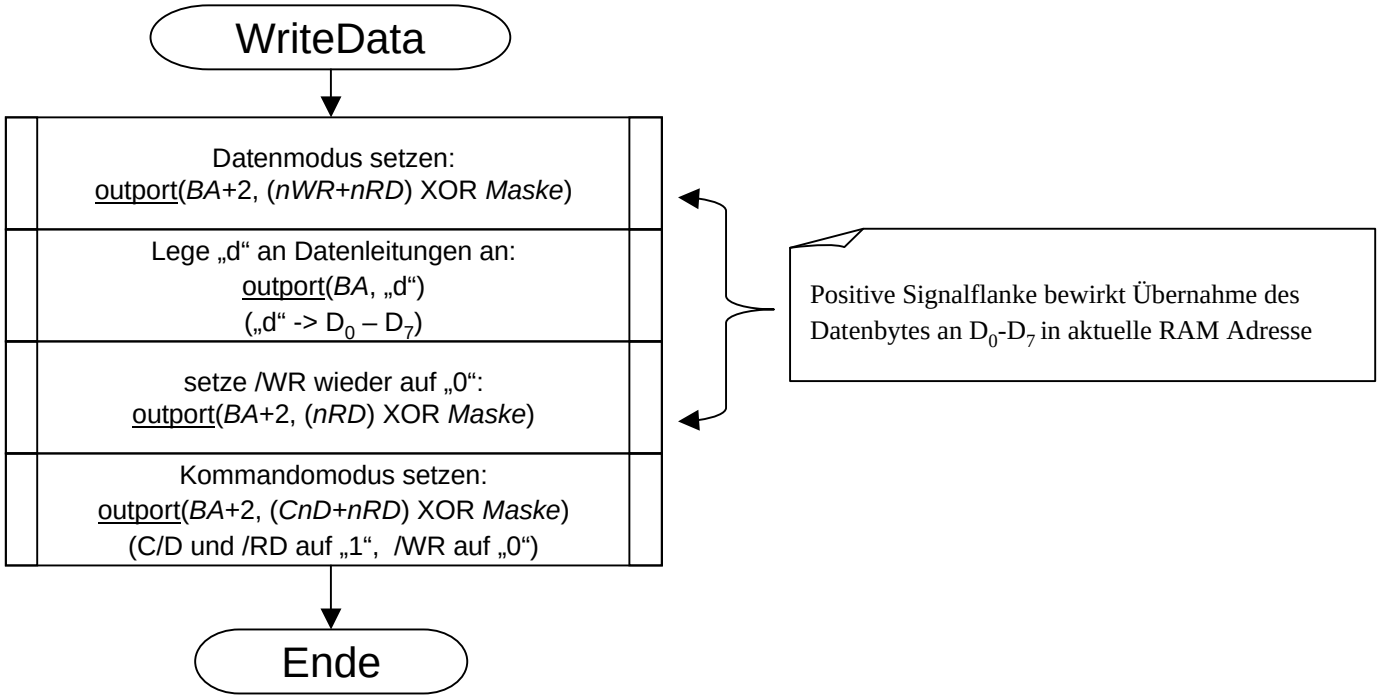
Prozedur TPortIOVFD.WriteCommand (c: byte)

Für Liste der Kommandos vgl. Dokumentation oder Displaydatenblatt.



Prozedur TPortIOVFD.WriteData (d: byte)

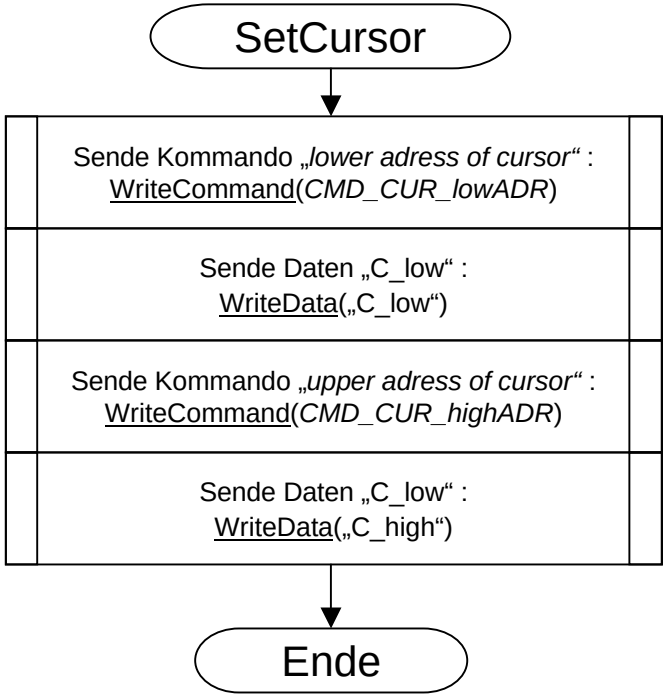
Diese Prozedur dient der Berschreibung einer Adresszelle im Display-RAM. Zuvor muss der Cursor mit SetCursor auf die betreffende Adresse gesetzt werden.



Grundlegende Methoden

Prozedur TPortIOVFD.SetCursor (C_low, C_high: byte)

SetCursor setz den Cursor auf die angegebene Adresse im Display-RAM.

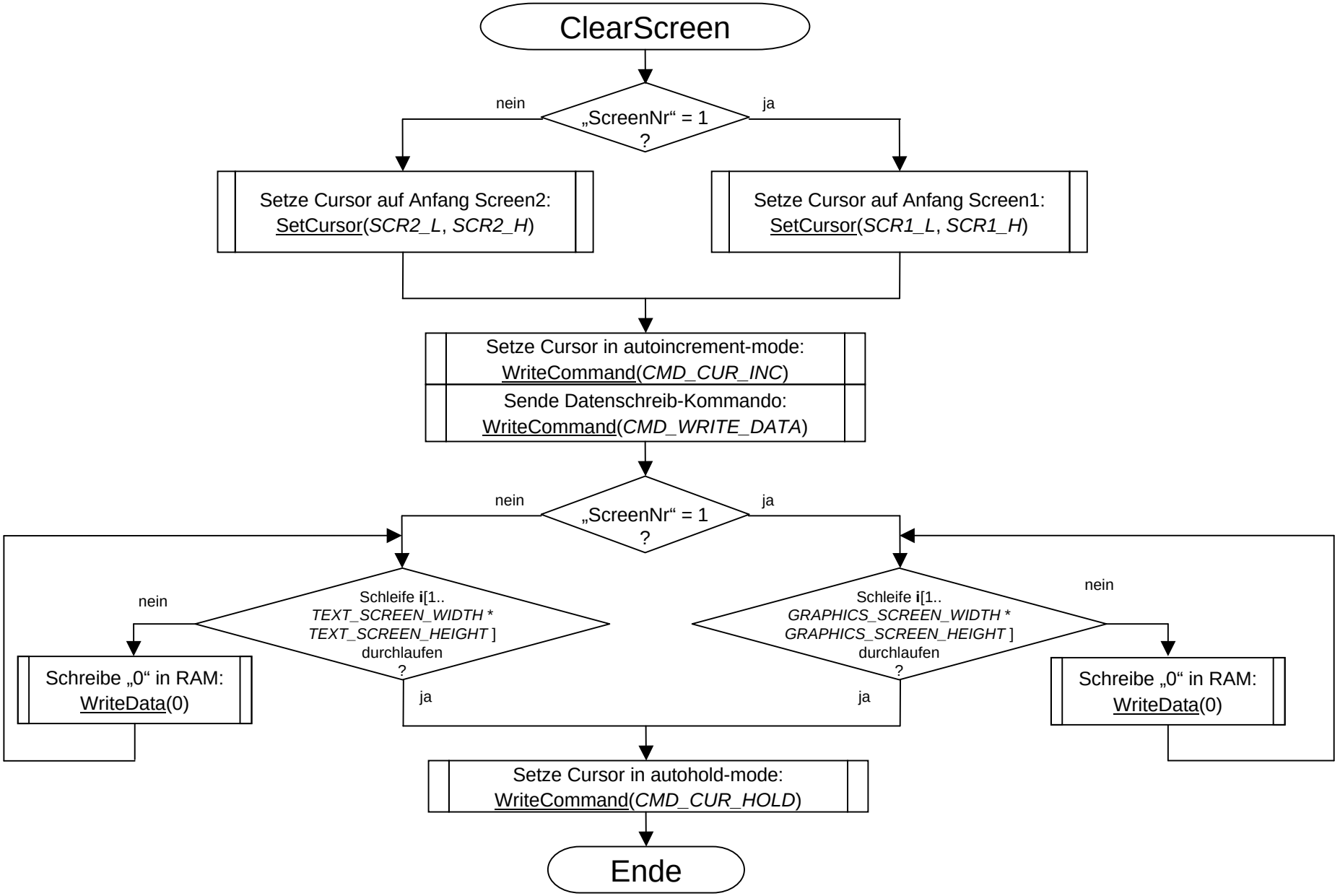


Wichtig ist, dass erst das untere Adressbyte geschrieben wird, dann das obere.

CMD_CUR_lowADR und *CMD_CUR_highADR* sind global definierte Konstanten

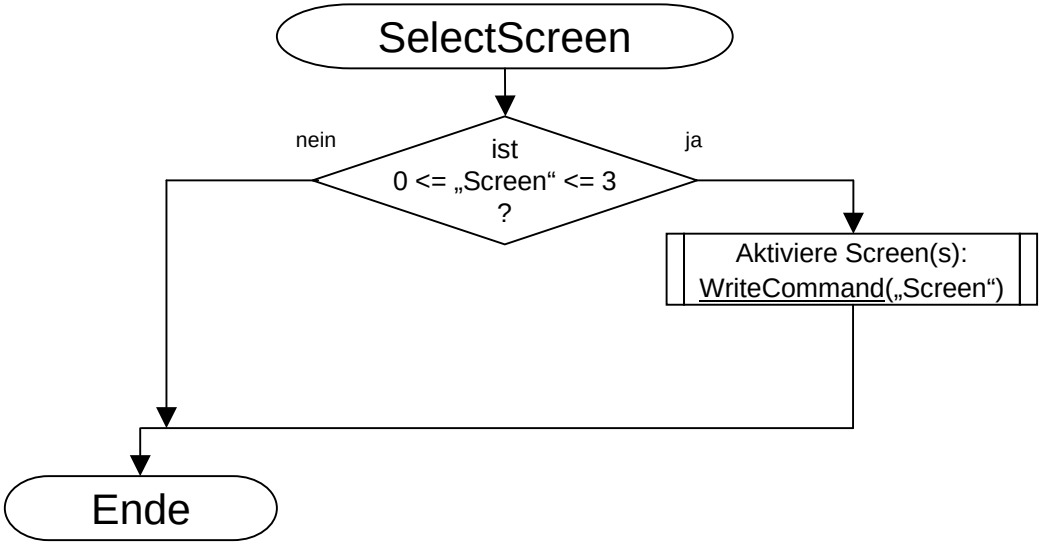
Prozedur TPortIOVFD.ClearScreen (ScreenNr: byte)

Löscht den angegebenen Screen.



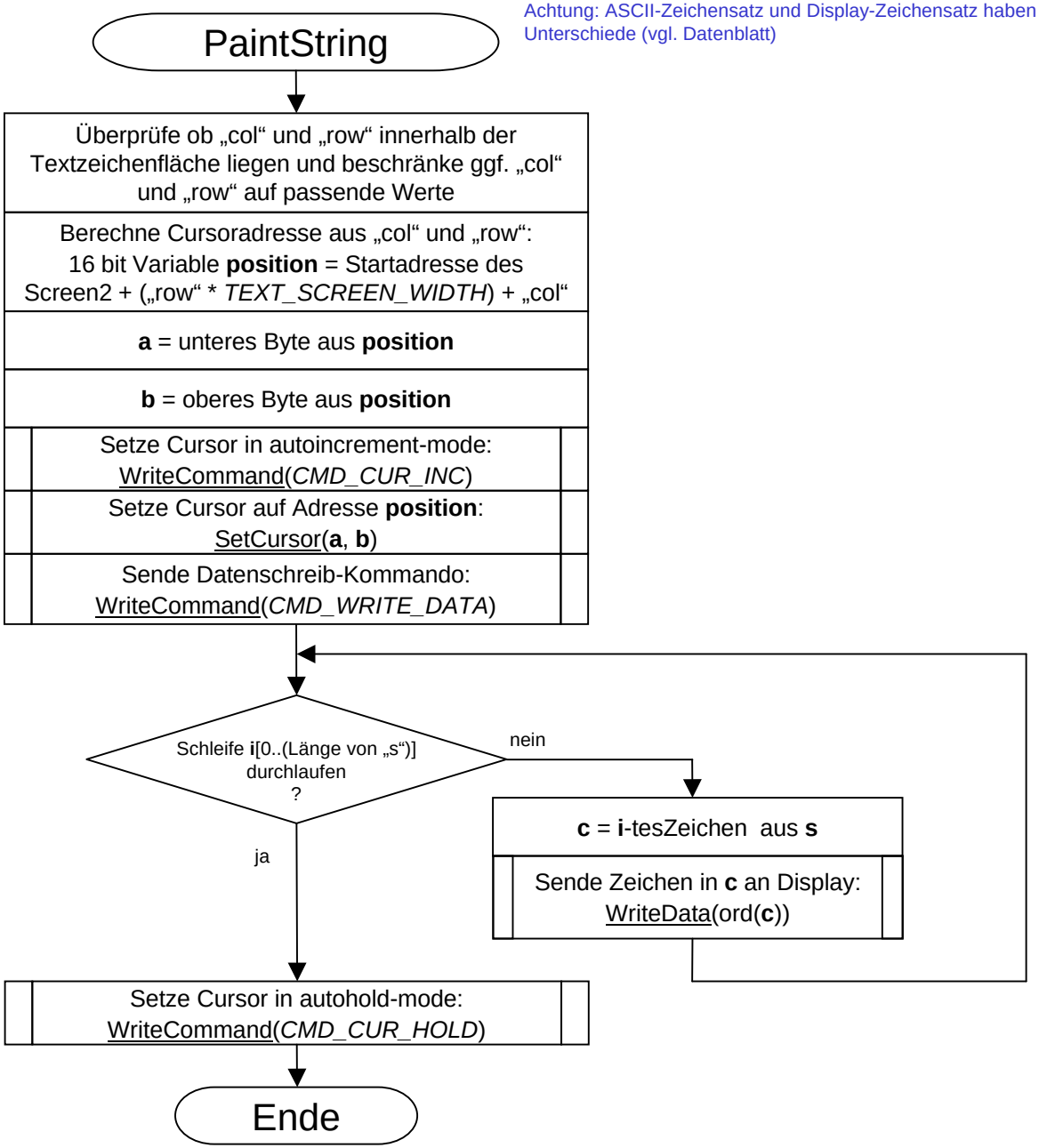
Prozedur TPortIOVFD.SelectScreen (Screen: byte)

Aktiviertden angegebenen Screen. D.h. der Grafik- oder Textbildschirm wird auf dem Display angezeigt.
Ist Parameter „Screen“ gleich 3, werden beide Screens aktiviert; ist „Screen“ gleich 0, wird kein Screen angezeigt.



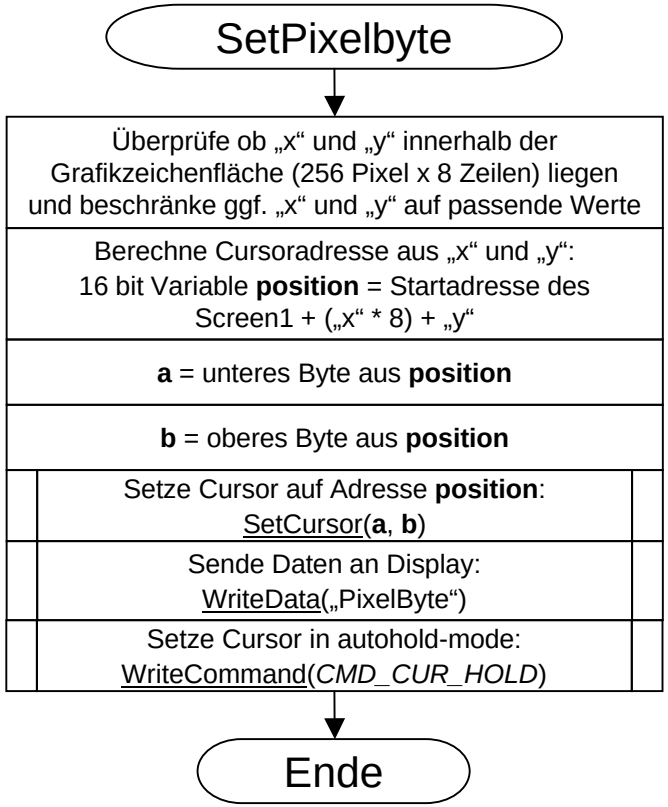
Prozedur TPortIOVFD.PaintString (s: **string**; col,row: byte)

Dient der Ausgabe einer String-Zeichenkette auf dem Display.
Achtung: ASCII-Zeichensatz und Display-Zeichensatz haben kleine Unterschiede (vgl. Datenblatt)



Prozedur TPortIOVFD.SetPixelbyte (PixelByte,x,y: byte)

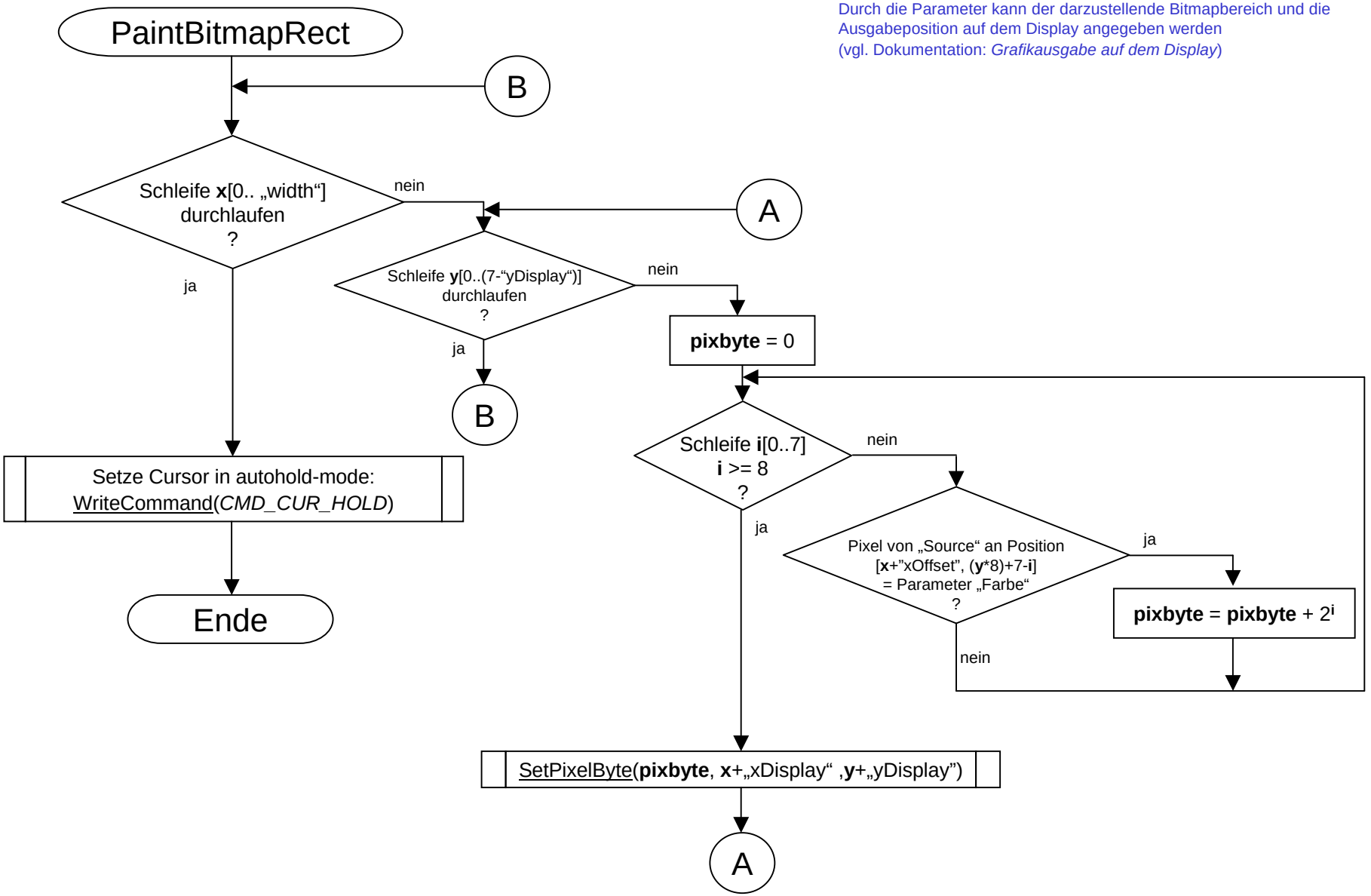
Dient der Darstellung eines „Pixel-Balkens“ - s. Dokumentation



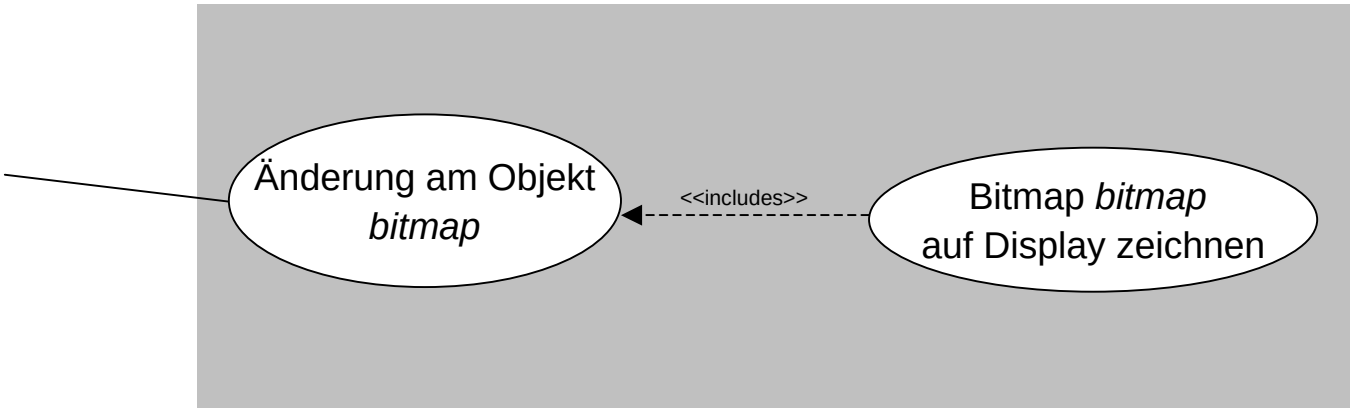
Grafikausgabe

Prozedur TPortIOVFD.PaintBitmapRect(`Source: TBitmap;xDisplay,yDisplay,width,xOffset: byte; Farbe: Tcolor)`

PaintBitmapRect kann Bitmaps (enthalten in „Source“) darstellen.
Durch die Parameter kann der darzustellende Bitmapbereich und die Ausgabeposition auf dem Display angegeben werden
(vgl. Dokumentation: *Grafikausgabe auf dem Display*)



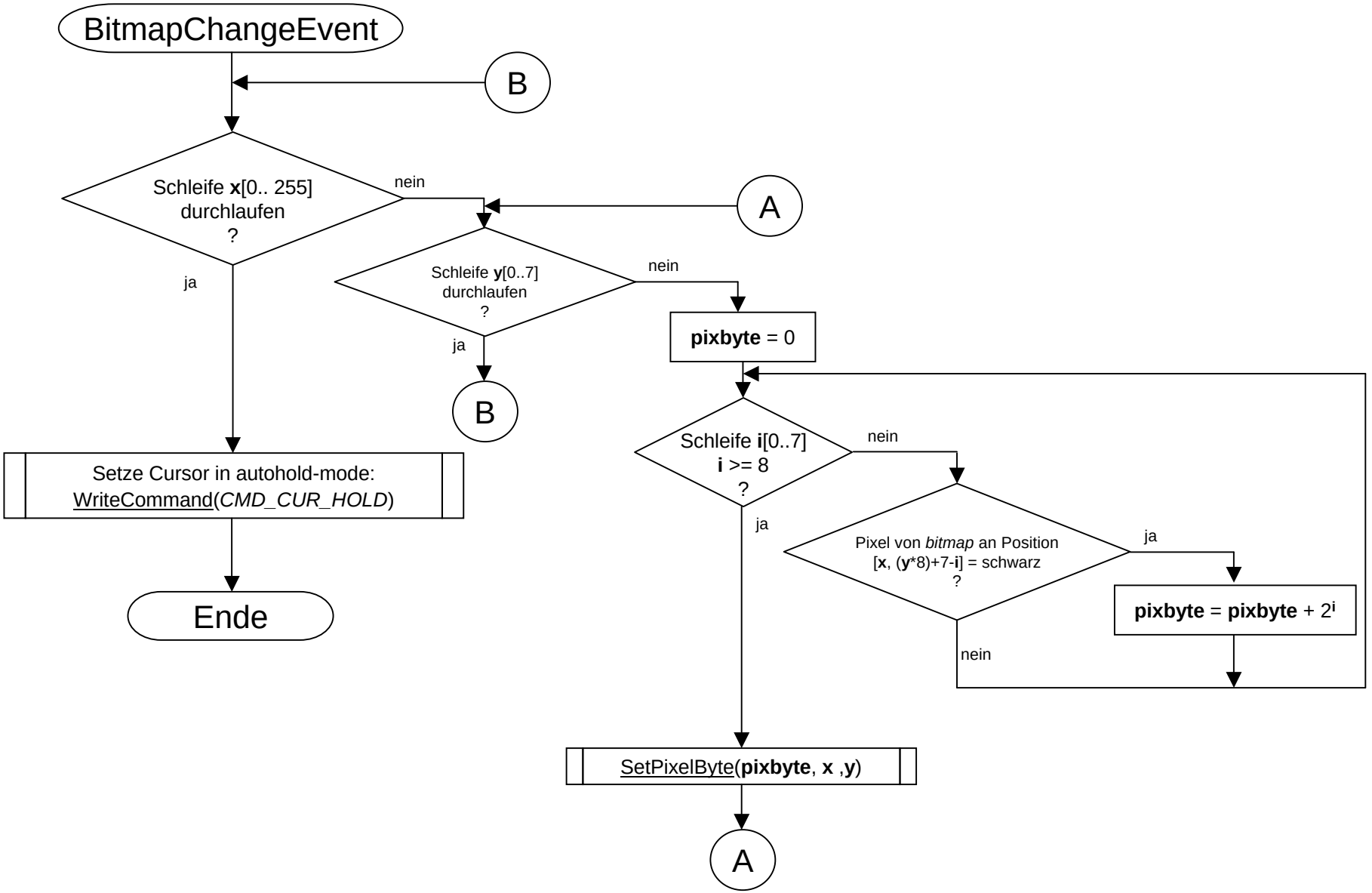
Anwendungsfall
Grafikausgabe mitt BitmapChangeEvent



Eine Zeichenoperation auf der Zeichenfläche Bitmapobjekt *bitmap* führt immer zum Aufruf der Prozedur *BitmapChangeEvent*, die die *bitmap*-Zeichenfläche auf das Display kopiert.
Soll *bitmap* geändert werden, ohne dass die Zeichenfläche auf das Display kopiert wird muss die Umleitung des OnChange-Events des Bitmapobjektes auf die Prozedur *BitmapChangeEvent* aufgehoben werden – siehe *ClearBitmap*.

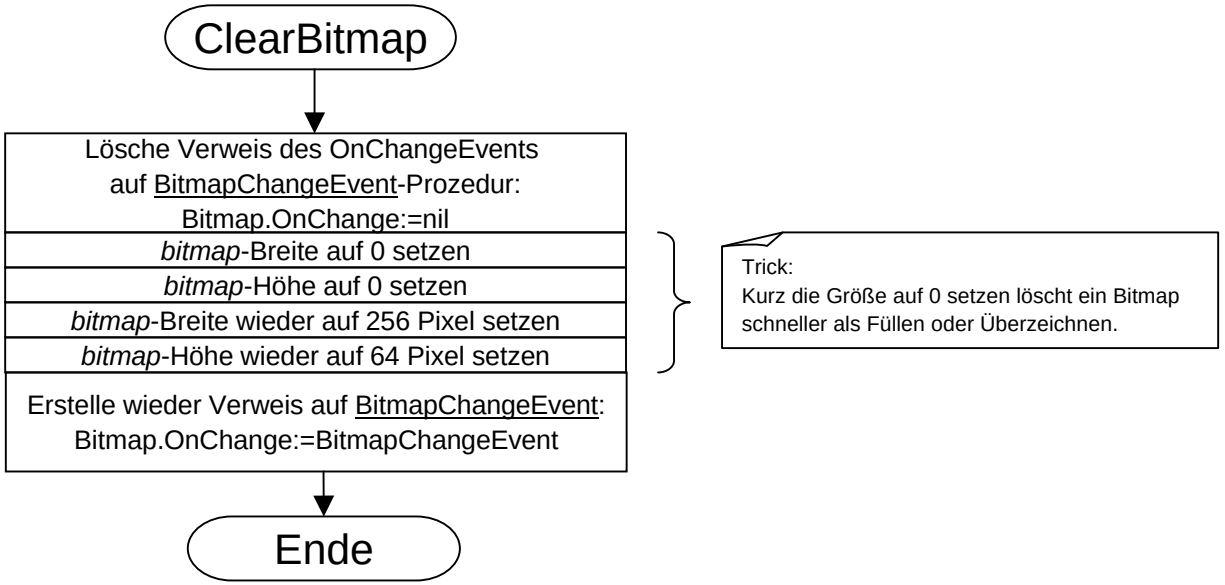
Prozedur TPortIOVFD.BitmapChangeEvent(`Sender: TObject`)

Diese Prozedur wird bei jeder Änderung des bitmap-Objektes *bitmap* aufgerufen.



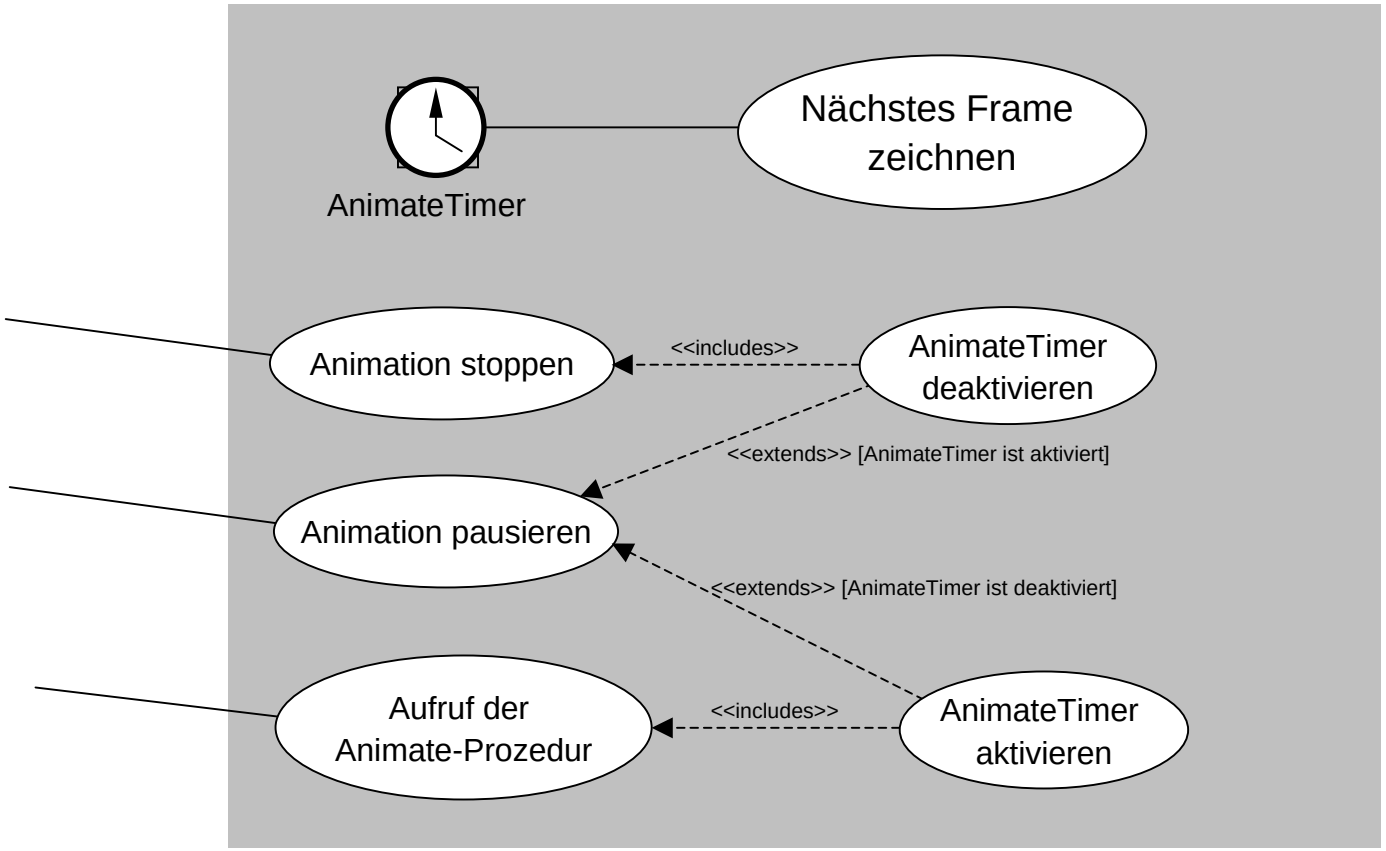
Prozedur TPortIOVFD.ClearBitmap

Diese public Prozedur löscht die Zeichenfläche des internen Bitmapobjekts *bitmap*, welches u.a. dazu benutzt wird Texte in versch. Schriftarten darzustellen.
Um zu verhindern, daß nach der Löschung die Bitmap auf das Display gezeichnet wird (siehe BitmapChangeEvent) wird dazu der Verweis des OnChangeEvents auf BitmapChangeEvent kurzfristig außer Kraft gesetzt.



Abspielen von Animationen

Anwendungsfall
Animation

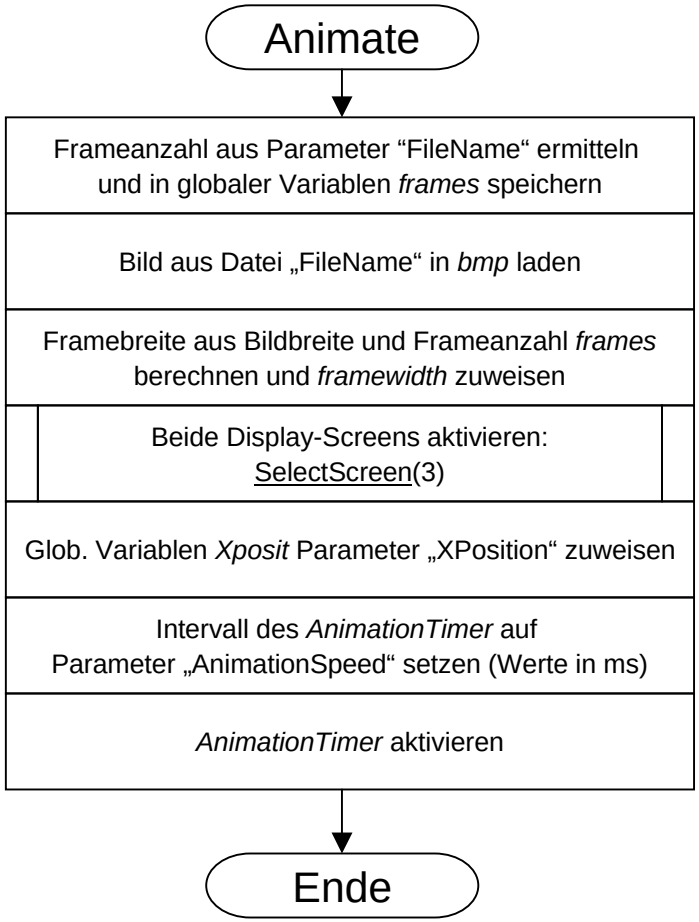


Durch den Aufruf der *Animate*-Prozedur wird eine Animation abgespielt. Dazu wird der *AnimateTimer* aktiviert, welcher bei jedem Intervall das nächste Einzelbild zeichnet.

Prozedur TPortIOVFD.Animate (FileName: TFilename; XPosition, AnimationSpeed: word)

Diese public Prozedur wird vom Hauptprogramm aufgerufen und übergibt den Dateinamen der Animation, X-Position auf dem Display und Abspielgeschwindigkeit.

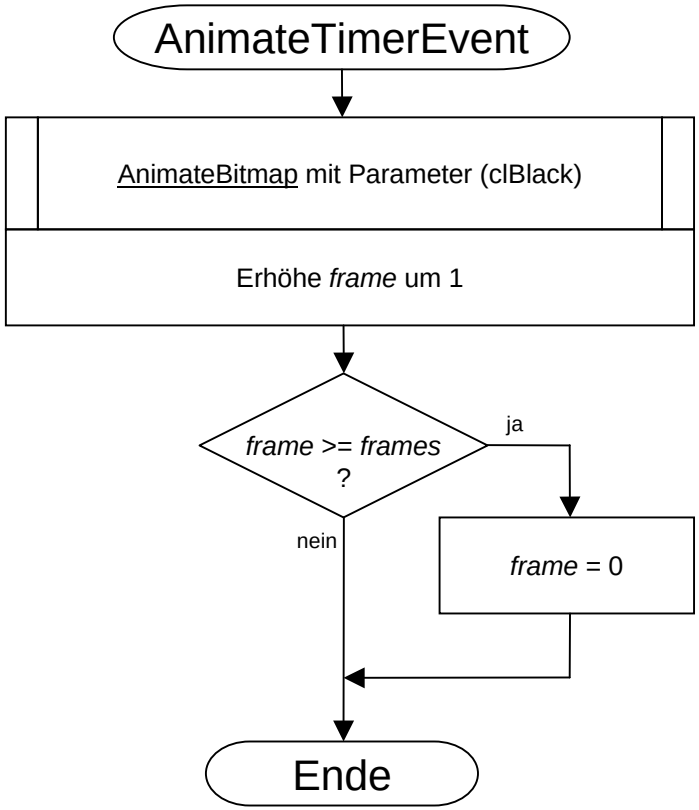
Die Abspielgeschwindigkeit „AnimationSpeed“ enthält die Anzeigedauer eines Frames in Millisekunden.



Prozedur TPortIOVFD.AnimateTimerEvent (Sender: TObject)

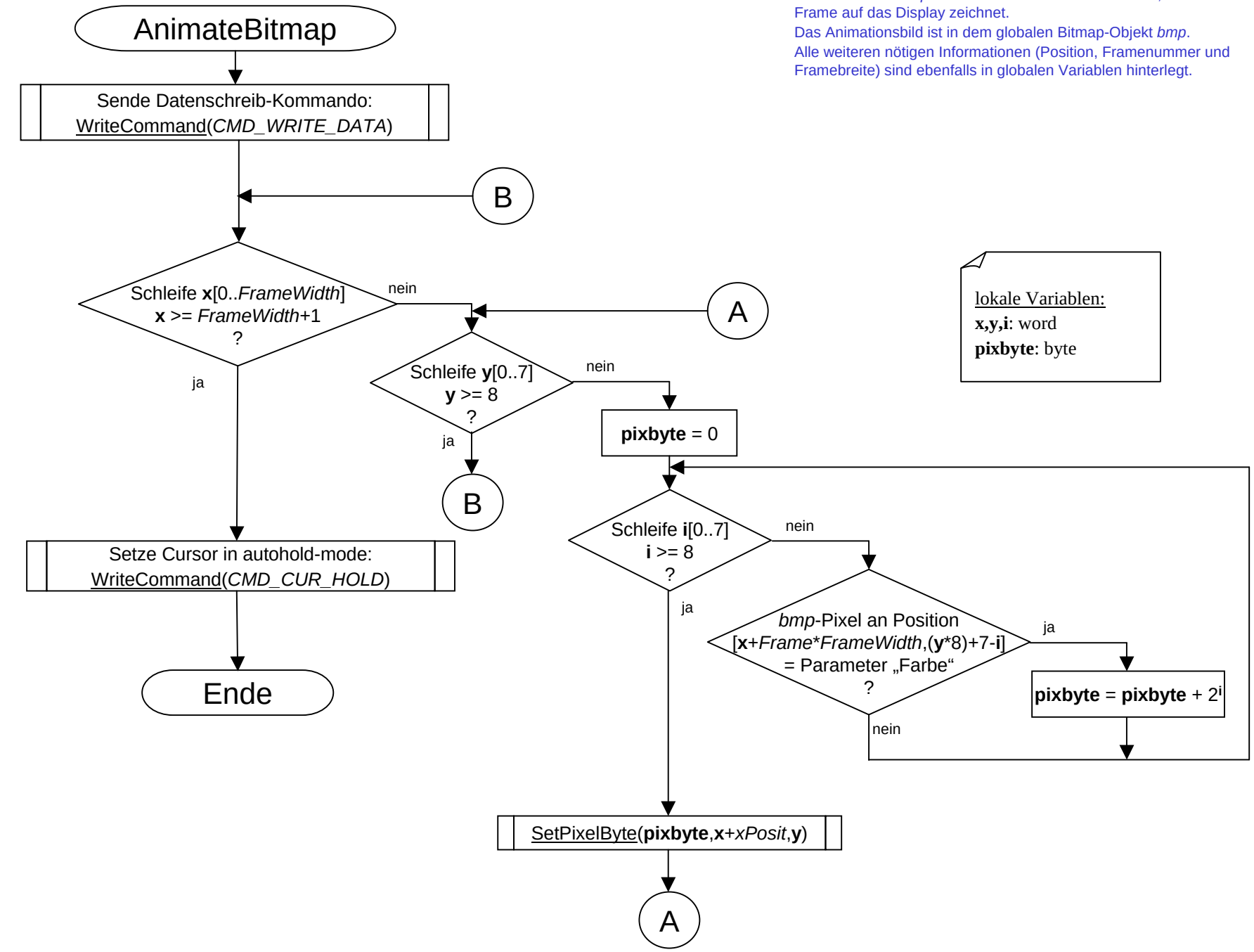
Sobald der *AnimateTimer* durch die Prozedur *Animate* aktiviert wurde, wird bei jedem OnTimer-Ereignis diese Prozedur aufgerufen.

Sie ruft ihrerseits die Prozedure *AnimateBitamp* auf, die das Frame zeichnet, Und erhöht anschließend den *frame*-Zähler, damit beim nächsten Timer-Event das folgende Frame gezeichnet wird.



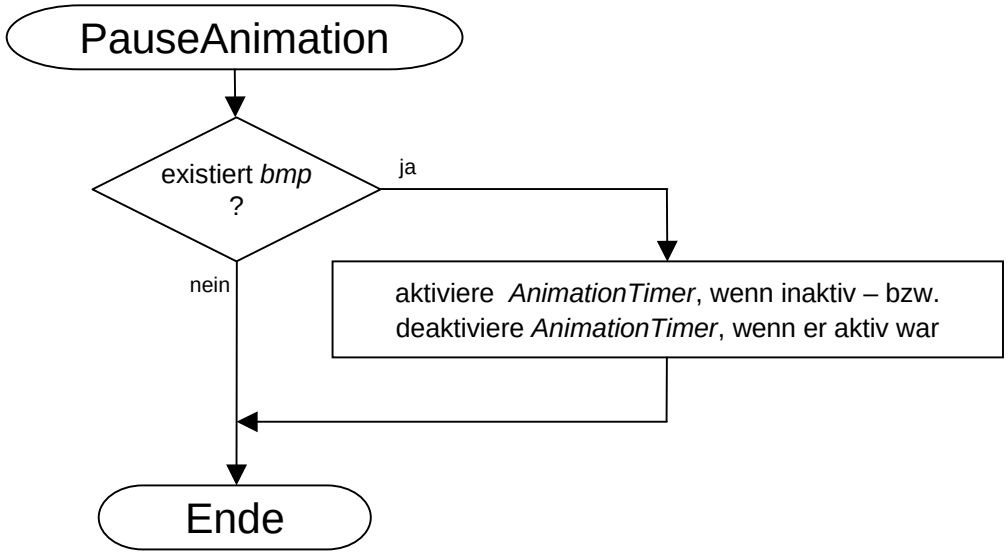
Prozedur TPortIOVFD.AnimateBitmap (Farbe: TColor)

Ähnlich *PaintBitmapRect* arbeitet auch diese Prozedur, die das aktuelle Frame auf das Display zeichnet.
Das Animationsbild ist in dem globalen Bitmap-Objekt *bmp*.
Alle weiteren nötigen Informationen (Position, Framenummer und Framebreite) sind ebenfalls in globalen Variablen hinterlegt.



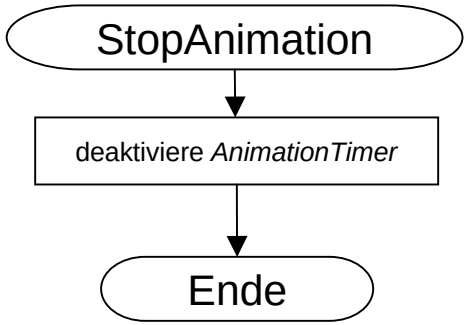
Prozedur TPortIOVFD.PauseAnimation

Hält den Timer an, bzw. startet ihn.



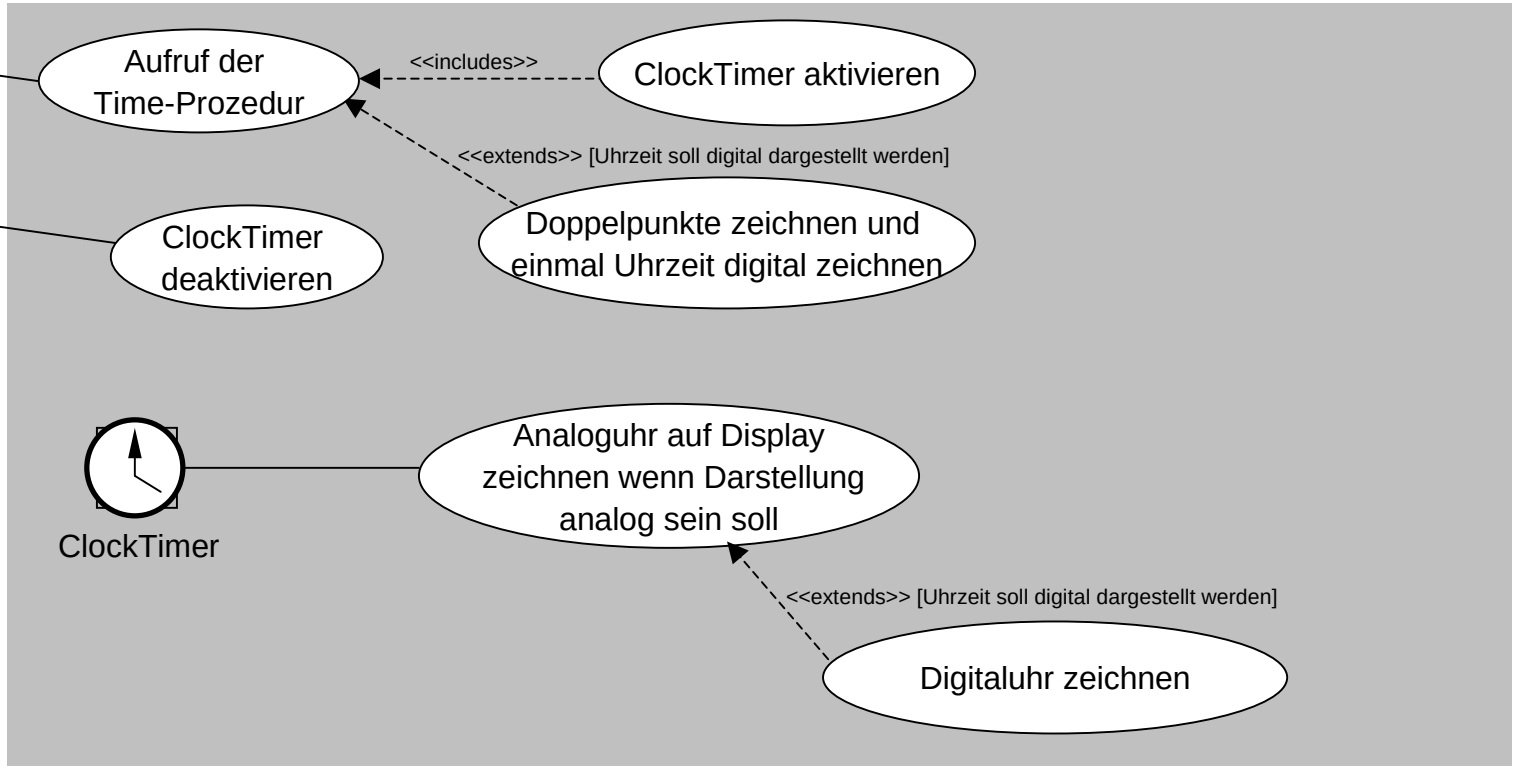
Prozedur TPortIOVFD.StopAnimation

Hält den Timer an.



Anzeige der Uhrzeit

Anwendungsfall
Uhrzeit



Die Uhrzeit kann wahlweise analog oder digital dargestellt werden. Die Time-Prozedur aktiviert den ClockTimer und zeichnet, falls die digitale Darstellungsart gewählt wurde die Doppelpunkte (: :) auf das Display.

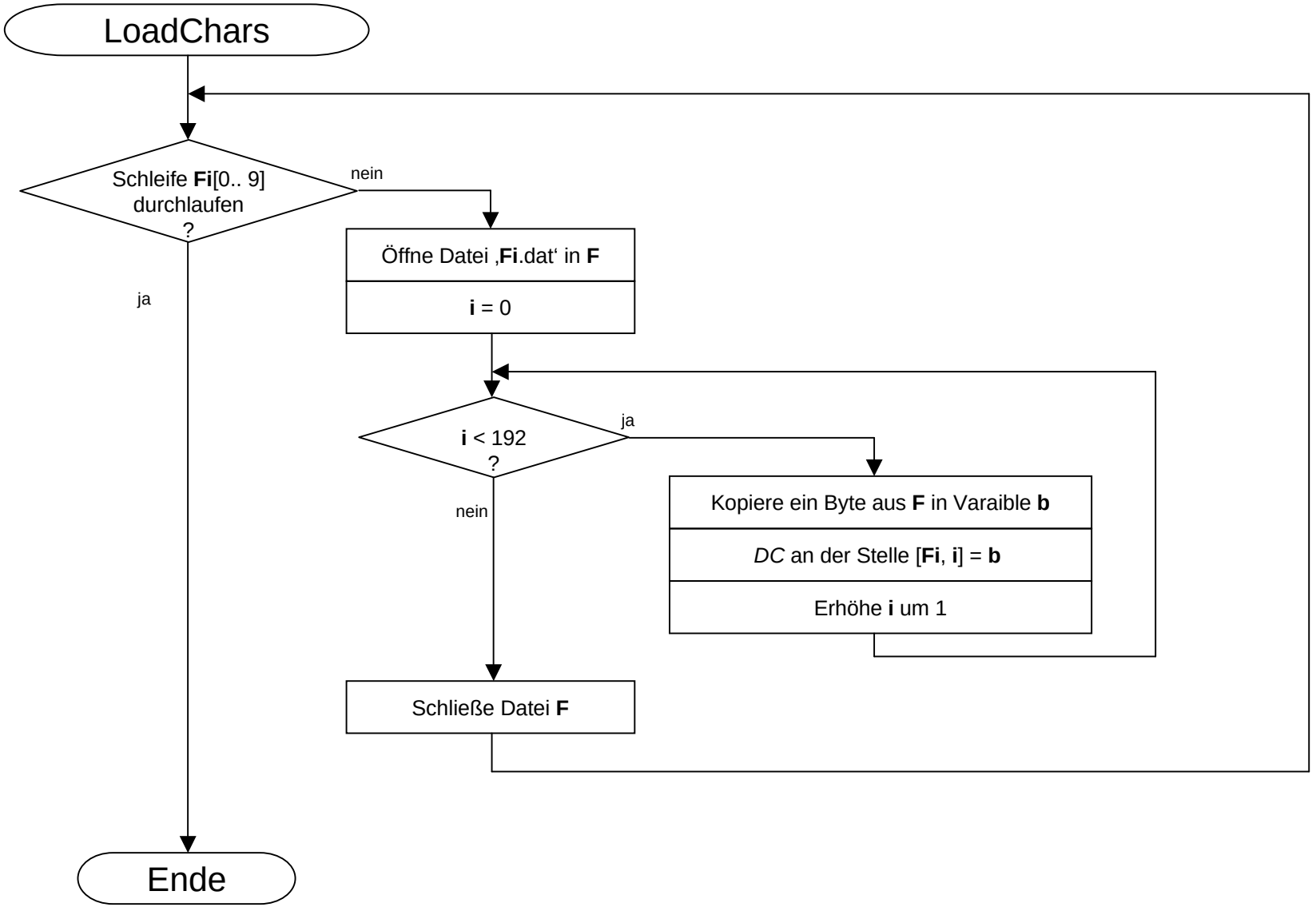
Der ClockTimer ruft bei jedem Intervall die ClockTimerEvent-Prozedur auf, die abhängig von der Darstellungsart die Analoguhr auf das Display zeichnet oder ihrerseits wiederum die Prozedur RefreshTime aufruft. RefreshTime zeichnet die Digitaluhr auf das Display.

Damit die digitale Uhrzeit dargestellt werden kann muss die Prozedutr LoadChars zuvor einmal aufgerufen worden sein,was durch die Prozedur InitDisplay geschieht.

Der ClockTimer wird vom Hauptfenster automatisch wieder deaktiviert, sobald ein neuer Screen aufgebaut wird (s. Dokumentation).

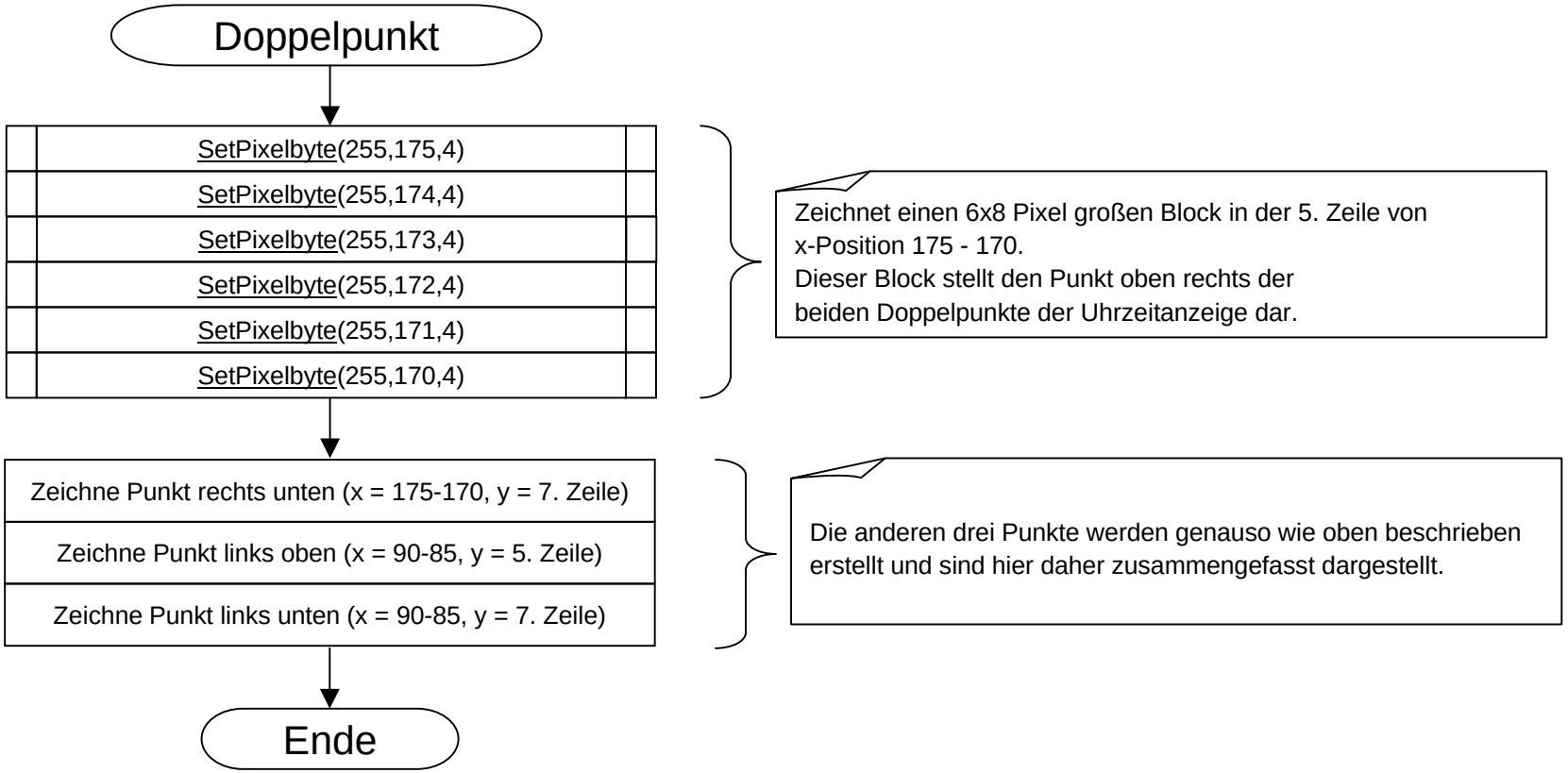
Prozedur TPortIOVFD.LoadChars

Kopiert den Zeichensatz für die Darstellung der Digitaluhrziffern aus externen Dateien in ein zweidimensionales Array im Speicher.
Siehe Dokumentation für geaue Erklärung.



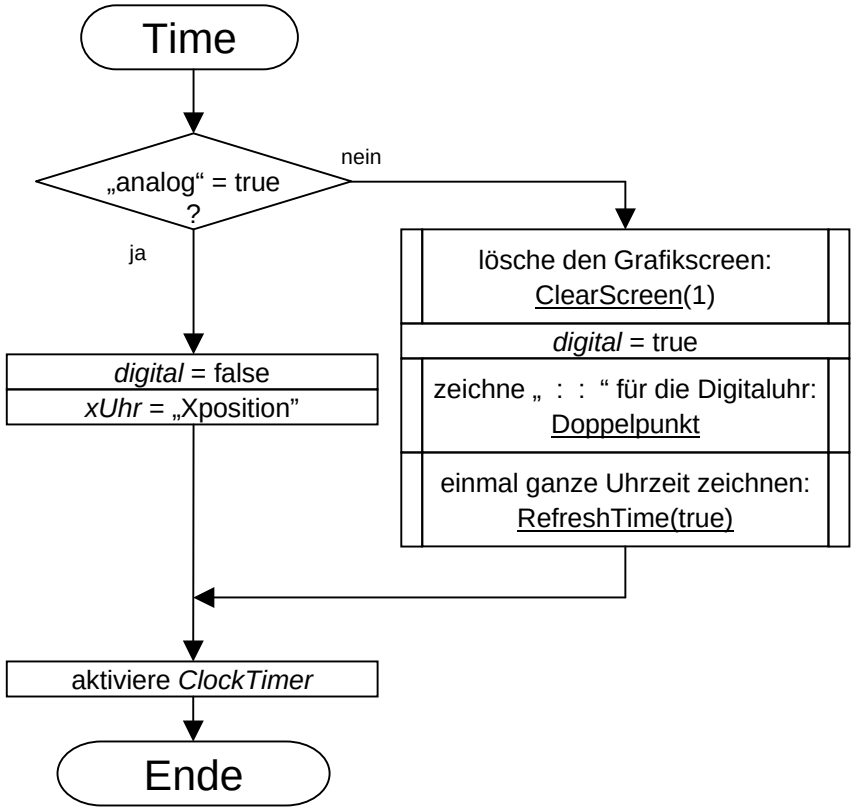
Prozedur TPortIOVFD.Doppelpunkt

Zeichnet „ : : “ auf das Display für die
Stunden-Minuten-Sekunden-Zeitformatierung der Digitaluhrenanzeige



Prozedur TPortIOVFD.Time(analog: boolean; Xposition: byte)

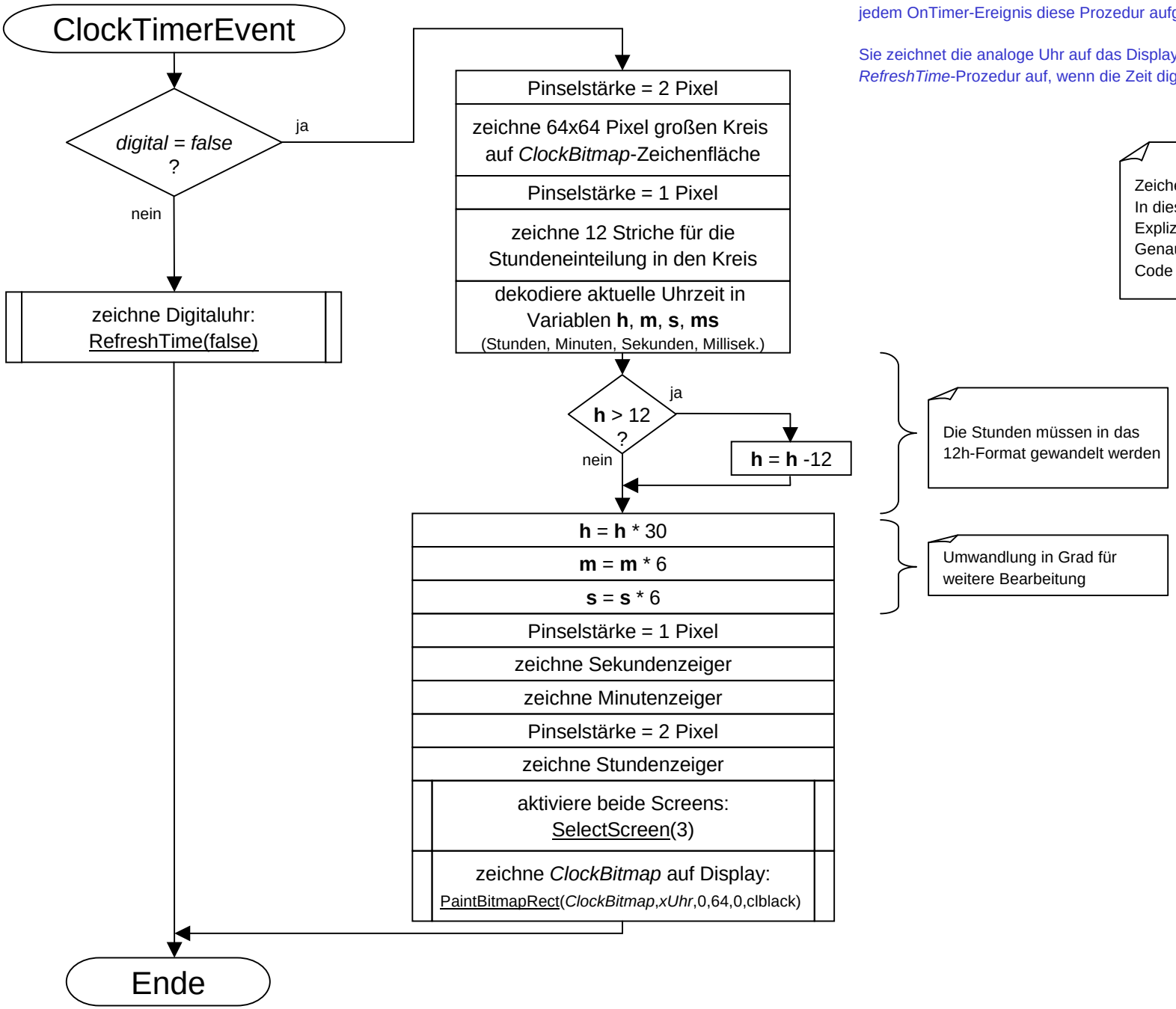
Diese public-Prozedutr zeigt die aktuelle Uhrzeit an.
Wahlweise in analoger Darstellung (Ziffernblatt) oder als Digitaluhr.
Bei Analogdarstellung kann die Position des 64x64 Pixel großen Ziffernblatt angegeben werden.Bei digitaler Darstellung bleibt dieser Parameter unberücksichtigt.



Prozedur TPortIOVFD.ClockTimerEvent (Sender: TObject)

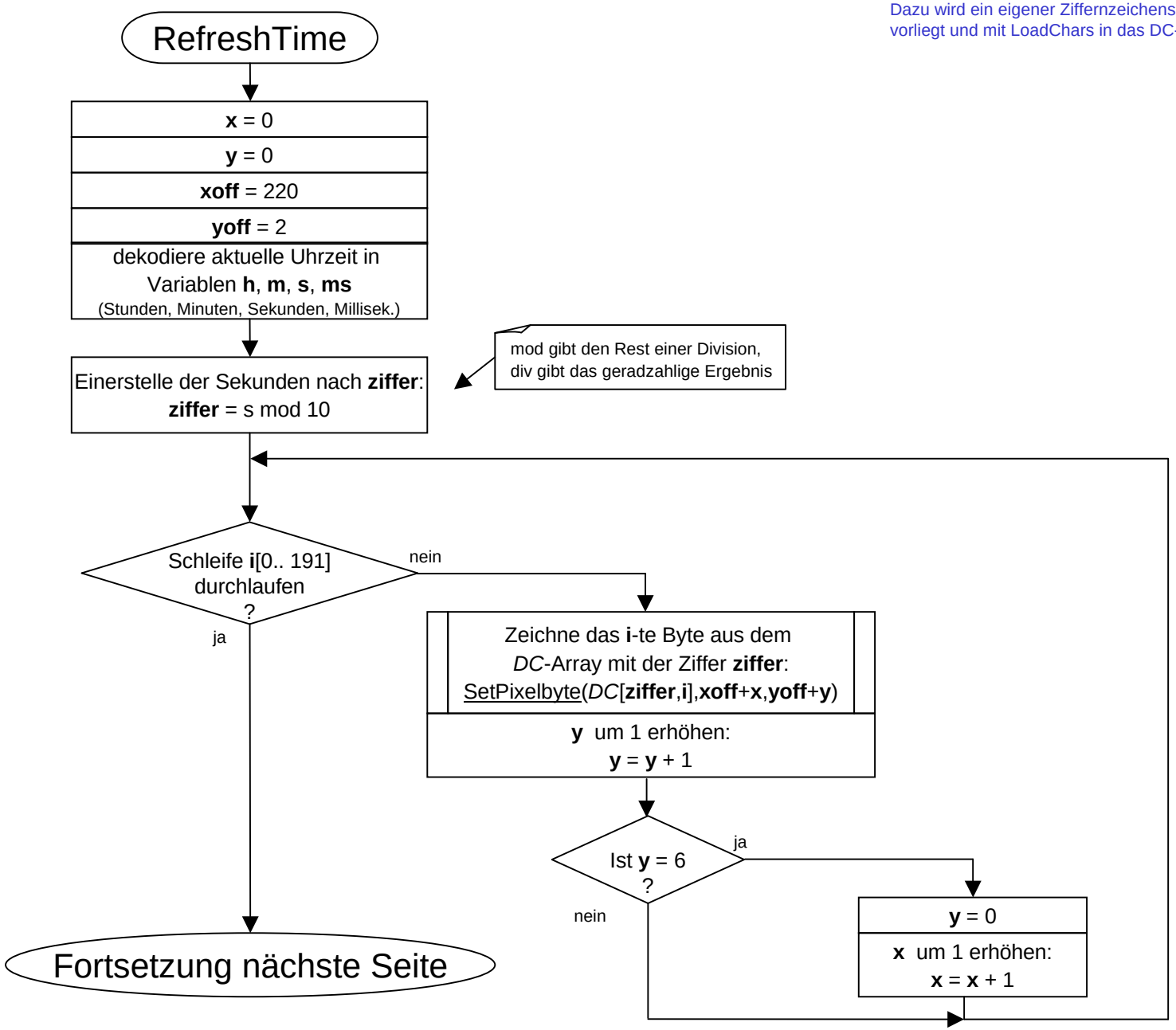
Sobald der *ClockTimer* durch die Prozedur *Time* aktiviert wurde, wird bei jedem OnTimer-Ereignis diese Prozedur aufgerufen.

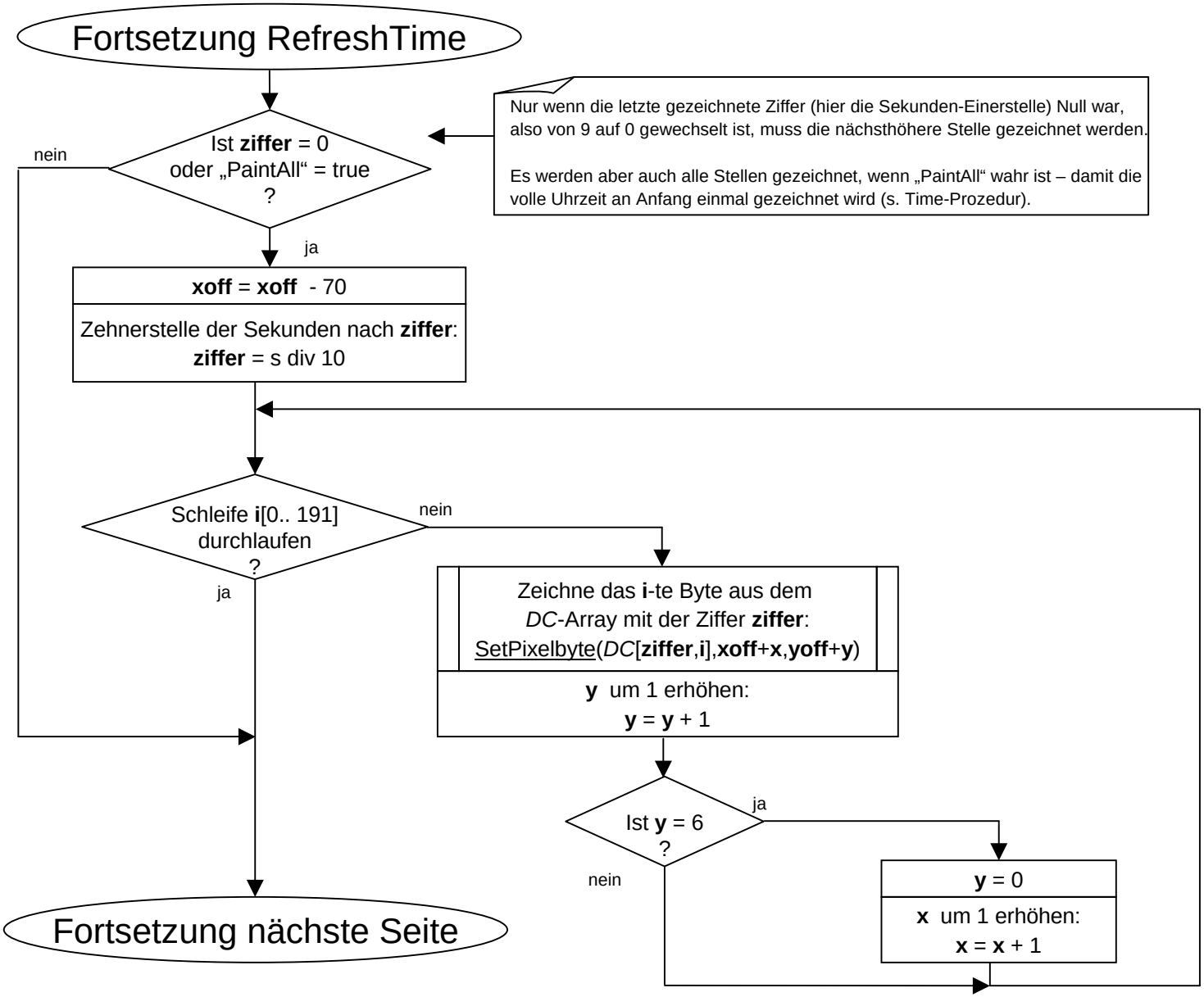
Sie zeichnet die analoge Uhr auf das Display oder ruft die *RefreshTime*-Prozedur auf, wenn die Zeit digital dargestellt werden soll.

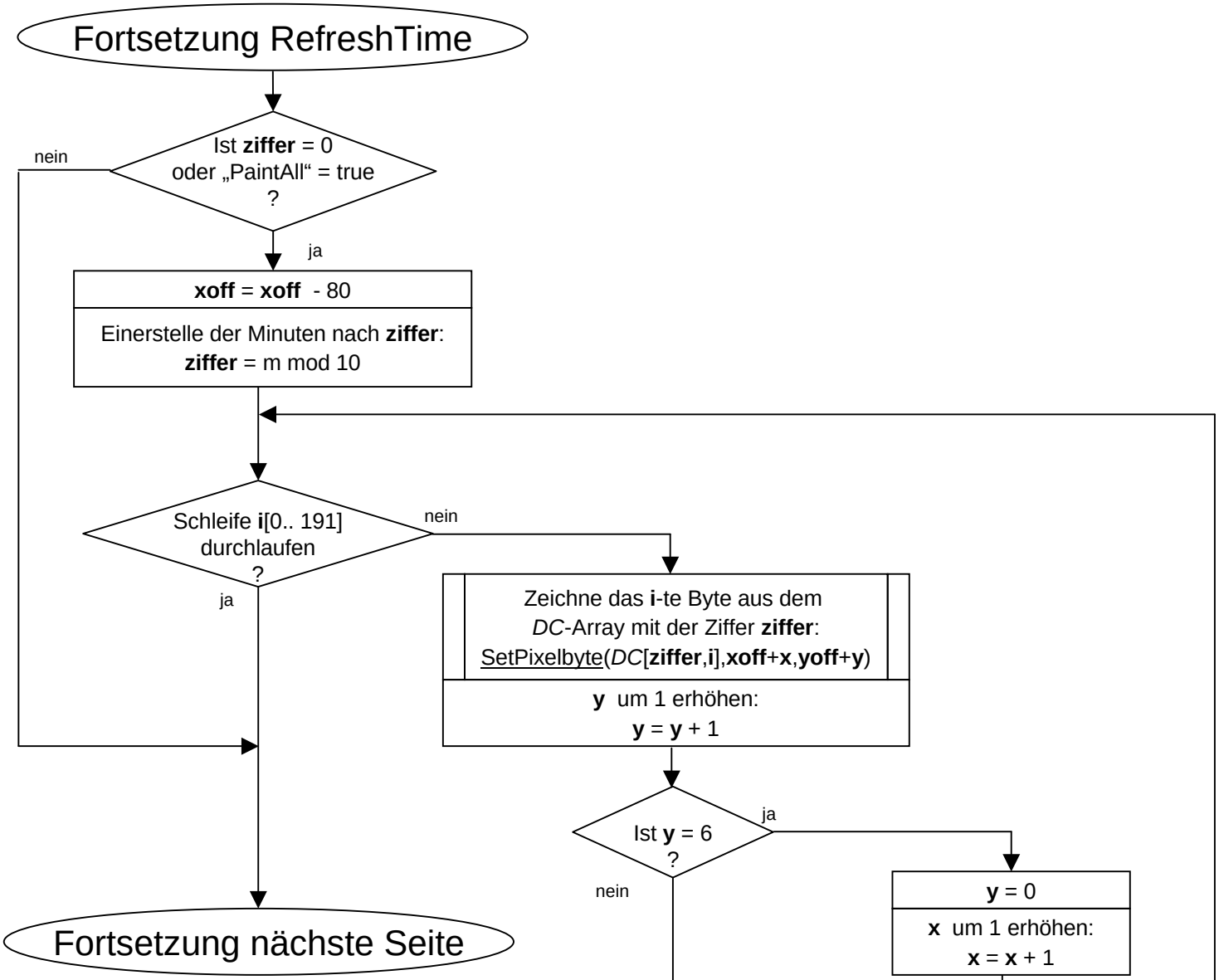


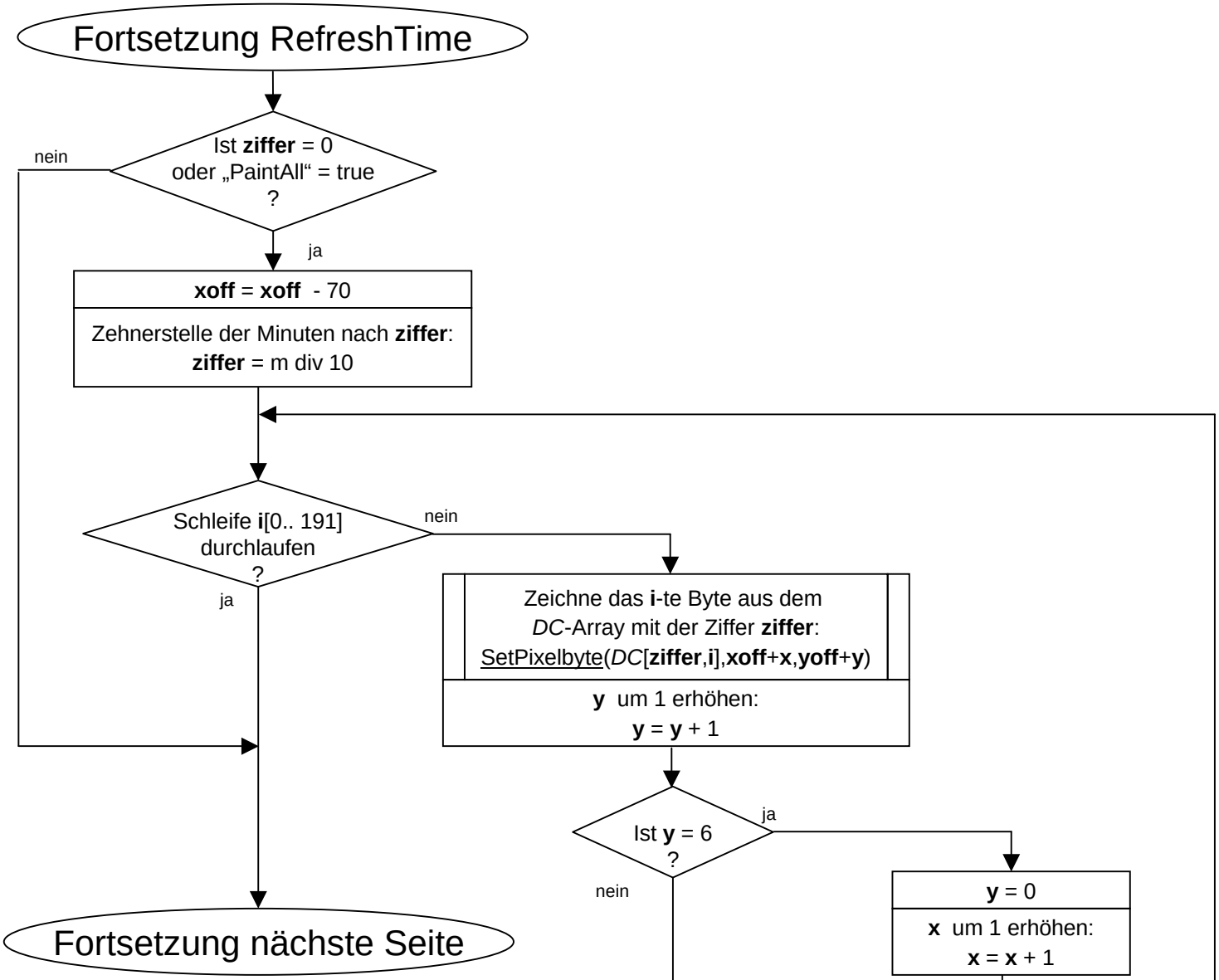
Prozedur TPortIOVFD.RefreshTime(PaintAll: boolean)

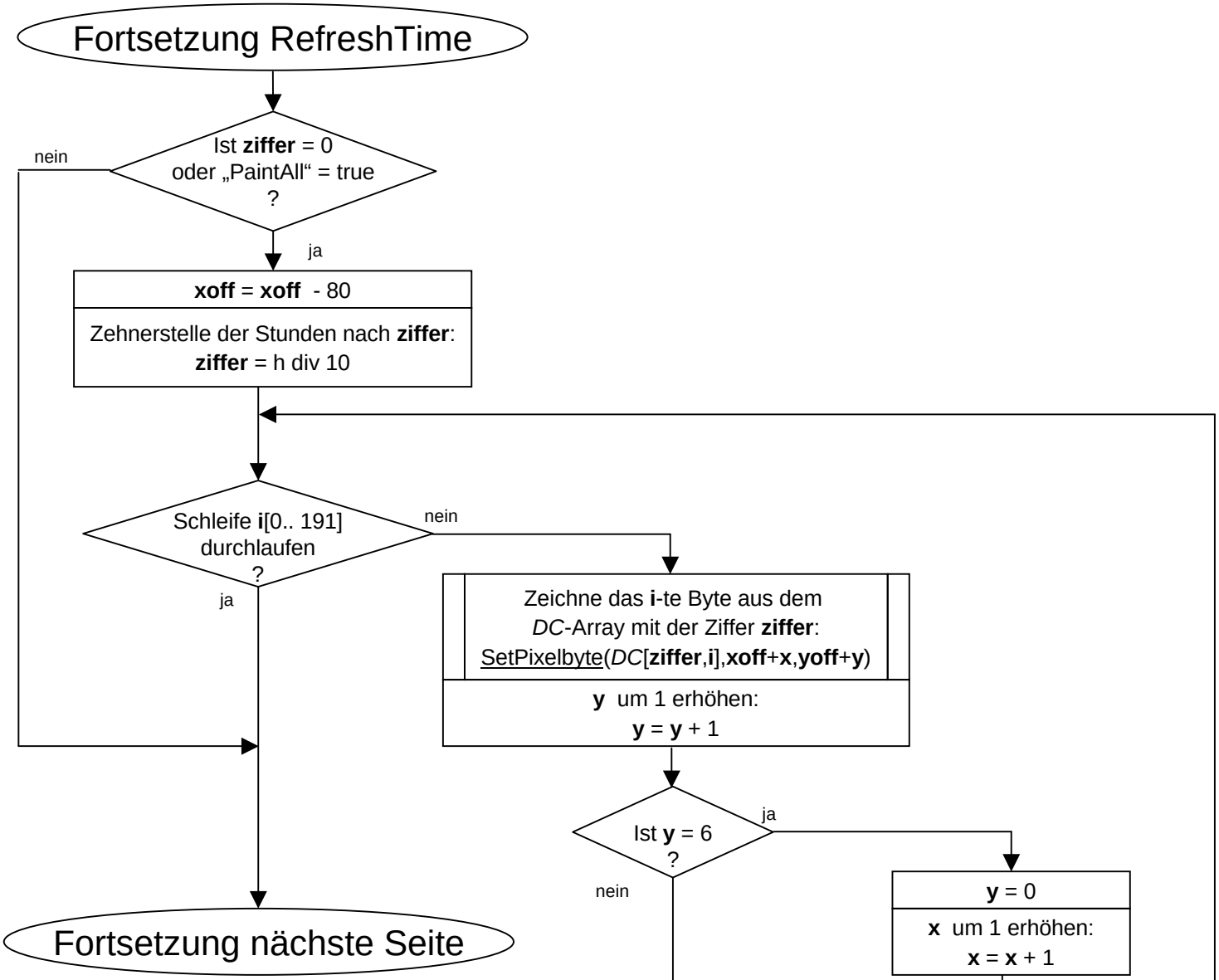
Die RefreshTime-Prozedur zeichnet die aktuelle Uhrzeit auf das Display. Dazu wird ein eigener Ziffernzeichensath verwendet, der in externen Dateien vorliegt und mit LoadChars in das DC-Array geladen wird.

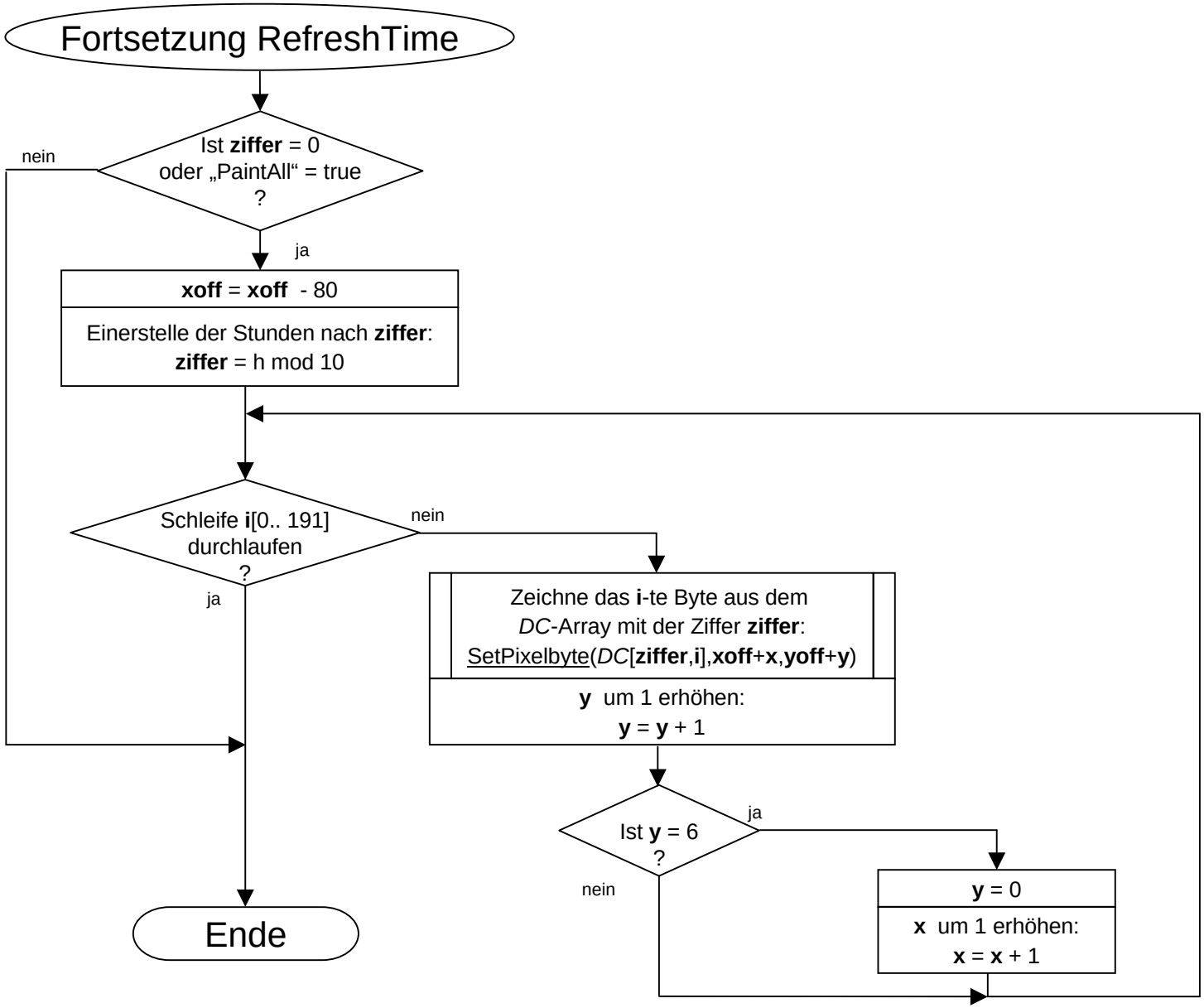



Fortsetzung RefreshTime
Nur wenn die letzte gezeichnete Ziffer (hier die Sekunden-Einerstelle) Null war, also von 9 auf 0 gewechselt ist, muss die nächsthöhere Stelle gezeichnet werden.
Es werden aber auch alle Stellen gezeichnet, wenn „PaintAll“ wahr ist – damit die volle Uhrzeit an Anfang einmal gezeichnet wird (s. Time-Prozedur).
Ist **ziffer** = 0 oder „PaintAll“ = true ?
nein
ja
xoff = **xoff** - 70
Zehnerstelle der Sekunden nach **ziffer**: **ziffer** = s div 10
Schleife **i**[0.. 191] durchlaufen ?
nein
ja
Zeichne das **i**-te Byte aus dem *DC*-Array mit der Ziffer **ziffer**: SetPixelbyte(*DC*[**ziffer**,**i**],**xoff**+**x**,**yoff**+**y**)
y um 1 erhöhen: **y** = **y** + 1
Ist **y** = 6 ?
ja
nein
y = 0
x um 1 erhöhen: **x** = **x** + 1
Fortsetzung nächste Seite


Fortsetzung RefreshTime
Ist **ziffer** = 0
oder „PaintAll“ = true
?
nein
ja
xoff = **xoff** - 80
Einerstelle der Minuten nach **ziffer**:
ziffer = m mod 10
Schleife **i**[0.. 191]
durchlaufen
?
nein
ja
Zeichne das **i**-te Byte aus dem
DC-Array mit der Ziffer **ziffer**:
SetPixelbyte(*DC*[**ziffer**,**i**],**xoff**+**x**,**yoff**+**y**)
y um 1 erhöhen:
y = **y** + 1
Ist **y** = 6
?
ja
nein
y = 0
x um 1 erhöhen:
x = **x** + 1
Fortsetzung nächste Seite

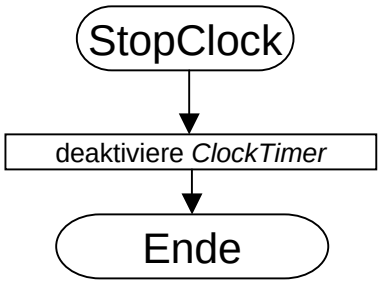

Fortsetzung RefreshTime
Ist **ziffer** = 0
oder „PaintAll“ = true
?
nein
ja
xoff = **xoff** - 70
Zehnerstelle der Minuten nach **ziffer**:
ziffer = m div 10
Schleife **i**[0.. 191]
durchlaufen
?
nein
ja
Zeichne das **i**-te Byte aus dem
DC-Array mit der Ziffer **ziffer**:
SetPixelbyte(*DC*[**ziffer**,**i**],**xoff**+**x**,**yoff**+**y**)
y um 1 erhöhen:
y = **y** + 1
Ist **y** = 6
?
ja
nein
y = 0
x um 1 erhöhen:
x = **x** + 1
Fortsetzung nächste Seite


Fortsetzung RefreshTime
Ist **ziffer** = 0
oder „PaintAll“ = true
?
nein
ja
xoff = **xoff** - 80
Zehnerstelle der Stunden nach **ziffer**:
ziffer = h div 10
Schleife **i**[0.. 191]
durchlaufen
?
nein
ja
Zeichne das **i**-te Byte aus dem
DC-Array mit der Ziffer **ziffer**:
SetPixelbyte(*DC*[**ziffer**,**i**],**xoff**+**x**,**yoff**+**y**)
y um 1 erhöhen:
y = **y** + 1
Ist **y** = 6
?
ja
nein
y = 0
x um 1 erhöhen:
x = **x** + 1
Fortsetzung nächste Seite


Fortsetzung RefreshTime
Ist **ziffer** = 0 oder „PaintAll“ = true ?
nein
ja
xoff = **xoff** - 80
Einerstelle der Stunden nach **ziffer**: **ziffer** = h mod 10
Schleife **i**[0.. 191] durchlaufen ?
nein
ja
Zeichne das **i**-te Byte aus dem *DC*-Array mit der Ziffer **ziffer**: SetPixelbyte(*DC*[**ziffer**,**i**],**xoff**+**x**,**yoff**+**y**)
y um 1 erhöhen: **y** = **y** + 1
Ist **y** = 6 ?
ja
nein
y = 0
x um 1 erhöhen: **x** = **x** + 1
Ende

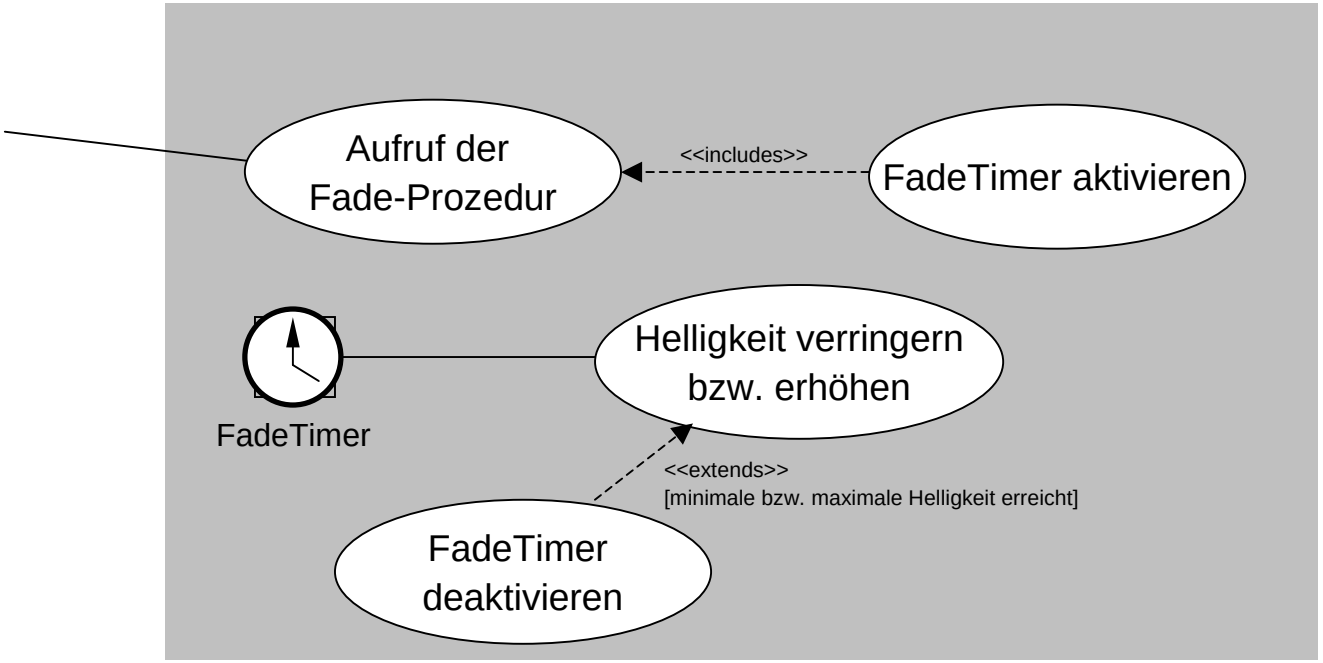
Prozedur TPortIOVFD.StopClock

Diese public-Prozedutr wird automatisch vom Hauptfenster aufgerufen sobald ein neuer Screen erstellt wird, d.h. wenn nicht mehr die Uhrzeit angezeigt werden soll sondern andere Informationen.
Andernfalls würde der ClockTimer aktiv bleiben und den neuen Screen wieder mit der Uhrzeit überschreiben.



Ein- und Ausblendeffekte

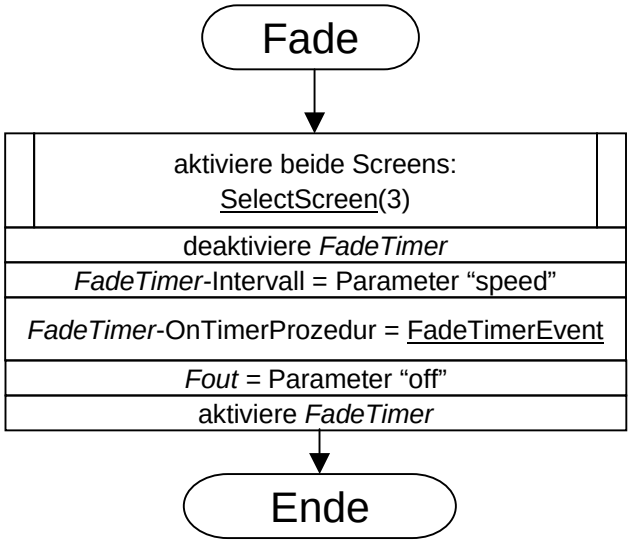
Anwendungsfall
Ein- oder Ausblenden



Die *Fade* Prozedur wird von außen (vom Hauptprogramm) aufgerufen und aktiviert den *FadeTimer*.
Dieser erhöht oder verringert bei jedem OnTimer-Event die Helligkeit des Displays.
Ob die Helligkeit erhöht oder verringert wird bestimmt die globale Variable *Fout*, die beim Aufruf der *Fade*-Prozedur auf ‚true' für Ausblenden, bzw. ‚false' für Einblenden gesetzt wird.
Ist das untere oder obere Ende der Helligkeitskala erreicht wird der FadeTimer deaktiviert.

Prozedur TPortIOVFD.Fade (off: boolean; Speed: word)

Mit dieser public Prozedur kann die Anzeige auf dem Display mit einstellbarer Geschwindigkeit ein- oder ausgeblendet werden.
Die Helligkeit wird dabei vom FadeTimer geregelt, dessen OnTimer-Event auf die Prozedur FadeTimerEvent verweist.



Prozedur TPortIOVFD.FadeTimerEvent (Sender: TObject)

Die FadeTimerEvent-Prozedur erhöht oder verringert je nach Fade-Richtung die globale LightLevel-Variable, und setzt die Helligkeit auf deren Wert.
Ist das obere oder untere Ende der Helligkeitsskala erreicht, schaltet sich wird der Timer deaktiviert.