



**MANUAL**  
**FLIGHT™ SDK**

---

**NEWSON ENGINEERING NV**

## Table of Contents

<b>1</b>	<b>GENERAL .....</b>	<b>4</b>
<b>2</b>	<b>SYSTEM REQUIREMENTS .....</b>	<b>5</b>
<b>3</b>	<b>REDISTRIBUTION .....</b>	<b>6</b>
<b>4</b>	<b>FLIGHT SDK: INTRODUCTION.....</b>	<b>7</b>
4.1	SYSTEM OVERVIEW .....	7
4.2	ACTIVE X TECHNOLOGY .....	7
4.3	RHOTHOR & FLIGHT TECHNOLOGY .....	7
4.4	FLIGHT SDK: STRUCTURE .....	7
<b>5</b>	<b>GETTING STARTED .....</b>	<b>9</b>
5.1	GENERAL.....	9
5.2	DATA TYPES AND HOLDERS.....	10
5.3	DATA REFERENCING .....	11
5.4	DATA STORAGE .....	12
5.5	DATA EDITING .....	13
5.6	DATA COMPILATION .....	13
5.7	DATA VISUALISATION .....	14
5.8	DATA MARKING .....	15
5.9	HARDWARE INTERFACE : COORDINATE SYSTEM.....	16
5.10	HARDWARE INTERFACE : LASER.....	16
5.11	HARDWARE INTERFACE : EXTRA .....	18
5.12	HARDWARE INTERFACE : DELAYS .....	18
<b>6</b>	<b>FLIGHT SOFTWARE DEVELOPMENT KIT REFERENCE.....</b>	<b>20</b>
6.1	GENERAL.....	20
6.2	FLIGHTOBJECT .....	20
6.3	FLIGHTTREE .....	23
6.4	FLIGHTCOLORS.....	27
6.5	FLIGHTCOLOR .....	30
6.6	FLIGHTPENS.....	31
6.7	FLIGHTPEN.....	33
6.8	FLIGHTFONTS .....	34
6.9	FLIGHTFONT .....	36
6.10	FLIGHTLAYOUTS .....	40
6.11	FLIGHTLAYOUT .....	42
6.12	FLIGHTVIEW .....	43
6.13	FLIGHTPROPLIST .....	46
6.14	FLIGHTEDITOR.....	47
6.15	FLIGHTMARKER .....	47
<b>7</b>	<b>APPENDIX: CHARACTER SET CODES.....</b>	<b>51</b>
<b>8</b>	<b>APPENDIX: BARCODE ECC 200 FONT SYMBOL SIZE.....</b>	<b>53</b>
<b>9</b>	<b>APPENDIX: READER RESPONSE.....</b>	<b>54</b>

#### TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your rhothor™ and ELEVAThor™ products. To this end we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new products and updates are introduced.

If you have any questions or comments regarding this publication, please contact Newson via E-mail at **info@newson.be** or fax the Reader Response Form in the back of this manual to +32 9 367 0693. We welcome your feedback.

Most Current Manual

(**NYR**) To obtain the most up-to-date version of this manual, please register at our Website at **<http://www.newson.be>**

You can determine the version of the manual on the bottom of any page.

# 1 GENERAL

Designed to easily integrate rhothor deflection technology into software environments, the flight SDK provides the software needed to create marking applications. With a set of ActiveX component the Software Development Kit offers a starters platform for integrators.

FLIGHT Software Development Kits are available with support for .NET, Visual Studio 6, 2002, 2003, 2005, 2008 for many languages as C#, C++, VB, VB.NET and many other development tools like Delphi, LabView...

Main features:

- build up laser marking software GUI
- import bitmap files comprising BMP, JPG, GIF
- import DXF vector files and HPGL plot files
- create text object using System fonts and Windows TrueType fonts
- create Code 39 and Code 39 Extended barcodes
- create 2D Datamatrix barcodes
- create a Pen set defining different vectoring processes
- create a Color set defining laser interface and marking process
- position, rotate and align objects in the marking field
- group objects and create step and repeat layouts

The purpose of this manual is to provide an overview of flight SDK and to detail the installation instructions. It is strongly recommended to read this manual before using.

## Legal Information

Information in this document is subject to change without notice.

FLIGHT and rhothor are trademarks of Newson Engineering NV.

ActiveX, Microsoft, Visual Basic, Visual C++, Visual Studio, Windows, Windows XP, Windows 2000 and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Macintosh and TrueType are registered trademarks of Apple Computer, Inc.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

## 2 SYSTEM REQUIREMENTS

### ***Deflection system***

The flight SDK supports rhothor laser deflection heads using the flight interface mode. It uses the flight DLL and rhothor DLL to communicate with the deflection head over USB.

### ***PC Hardware***

- IBM PC or compatible with at least a Pentium-class processor (120 MHz or better).
- Minimum system memory of 64 Mb. Additional memory may be necessary to support the development platform and the embedded application.
- Minimum hard drive space required: 20 MB. Marking files may require additional harddisk space.
- An available CD-ROM drive for installation.
- Graphics hardware supporting OpenGL.
- A free USB slot to connect the PC with the rhothor deflection system.

### ***Operating Systems***

Flight Software Development Kit requires one of the following operating systems

- Windows 2000
- Windows XP 32 bit
- Windows Vista 32 bit

### ***Software***

The Flight Software Development Kit needs at least the .NET Framework 2.0 in order to run.

### ***Development Platforms Supported***

Application can be developed using any platform that supports the Microsoft Component Object Model (COM).

Newson has developed the Demo applications using Visual Basic .NET from Visual Studio 2008.

### ***Skill required***

Programmers embedding the flight ActiveX components in their applications should be proficient in Windows programming on chosen development platform and know the interfacing of the development platform with ActiveX components.

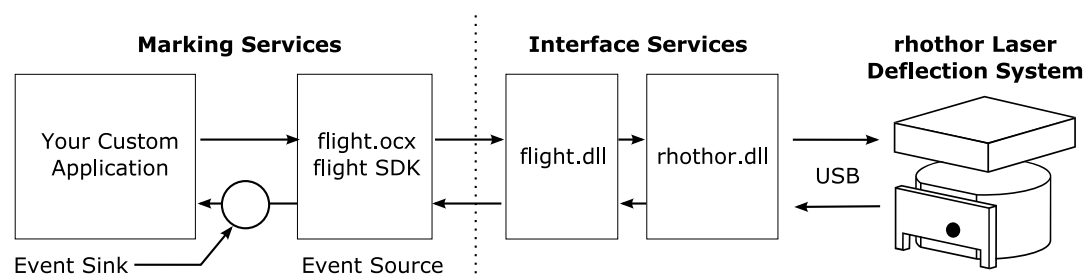
### 3 REDISTRIBUTION

1. Since ActiveX controls are dynamically linked, the ActiveX controls must be distributed with the applications that use them.
2. At installation of the application, the flight.ocx must be registered. This can be done with Regsvr32.exe.  
regsvr32 /s <path to flight.ocx>
3. Applications built with the flight SDK require a number of supporting dynamic-link libraries (DLLs). The software development kit uses the windows MFC DLLs and requires also the following redistributable files:
  - flight.ocx                      the flight ActiveX control
  - rhothor.dll                  hardware interface DLL
  - flight.dll                      flight interface DLL
  - nwsn0001.dll                  DXF/HPGL import DLL
  - nwsn0002.dll                  bitmap import DLL

## 4 FLIGHT SDK: INTRODUCTION

### 4.1 SYSTEM OVERVIEW

The flight Software Development Kit is a set of ActiveX components for the development of custom laser marking applications. ActiveX controls, formerly known as OLE controls are based on the Common Object Model (COM). Marking services use the flight DLL to interface with the deflection head over USB. The diagram below shows how these interact.



Using the ActiveX object available for laser marking services, you can write your program to control the deflection system and its actions. Embedded laser applications may be developed in a variety of languages both compiled and interpreted. Supported languages include Visual C++, Visual Basic, C#, VB.NET and VBA.

### 4.2 ACTIVEX TECHNOLOGY

ActiveX is a marketing name for windows third generation OLE technology. It is a framework that allows software components to co-operate even when they are written by different vendors, using different tools and different languages. Underneath ActiveX is the generic Common Object Model that defines the binary interface between objects. An ActiveX control is a software component that gives a specific item of functionality to a parent application. It exposes properties and methods and can fire events. The flight SDK provides a set of these components to provide the parent application laser marking functionality.

### 4.3 RHOTHOR & FLIGHT TECHNOLOGY

The rhothor deflection systems can work without additional controller card. The communication runs over the rhothor.dll. This dll provides a low level set of commandos to control tracking and laser. For more information read the rhothor manual.

The FLIGHT SDK is build on top of this rhothor.dll providing a higher level language to create marking images and mark them using rhothor deflection systems. Its purpose is to provide integrators an easy to use toolset to integrate the rhothor technology into laser machines. The SDK comes with an open source example that can be customized.

### 4.4 FLIGHT SDK: STRUCTURE

The flight Software Development Kit is build out of different logical blocks, which work together to provide a complete marking solution. Every logical block is represented by a set of ActiveX controls.

The core structure is a master-slave structure, where one master object controls interaction between all others. All logical blocks are build around this master object

The toolkit consist out of following controls:

FlightMarker	the master core component, controls interaction between all other flight controls and represents the rhothor deflection system.
--------------	---

FlightColors	a set of laser processing parameters. This control defines the interface to the laser and is implemented as a ComboBox.
FlightPens	a set of pen parameters representing the vectoring process of data. A pin will define how data will be transformed into information that can be marked by the deflection head. This control is implemented as a ComboBox.
FlightFonts	a set of different fonts representing the possible fonts to mark text objects and comprising Windows TrueType fonts, System fonts and barcode fonts. This control is implemented as a ComboBox.
FlightLayouts	a set of different layouts defining marking layouts, including step and repeat structures. This control is implemented as a ComboBox.
FlightTree	a hierarchical list of FlightObjects representing the vector data. Each FlightObject defines a markable object or represents a group of other FlightObjects.
FlightView	a window that displays the marking field and all markable objects on it.
FlightPropList	encapsulates the functionality of a property list control displaying a collection of properties of an object and providing the ability to change them.
FlightEditor	provides a CAD system with basic drawing functionality to create or edit vector structures that can be marked.



## 5 GETTING STARTED

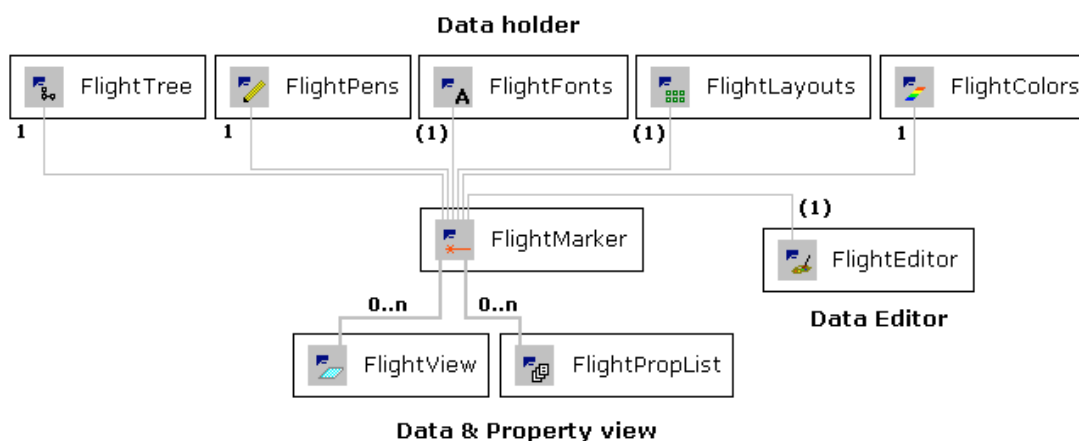
### 5.1 GENERAL

As already mentioned the flight SDK is a set of ActiveX components in a master slave structure. In order to create a marking solution a project needs at least:

- 1 FlightMarker object, this is the master object
- 1 FlightPens object, this set of Pens will define the vectoring process
- 1 FlightColors object, this set of Colors will define the laser parameters
- 1 FlightTree object, this Tree will contain all markable entities.

This can be extended with:

- 1 FlightFonts object, this set of Fonts will provide the available fonts in order to use text objects.
- 1 FlightLayouts objects, this set of Layouts will provide step and repeat structures in order to use group objects
- 1 FlightEditor object, this object can show a basic CAD/CAM program in order to edit existing structure files.
- A number of FlightView objects, who represent the marking field with the mark able entities on it.
- A number of FlightPropList objects, who are able to show properties and provide functionality to change these.



Every Flight ActiveX control should be attached to the master:

#### VB.Net: Attach Flight ActiveX controls

With FlightMarker

```

.Attach(FlightColors)
.Attach(FlightPens)
.Attach(FlightFonts)
.Attach(FlightLayouts)
.Attach(FlightTree)
.Attach(FlightPropList1)
.Attach(FlightPropList2)
.Attach(FlightView)
.Attach(FlightEditor)

```

End Width

## 5.2 DATA TYPES AND HOLDERS

The purpose of a laser marking software program is to create a vector image that can be marked using a deflection head. This process can be split up in

- Adding data
- Compiling data into vector format
- Mark vector format data with certain process parameters.

This complete process is controlled by some parameters which are stored in the SDK controls.

- **FlightPens:** Vectoring parameters, define how the data is compiled..
- **FlightColors:** Laser process parameters, define how the vector objects are lasered.
- **FlightLayouts:** Layout parameters, define how the same vector objects are marked on different places.
- **FlightFonts:** Fonts, define which font sets can be used to create text objects and barcodes.

These parameters can be accessed and modified through the methods and properties from the controls.

### VB.Net: Create and initialise 6 Pens

```
Private Sub Init_Pens()
    For i = 0 to 5
        FlightPens.AddPen()
        With FlightPens.get_Data(i)
            .HatchPitch = 0.1
            .HatchStyle = 1
            .Outlines = 1
            .Pitch = 0.1 * i
            .Optimize = True
        End With
    Next i
    FlightPens.ListIndex = 0
End Sub
```

Besides these parameters the SDK also supports Vector objects. Vector objects are entities that can be marked with a laser. They include bitmap, structure and text objects. Vector objects use a FlightPen object to compile their source into vector data. There are different object types:

#### • *Bitmap objects*

Bitmap files with bmp, jpg, gif or png extension can be imported. The flight SDK will transform the image into a black and white image and then the vectoring process searches the outlines of the shapes. Hatching is also possible.

#### • *Struct objects*

The SDK can import files DXF format 9. DXF stands for Drawing Exchange Format and is developed by AutoDesk to exchange data between AutoCAD and other applications. Import includes Line, Arc, Circle and Polyline. When Points are available in the DXF the first one will be used as reference.

Besides DXF, the Flight Software Development Kit also support HPGL plot files, HPGL stands for Hewlett-Packard Graphics Language and is a standard for almost all plotters. The Flight SDK understands most of the command set of HPGL and some commands of the HPGL/2.

#### • *Text objects*

Fonts imported as document objects can be used to create text objects. Fonts

include TrueType fonts, Single Stroke fonts and barcodes. Barcodes include Code 39, Code 39 extended and 2D Datamatrix barcode.

Datamatrix is a two-dimensional matrix code consisting of black and white cells build up in a square pattern. The flight SDK support ECC200 Data Matrix barcodes which supports advanced encoding error checking and correction algorithms. It allows reconstruction of the data string when the symbol sustained 30% damage.

Code 39 is a one-dimensional barcode that can encode uppercase letters, digits and some special characters. Code 39 is restricted to 44 characters. Code 39 Extended allows to encode data with the Full ASCII set. A check digit is not often used but a few critical applications require one. The check digit is a modulus 43 sum of all character values in the encoded message.

#### • Group objects

Different objects can be grouped using group objects. Each group has also a layout definition that defines the positioning of the group. In this way the elements in the group can be repeated on different places. Use the layout to create “step and repeat” marking images. Groups in groups are not possible.

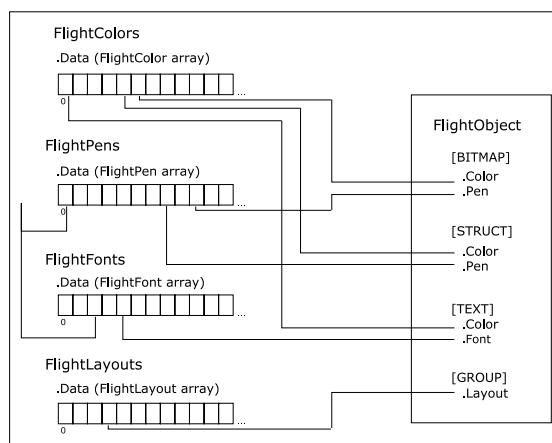
The vector objects are embedded in a tree control, a FlightTree component. It represents all objects as a hierarchical list of items. A FlightObject object represents a single vector object. It is possible to retrieve the objects from the FlightTree control using the GetFlightObject method. To insert a new vector object you first create a FlightObject and then insert it into the tree.

#### VB.Net: Insert vector struct object

```
Dim obj As FlightObject = New FlightObject
If obj.CreateStruct("") Then
    FlightTree.InsertObject(obj, FlightConstants.ftvi_Root, FlightConstants.ftvi_Last)
    FlightMarker.UpdateData()
End if
```

## 5.3 DATA REFERENCING

The data is stored locally inside the different data holder components. The different data holders are linked to each other with unique numbers representing the index of the objects inside an embedded array. For example FlightPens has an internal array of FlightPen objects. A FlightObject can have a reference to one of the FlightPen objects by using the index of its position inside the array.



- Bitmap, Struct and Text objects have a reference to a Color, representing the laser parameters used to mark the object.
- Bitmap and Struct objects use a Pen to define the vectoring process.
- Text objects have a reference to a Font, representing the character set used to create a Text object. The Font itself has a reference to a Pen defining the vectoring process of the Font's character set.
- Group objects have a reference to a Layout, which determines the physical layout.

**Remark:** It's the job of the programmer to keep the data references valid.

## 5.4 DATA STORAGE

The data embedded in the data holder components can be stored to or loaded from a file. The flight SDK stores the data in an XML file format. XML stands for eXtensible Markup Language, a standard developed language for transferring structured data on the Web. An XML document has a tree structure and enables the possibility to store the data into separate files or in one global file. One of the possible implementations would be to create a laser parameter database on a remote server and load the color sets into the application using an internet connection. In the open source example everything is stored in the same file.

### VB.Net: Store XML flight objects

```
FlightPens.Store(FileName)
FlightColors.Store(FileName)
FlightFonts.Store(FileName)
FlightLayouts.Store(FileName)
FlightTree.Store(FileName)
```

### VB.Net: Load XML flight objects

```
FlightPens.Load(FileName)
FlightColors.Load(FileName)
FlightFonts.Load(FileName)
FlightLayouts.Load(FileName)
FlightTree.Load(FileName)
```

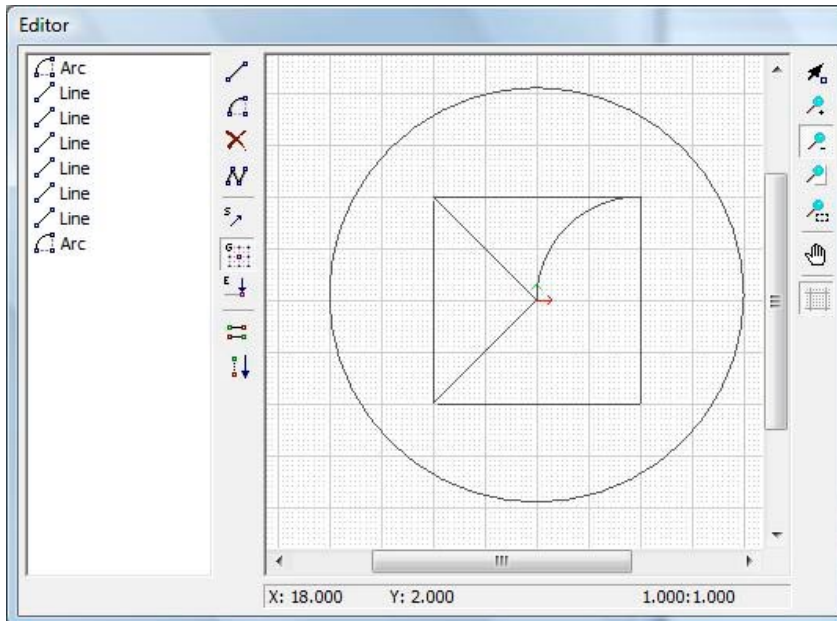
It's even possible to extent the flight XML data structure by creating your own subtree under the XML flight root.

XML files can be opened in Notepad or Internet Explorer to see the content.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <Flight Version="1">
+ <Objects Version="1">
- <ColorData Version="1">
+ <Color id="0">
  </ColorData>
- <PenData Version="2">
  - <Pen id="0">
    <Outlines>1</Outlines>
    <Pitch>0.100</Pitch>
    <HatchStyle>0</HatchStyle>
    <HatchPitch>0.100</HatchPitch>
    <Optimize>1</Optimize>
  </Pen>
  </PenData>
- <FontData Version="2">
  - <Font id="0" Type="Truetype">
    <FaceName>Arial</FaceName>
    <Height>2.000</Height>
    <LineSpacing>2.000</LineSpacing>
    <Style>0</Style>
    <CharSet>0</CharSet>
    <Pen>0</Pen>
  </Font>
  </FontData>
+ <LayoutData Version="2">
</Flight>
```

## 5.5 DATA EDITING

The flight SDK provides a simple CAD/CAM editor which allows the user to create simple drawing or to modify imported vector drawings (DXF / HPGL files). The editor is implemented as a tool dialog that can be shown to the user.



Main functionality:

- Draw Lines, Polylines and Arcs
- Snap to grid or end points
- Sort drawing objects manually or use software sort algorithm
- Swap begin and endpoints
- Reposition vectors
- Turn grid on/off
- Zoom and pan

Once the FlightEditor object is inserted and attached to a FlightMarker you can Edit a FlightObject by calling the Edit function of the FlightEditor object. Use the method ShowEditor and HideEditor to swap the visibility of the dialog. Double Click in the Title bar to maximize and restore the view. Only FlightObjects of type struct can be edited.

### **VB.Net: Edit currently selected struct object**

```
Editor.Edit(FlightTree.GetFlightObject(FlightTree.GetSelectedItem()))
Editor.ShowEditor()
```

After the user has edited the FlightObject a call to FlightMarker.UpdateData will recompile the data and reflect the changes.

## 5.6 DATA COMPILATION

Rhothor deflection heads can mark lines, arcs and execute burst commands on a certain position. The transformation process of source data into vector code that can be marked by the deflection head is called compilation. When the compilation process needs to transform raster data into vector data, the property FlightMarker.Resolution is used to determine the minimal vector length. This global property can be a value in the range 0.001 to 0.050 and should remain constant. A smaller resolution value will result in a more precise compilation but will also take a longer compile time and will need more resources.

Setting compilation process parameter in the flight SDK is done by modifying the pen properties. This pen contains all parameters that determine how the data source code is turned into vector data. An object can be vectorised in two ways by generating outlines and by creating hatching pattern.

The parameters of a pen are:

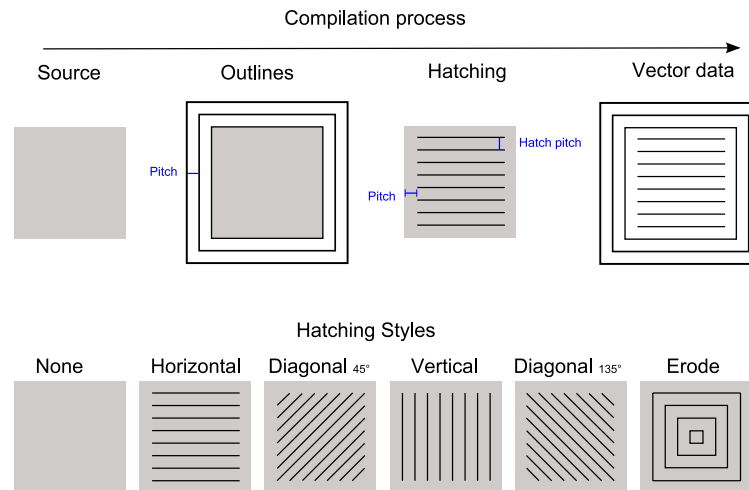
Outlines: Number of outlines;

Pitch: Distance between two adjacent outlines, expressed in mm;

HatchStyle: Defines the hatching pattern;

HatchPitch: Distance between two adjacent hatch lines, expressed in mm;

Optimise: Determines whether the vector data should be resorted or if the original order should be kept.



Remark: Through a FlightEditor object a user can change the order of the vector data for FlightObjects of the type struct. Keep in mind that when this struct object has a pen where the Optimise property is set on TRUE, this order will be overwritten during compilation process.

Depending on the FlightObject type some pen properties are ignored/unused. On struct objects hatching is not possible.

Type	Outlines	Pitch	HatchStyle	HatchPitch	Optimise
Bitmap object	x	x	x	x	x
Struct object	x	x			x
Text object with True Type Font	x	x	x	x	x
Text object with Single Stroke Font					
Text object with Code39 (Extended) Barcode Font				x	
Text object with Datamatrix Barcode Font	x	x	x	x	

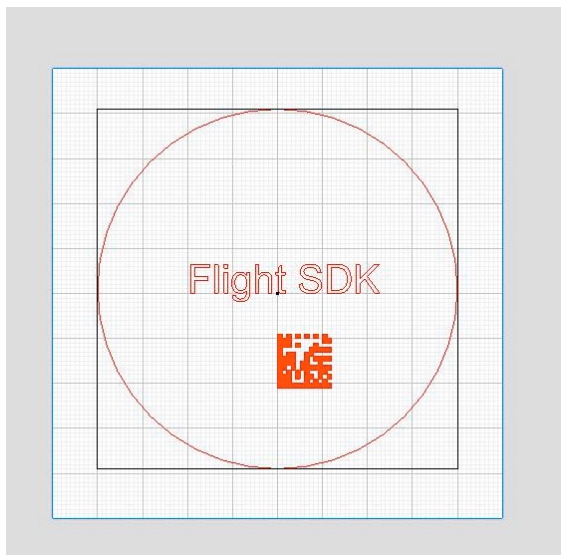
Whenever you change properties you will need to recompile in order to see the new vector data. Recompile can be done at any time by calling the function UpdateData from the FlightMarker object.

## 5.7 DATA VISUALISATION

The FlightView object provides visualisation of the marking field with markable entities of the current job. Every FlightObject from the FlightTree will be shown with appropriate color. This is the color associated with the FlightColor object where the FlightObject references to.

The bounding box shows the marking field, which should be set depending on the deflection system. It can be changed by modifying the FieldWidth And FieldHeight property of the FlightView object.

The FlightView ActiveX control provides build in functionality to Zoom, Pan and scroll the view. Change the ViewMode property to change the view mode.



FlightObject, FlightColor, FlightFont, FlightLayout and FlightPen objects can show their properties to the user using a FlightPropList ActiveX control. To do so change the PropSrc member of the FlightPropList control.

*' Show the properties of the currently selected pen*

```
FlightPropList.PropSrc = FlightPens.get_Data(FlightPens.ListIndex)
```

*'*

*Show the properties of the currently selected flight object*


```
FlightPropList.PropSrc = FlightTree.GetFlightObject(FlightTree.GetSelectedItem)
```

The FlightPropList control allows the user to change the properties. Same is possible through scripting. To reflect the changes to the vector data itself, call the UpdateData method from the master FlightMarker object.

*' Change the properties of the currently selected color*

```
FlightPens.get_Data(FlightPens.ListIndex).Outlines = 2
```

Call FlightMarker.UpdateData

Outlines	1
Pitch	0.100
Hatch style	 Horizontal
Hatch pitch	0.100
Optimize	<input checked="" type="checkbox"/> True

## 5.8 DATA MARKING

### 5.8.1 Calibration

Deflection systems suffer from positional errors resulting from different sources like f-theta distortions, pin cushion distortion, alignment problems... The rhotor deflection head offer a solution for all those error sources through a calibration algorithm that is based on lookup tables.

Those calibration lookup tables reside in the memory in the deflection system, which is volatile. Therefore after each power up these tables have to be reloaded into the memory of the deflection head. The LoadCalibrationFile function from the FlightMarker will load a calibration file from the hard disk into the deflection head.

For more information about calibrating the deflection head and generating the calibration files, read the rhothor manual.

## 5.8.2 Mark

When the data is compiled, and laser process parameters are setup through the FlightColors object, it can be marked using the Mark method from the FlightMarker Object. The SDK will search for the deflection head defined by the Serial property of the FlightMarker object. When this value is -1 the flight SDK will open the first available deflection head. The ActiveX component will fire an event when the marking is finished.

Remark: Deflection heads don't allow multiple owners. This means that using the rhothor.exe software program and marking with the flight SDK are mutual exclusive.

## 5.9 HARDWARE INTERFACE : COORDINATE SYSTEM

The flight coordinate system is a Cartesian coordinate system. Default positive X coordinates are located right of the origin and positive Y coordinates are located above the origin. The origin is located in the center of the marking field.

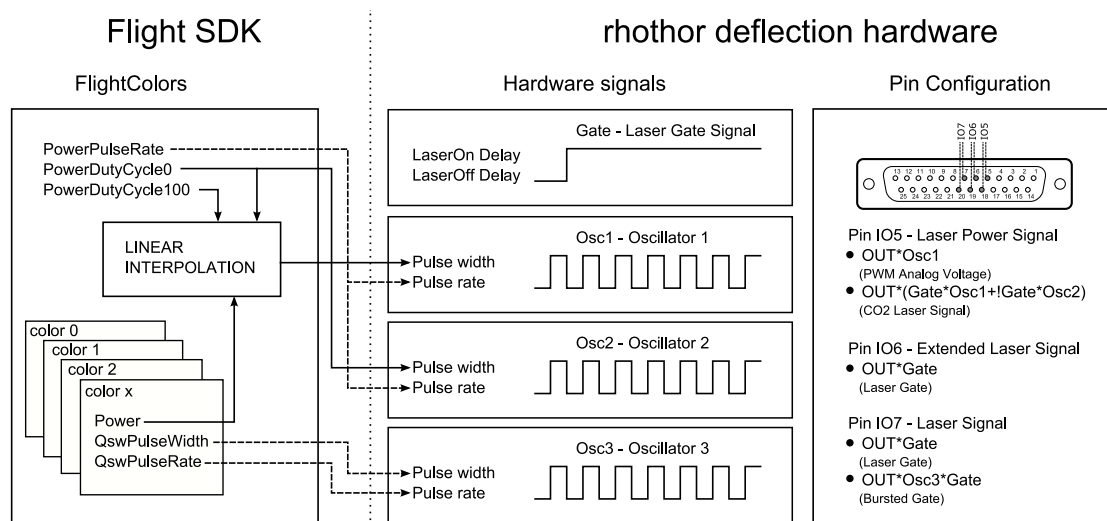
The hardware coordinate system can be changed using the rhothor.exe software. Axis can be swapped and mirrored.

## 5.10 HARDWARE INTERFACE : LASER

rhothor deflection head can interface with nearly any kind of laser. The triggering of the laser is synchronized with steering of the motors. To interface with a laser, IO pins 5,6 and 7 are being used. Those provide RS485 level signals.

The deflection head itself contains three internal oscillators and generates a gate signal. Through these oscillators, complex laser signals can be generated combining one or two oscillators and the gate signal.

Next scheme gives a global overview about how the flight SDK interfaces with the three oscillators to provide signals for power and gate. Setting up parameters for the laser are done through a FlightColors object which is basically an array of laser parameters containing local parameters together with some global parameters. Those parameters will be used to setup the three oscillators.





Local parameters QswPulseRate and QswPulseWidth define Osc3. Local parameter Power is being linear interpolated between PowerPulseRate0 and PowerPulseRate100 to determine the actual duty cycle of Osc2. PowerPulseRate0 also defines the duty cycle of Osc1.

Both Osc1 and Osc2 have a pulse rate defined by global parameter PowerPulseRate.

Example linear interpolation

FlightColors.PowerPulseRate = 50000 '50 kHz

FlightColors.PowerDutycycle0 = 10 '10%

FlightColors.PowerDutycycle100 = 100 '100%

A Color with Power = 50% will generate an actual duty cycle of 55%, meaning a pulse width of 11  $\mu$ s. A power of 0% will generate an actual duty cycle of 10%, meaning a pulse width of 2  $\mu$ s.

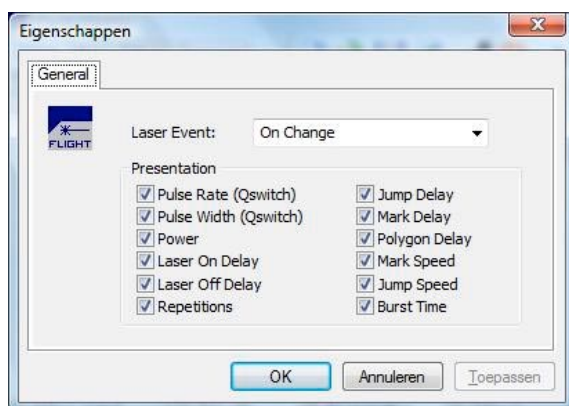
Remark: Duty cycle is defined as the ration between the pulse duration and the period of a rectangular waveform. For example, or a pulse train in which the pulse duration is 1  $\mu$ s and the pulse period is 4  $\mu$ s, the duty cycle is 25%.

Now a combination of these Oscillator signals and Gate signal can be used to generate signals for the laser. Selection can be done using the rhorhor.exe program (see rhorhor manual)

- On a laser with analog power control the mode OUT\*Osc1 on Pin IO5 can be used to generate an analog voltage using an external lowpass filter or for a CO<sub>2</sub> laser with no seed pulse. Signal will have a pulse rate define by the PowerPulseRate parameter and an output level defined by the Power parameter.
- On a CO<sub>2</sub> Laser Pin IO5 is set to OUT\*(Gate\*Osc1+!Gate\*Osc2) creating a gate controlled switch between DutyCycle0 and the DutyCycle defined by the Power parameter.
- A Qswitch controlled YAG laser can use Pin IO7 to control the Qswitch. Set Pin IO7 to OUT\*Gate\*Osc3 for a burst gate signal or choose OUT\*Gate for a standard Gate signal.
- Pin IO6 can also be configured as additional Gate signal.

#### REMARKS

- Some Laser types (eg. most fiber lasers) have an external interface like a serial interface or DLL software interface. For those interfaces it is possible to generate a software event during the marking to do custom interfacing. To enable these events change the LaserEvent parameter to 1 or 2. 1 - Laser events are generated by the FlightMarker object whenever the laser parameters change / 2 - Laser events are generated by the FlightMarker object whenever an entity is marked. The laser event itself will provide a structure which contains all changeable parameters, these are parameters QswPulseRate, QswPulseWidth and the actual power duty cycle defined by Power.
- All Laser interfaces on the IO pins have a combination of the form "OUT\*" which mean they are depending on the virtual output state of the pin. This means you have to set the output states of the pins to high in order to enable the laser interface. This can be done by using the method SetIO from the FlightMarker object. Eg. "FlightMarker.SetIO(112, 112)" enables the laser interfaces of the pins 5,6 and 7. Setting the IO's to low can be useful when generating a marking simulation.
- Depending on the laser type some parameters are unused. Using the Presentation parameter of the FlightColors object it is possible to prevent these properties from being showed to the user in the FlightPropList objects. Hidden properties will still be stored in the XML file.
- Presentation and Laser parameter can easily been changed in your software development by showing the ActiveX - Properties of the FlightColors component.



## 5.11 HARDWARE INTERFACE : EXTRA

The FlightMarker object who provides the hardware interface in the Flight SDK has some extra functionality to enable extra features:

- For Laser power mapping:
  - MoveTo(X, Y, Z): move the deflection head to a certain position.
  - SelectColor(Color): setup the laser parameters of one of the colors.
  - SetLaser(LaserOn): change laser mode between continuously on / gate driven.
- For Calibration
  - LoadCalibrationFile(FileName): Load calibration table from a file into the deflection head. For more information about calibration read the rhothor.exe manual.
- For Input/Output
  - SetIO(Value, Mask) Set the output state of some pins.
  - GetIO() Read the state of the pins.

Besides the flight SDK, rhothor.exe provides some extra features which are transparent to the Flight SDK. They include speed frequency modulation, speed pulse width modulation and position pulse width modulation. For more information read the rhothor.exe manual.

## 5.12 HARDWARE INTERFACE : DELAYS

For high speed marking some considerations have to be made. A deflection head needs to be given some time to accelerate. This drag error, the time the deflection head lags behind the command stream, causes unwanted effects in the marking. To compensate, the flight SDK provides some delays which give the ability to create high speed markings with good quality.

These delays must be matched to the marking and jump speed and can be set through the FlightColors component.

### VB.Net: Setting up delays through color 0

With FlightColors.get\_Data(0)

```
.JumpDelay = 80
.MarkDelay = 80
.PolygonDelay = 80
.LaserOnDelay = 160
.LaserOffDelay = 160
```

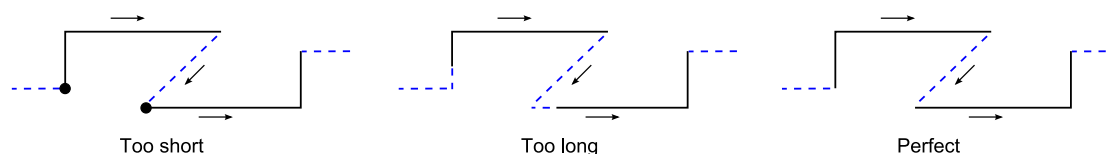
End With

When the deflection head starts marking a vector list. The laser is turned on before the deflection head has reached the angular velocity. This can lead to burn-in effects at the start points of the different vectors. With LaserOn delay the

moment when the laser is turned on is shifted in time. Therefore it is possible to prevent these burn-in effects. LaserOn delay can also compensate for lasers having a laser delay.

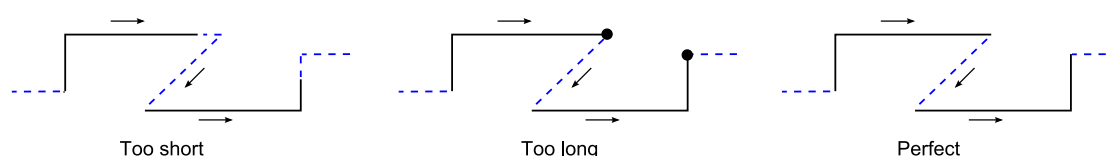
When LaserOn delay is set too short (can also be negative), burn-in effects will be shown at the beginning of the vector lists. When LaserOn is set too long the beginning of the vector lists will not be marked.

### LaserOn Delay



Similar to the LaserOn delay, the LaserOff delay shifts the moment when the laser is turned off at the end of the marking (of a vector list). When LaserOff delay is set too short (can also be negative), the end of the vector lists will not be marked. Setting the LaserOff delay too long will cause burn-in effects.

### LaserOff Delay



LaserOn and LaserOff delay do not extent the scanning time but only the marking quality. Besides these laser delays there are also some other delays to improve quality.

- Jump Delay:

After the jump to the start of the marking (beginning of a vector list), the mirrors of the deflection head are not settled yet when the marking starts. An oscillation will be visible. Jump delay inserts a delay till the mirrors are settled before starting the marking. If jump delay is set too high, only the marking time will increase.

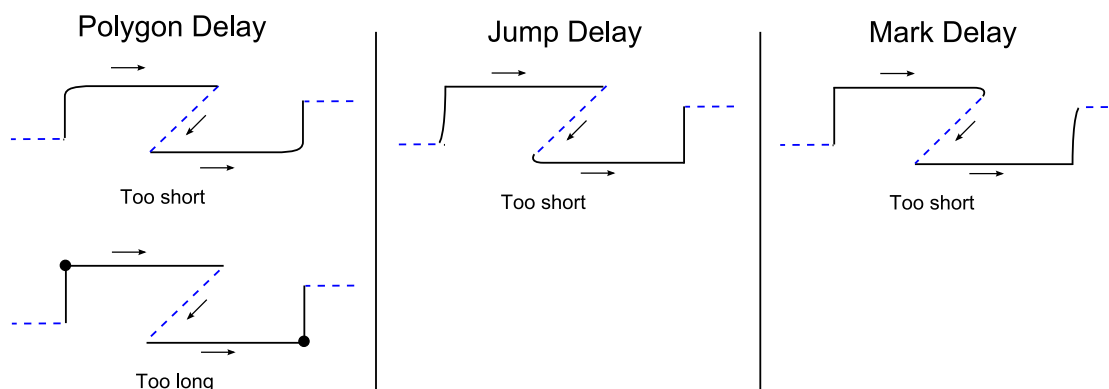
- Mark Delay:

After the end of a vector list the mirrors will not have reached end position yet when the next jump is already executed, giving the vector list a "tail". Mark delay inserts a delay so that the mirrors have time to reach the end position, before executing the next jump. If mark delay is set too high, only the marking time will increase.

- Polygon Delay

Polygon delay inserts a delay when the vector list contains a direction change. A too long polygon delay will result in burn-in effects at these points. Polygon delay improves the sharpness of the marking.

(Remark: Polygon delay is only implemented for struct oriented objects. On bitmap oriented objects like bitmap objects, text objects with TrueType fonts or struct objects with multiple outlines this parameter is ignored!)



## 6 FLIGHT SOFTWARE DEVELOPMENT KIT REFERENCE

### 6.1 GENERAL

This section gives an overview of the possible Flight ActiveX controls with their methods, properties and events. Some properties are read-only; for other properties a value must be set. In the overview tables this is indicated in column Access Mode.

### 6.2 FLIGHTOBJECT

This object represents a vector entity and comprises the marking data.

#### 6.2.1 Properties Overview

Property	Access Mode	Description	Type	Range
Caption	Value	Sets/Gets the caption of the object.	BSTR	
Color	Value	Sets/Gets the reference to the color defining the laser parameters of the object, i.e. the index of the color in the color data array of a FlightColors object.	LONG 32 bit	
Enabled	Value	Sets/Gets whether the object is enabled and marked.	VARIANT_BOOL 16 bit	Boolean
Font	Value	Sets/Gets the reference to the font used by a text object, i.e. the index of the font in the fonts data array of a FlightFonts object	LONG 32 bit	
Height	Value	Sets/Gets the height of the object. [mm]	DOUBLE 64 bit	
ImageMode	Value	Sets/Gets the image mode of the object.	SHORT 16 bit	0,1 (*3)
Handle	Read-only	Gets the handle of the object.	LONG 32 bit	

Layout	Value	Sets/Gets the reference to the layout used by a group object, i.e. the index of the layout in the layout data array of a FlightLayouts object.	LONG 32 bit	
Pen	Value	Sets/Gets the reference to the pen used during vectoring proces, i.e. the index of the pen in the pens data array of a FlightPens object	LONG 32 bit	
Reference	Value	Sets/Gets the reference of the object.	SHORT 16 bit	0,1,2,3,4 (*2)
Rotation	Value	Sets/Gets the rotation of the object around the object's reference position (X,Y). CCW = positive value. [°]	DOUBLE 64 bit	
Scale	Value	Sets/Gets the scale factor of the object.	DOUBLE 64 bit	
Type	Read-only	Gets the type of the object.	LONG 32 bit	0,1,2,3,4 (*1)
Width	Value	Sets/Gets the width of the object. [mm]	DOUBLE 64 bit	
X	Value	Sets/Gets the X co-ordinate of the object. [mm]	DOUBLE 64 bit	
Y	Value	Sets/Gets the Y co-ordinate of the object. [mm]	DOUBLE 64 bit	

(\*1) 0 - Undefined

1 - Group object

2 - Bitmap object

3 - Struct object

4 - Text object

(\*2) 0 - TopLeft, upper left corner of the object

1 - TopRight, upper right corner of the object

- 2 - BottomRight, lower right corner of the object
- 3 - BottomLeft, lower left corner of the object
- 4 - Center
- (\*)3 0 - Normal (default value)
- 1 - Inverted

6.2.2 Overview Valid Properties

Not all properties are valid for all object types, as shown in following table.

Type	1 - Group	2 - Bitmap	3 - Struct	4 - Text
Handle	x	x	x	x
X		x	x	x
Y		x	x	x
Rotation		x	x	x
Reference		x	x	
Height		x	x	
Width		x	x	
Scale			x	
Enabled	x	x	x	x
ImageMode		x		
Caption				x
Color		x	x	x
Pen		x	x	

Font					x
Layout		x			

6.2.3 Methods Overview

Method	Description				Return Value
VARIANT_BOOL CreateGroup()	Creates a group object.				VARIANT_TRUE when succeeded
VARIANT_BOOL CreateBitmap(BSTR FileName)	Creates a bitmap object by reading data from an image file (bmp) (*1)				VARIANT_TRUE when succeeded
VARIANT_BOOL CreateStruct(BSTR FileName, SHORT ImportTool)	Creates a struct object by reading data from an vector file (dxf/hpgl) (*2)				VARIANT_TRUE when succeeded
VARIANT_BOOL CreateEmptyStruct()	Creates an empty struct object.				VARIANT_TRUE when succeeded
VARIANT_BOOL CreateText(BSTR Caption)	Creates a text object with a specified caption.				VARIANT_TRUE when succeeded

- (\*1):  
FileName is a string expression specifying the path to the desired bitmap file. The path can be absolute, relative or a network name. If FileName is empty this function provides an open file dialog to select the file.
- (\*2):  
FileName is a string expression specifying the path to the desired DXF/HPGL file. The path can be absolute, relative or a network name.  
ImportTool defines the type of file FileName specifies.  
0 - DXF Import  
1 - HPGL Import

6.3 FLIGHTTREE

The marking image can be represented as a hierarchical list of items, a tree. Each item represents a marking object, or in case of a group item, it can have a list of subitems associated with it.

### 6.3.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the tree control.	LONG 32 bit	
DragDrop	Value	Returns/Gets a boolean value specifying whether the tree control allows drag&drop operation of tree items.	VARIANT_BOOL 16 bit	Boolean
EditLabels	Value	Returns/Gets a boolean value specifying whether the user is allowed to edit the labels of the tree items.	VARIANT_BOOL 16 bit	Boolean
Enabled	Value	Returns/Gets a boolean value specifying whether the control is enabled.	VARIANT_BOOL 16 bit	Boolean
Visible	Value	Returns/Gets a boolean value specifying whether the control is visible.	VARIANT_BOOL 16 bit	Boolean

### 6.3.2 Methods Overview

Method	Description	Return Value
LONG GetRootItem()	Retrieves root item of the tree control	Handle root item Or 0
LONG GetParentItem(LONG hItem)	Retrieves parent of tree item with handle hItem	Handle parent item Or 0
LONG GetChildItem(LONG hItem)	Retrieves child of tree item with handle hItem	Handle child item Or 0



LONG GetPrevSiblingItem(LONG hItem)	Retrieves the previous sibling of tree item with handle hItem	Handle prev. sibling item Or 0
LONG GetNextSiblingItem(LONG hItem)	Retrieves the next sibling of tree item with handle hItem	Handle next sibling item Or 0
LONG GetSelectedItem()	Retrieves the current selected tree item.	handle currently selected tree item Or 0
VARIANT_BOOL SelectItem(LONG hItem)	Selects tree item with handle hItem	VARIANT_TRUE if successful
VARIANT_BOOL ItemHasChildren(LONG hItem)	Determines whether tree item with handle hItem has child items	VARIANT_TRUE if the tree item specified by hItem has child items
Void Refresh()	Causes the control to refresh, i.e. a complete redraw of the control	
LONG Load(BSTR FileName)	Loads vector data from a file (XML). FileName specifying the path to the desired file. The path can be absolute, relative or a network name	-1 if the function was successful; Or error value
LONG Store(BSTR FileName)	Stores vector data to a file (XML) FileName specifying the path to the desired file. The path can be absolute, relative or a network name	-1 if the function was successful; Or error value
LONG InsertObject(FlightObject Object, LONG hParent, LONG hAfter)	This method inserts a tree item into the tree with handle hParent and associates FlightObject to this tree item. Instead of hParent, constant fwi_Root (0xFFFF0000) is also valid. HAfter indicates the handle of the item whereafter the new tree item is inserted. Constants Fwi_First (0xFFFF0001) and Fwi_Last (0xFFFF0001) are also valid and result in insertion as first/last sibling item. (*1)	Handle of the new item if successful; Or 0

FlightObject RemoveObject(LONG hItem)	Removes tree item with handle hItem and associated FlightObject from the tree	FlightObject associated with the deleted item if successful; Or NULL
FlightObject GetFlightObject(LONG hItem)	Retrieves associated FlightObject from a tree item with handle hItem	FlightObject associated with the tree item if successful; Or NULL
Void DeleteAllItems()	Deletes all tree items and associated FlightObjects.	
LONG FindFlightObject(FlightObject Object)	Searches tree control for a tree item having specified FlightObject associated.	Handle tree item associated with the FlightObject Or 0
LONG FindItem(String Title)	Searches tree control for a tree item with text sTitle.	Handle of found tree item Or 0
String GetItemText(LONG hItem)	Returns the text of the item specified by hItem	String containing the item's text
BOOL SetItemText(LONG hItem, String Title)	Sets the text of the item specified by hItem	Boolean value indicating whether method is successful.
Void SetFocus()	Sets focus to the control.	

### 6.3.3 Events Overview

Event	Description
Change	Notifies that the selection has changed.

## 6.4 FLIGHTCOLORS

This control is a drop-down combo box comprising an array of colors. Colors are used as an interface with a laser.

### 6.4.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	
Enabled	Value	Returns or sets whether the control is enabled.	VARIANT_BOOL 16 bit	Boolean
Visible	Value	Returns or sets whether the control is visible.	VARIANT_BOOL 16 bit	Boolean
ListCount	Read-only	Returns the number of items (i.e. the number of colors) in the control. This is the size of the color data array.	LONG 32 bit	
ListIndex	Value	Returns or sets the color array index of the currently selected Color in the control. A Value of -1 indicates no item is currently selected.	LONG 32 bit	
Data(LONG Index)	Value	Returns or sets Color from the data array where Index indicates the position in the color array. Index = 0 for the first element.	FlightColor	
PowerDutyCycle0	Value	Returns or sets duty cycle at 0% laser power. Expressed in %. (*1)	LONG 32 bit	
PowerDutyCycle100	Value	Returns or sets duty cycle at 100% laser power. Expressed in %. (*1)	LONG 32 bit	

PowerPulseRate	Value	Returns or sets the base frequency of the power pulse width modulated signal. Expressed in Hz. (*2)	LONG 32 bit	
LaserEvent	Value	Returns or sets the Laser Event mode.	LONG 32 bit	0,1,2 (*3)
Presentation	Value	Returns or sets the Color presentation style, that determines the properties of the color that are shown to the user when connected to a FlightPropList control.	LONG 32 bit	(*4)

(\*1)

PowerDutyCycle0 and PowerDutyCycle100 are use for linear interpolation of the Power property.

(\*2)

Power pulse rate defines the base frequency of the PWM signal.

(\*3)

0 - None: FlightMarker object will fire no events.

1 - By Change: FlightMarker object will fire an event when laser properties change.

2 - Always: FlightMarker object will always fire an event.

(\*4)

The Presentation value is a bitwise combination of following constants:

- (0x0001) fcps\_MarkSpeed
- (0x0002) fcps\_JumpSpeed
- (0x0004) fcps\_PulseRate
- (0x0008) fcps\_PulseWidth
- (0x0010) fcps\_Power
- (0x0020) fcps\_BurstTime
- (0x0040) fcps\_Repetitions
- (0x0080) fcps\_LaserOnDelayfcps\_LaserOffDelay (0x0100)
- (0x0200) fcps\_JumpDelay
- (0x0400) fcps\_MarkDelay
- (0x0800) fcps\_PolygonDelay

example: Value : MarkSpeed, PulseRate, PulseWidth, Power

6.4.2 Methods Overview

Method	Description	Return Value
void AddColor()	Adds a Color at the end of the data array.	
void RemoveColor()	Removes last Color from the the data array.	
void Clear()	Deletes the complete data array.	
LONG Load(BSTR FileName)	Loads the data array from a file (XML). (*1)	-1 if the function was successfull; Or an error value
LONG Store(BSTR FileName)	Stores the data array to a file (XML). (*1)	-1 if the function was successfull; Or an error value
void Refresh()	Causes the control to refresh, i.e. a complete redraw of the control.	
void SetFocus()	Sets focus to the control.	

(\*1): FileName is a string expression specifying the path to the desired file. The path can be absolute, relative or a network name.

6.4.3 Events Overview

Event	Description
void Change()	Indicates that the content of the control have changed.
void Click()	User selects an item in the control, either by pressing the arrow keys or by clicking the mouse button.
void DropDown()	The list portion of the control is about to drop down.
void KeyDown(Short KeyCode, Short ShiftState)	User presses a key while the control has the focus.
void KeyPress(Short KeyCode)	User presses and releases a key while control has the focus.

void KeyUp(Short KeyCode, Short ShiftState)	User releases a key while the control has the focus.
---	--

## 6.5 FLIGHTCOLOR

This object represents a laser color. It cannot exist without his parent (FlightColors object).

### 6.5.1 Properties Overview

Property	Access Mode	Description	Type	Range
RGB	Value	Returns or sets the RGB color, used for drawing in a FlightView object.	OLE_COLOR	
Speed	Value	Returns or sets the marking speed, expressed in mm/s.	DOUBLE 64 bit	
JumpSpeed	Value	Returns or sets the jump speed, expressed in mm/s.	DOUBLE 64 bit	
Repetitions	Value	Returns or sets the number of repetitions..	LONG 32 bit	
Power	Value	Returns or sets the laser power, expressed in %.	SHORT 16 bit	[0, 100]
QswPulseRate	Value	Returns or sets the laser qswitch frequency, expressed in Hz.	LONG 32 bit	
QswPulseWidth	Value	Returns or sets the laser qswitch pulse width, expressed in $\mu$ s.	DOUBLE 64 bit	
BurstTime	Value	Returns or sets the laser burst time, expressed in $\mu$ s.	LONG	

				32 bit	
JumpDelay	Value	Returns or sets the jump delay, expressed in $\mu$ s. A jump delay can be used to insert a delay after each jump to ensure the deflection head is in position when the marking starts.		LONG 32 bit	
MarkDelay	Value	Returns or sets the mark delay, expressed in $\mu$ s. A mark delay can, be used to eliminate possible tails at the end of the vectors.		LONG 32 bit	
PolygonDelay	Value	Returns or sets the polygon delay, expressed in $\mu$ s. A polygon delay can be used to set a delay at corners of vector lines to ensure 'sharpness'.		LONG 32 bit	
LaserOnDelay	Value	Returns or sets the laser on delay, expressed in $\mu$ s. LaserOnDelay can be used to minimize the hot spots at the beginning of the vectors.		LONG 32 bit	
LaserOffDelay	Value	Returns or sets the laser off delay, expressed in $\mu$ s. LaserOff delay can be used to maintain the beam output at the end of the vectors to ensure completion of the vectors before jumping to the next ones.		LONG 32 bit	

## 6.6 FLIGHTPENS

The FlightPens control is a drop-down combo box. It's a container for an array of pens. Pens are used as to define the parameters of a vectoring process.

### 6.6.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	
Enabled	Value	Returns or sets whether the control is enabled.	VARIANT_BOOL	Boolean

				16 bit	
Visible	Value	Returns or sets whether the control is visible.		VARIANT_BOOL 16 bit	Boolean
ListCount	Read-only	Returns the number of items (Pens) in the control, this is the size of the Pen data array.		LONG 32 bit	-1 indicates no item is currently selected. Or the currently selected Pen from the array.
ListIndex	Value	Returns or sets the index of the currently selected Pen in the control.		LONG 32 bit	
Data(LONG Index)	Value	Returns or sets Pen from the data array where Index indicates the position in the pen array. Index = 0 for the first element.		FlightPen	

## 6.6.2 Methods Overview

Method	Description	Return Value
void AddPen()	Adds a Pen at the end of the data array.	
void RemovePen()	Removes last Pen from the the data array.	
void Clear()	Deletes the complete data array.	
LONG Load(BSTR FileName)	Loads the data array from a file (XML). (*1)	-1 if the function was successful; Or an error value
LONG Store(BSTR FileName)	Stores the data array to a file (XML). (*1)	-1 if the function was successful;



			Or an error value
void Refresh()		Causes the control to refresh, i.e. a complete redraw of the control.	
void SetFocus()		Sets focus to the control.	

(\*1): FileName is a string expression specifying the path to the desired file. The path can be absolute, relative or a network name.

### 6.6.3 Events Overview

Event	Description
void Change()	Indicates that the content of the control have changed.
void Click()	User selects an item in the control, either by pressing the arrow keys or by clicking the mouse button.
void DropDown()	The list portion of the control is about to drop down.
void KeyDown(Short KeyCode, Short ShiftState)	User presses a key while the control has the focus.
void KeyPress(Short KeyCode)	User presses and releases a key while control has the focus.
void KeyUp(Short KeyCode, Short ShiftState)	User releases a key while the control has the focus.

## 6.7 FLIGHTPEN

This object represents a vectoring pen. It cannot exist without his parent (FlightPens object).

### 6.7.1 Properties Overview

Property	Access Mode	Description	Type	Range
Outlines	Value	Returns or sets the number of outlines.	LONG 32 bit	

Pitch	Value	Returns or sets the distance between two adjacent outlines, expressed in mm.	DOUBLE 64 bit	
HatchStyle	Value	Returns or sets hatching mode. Hatching can only be generated on bitmap objects and bitmap fonts.	SHORT 16 bit	0, 1, 2, 3, 4, 5 (*1)
HatchPitch	Value	Returns or sets the distance between two adjacent hatch lines, expressed in mm.	DOUBLE 64 bit	
Optimize	Value	Returns or sets the whether the vector lines, generated from vectoring process, are sorted.	VARIANT_BOOL 16 bit	Boolean

(\*1)

- 0 - No hatching
- 1 - Horizontal hatching pattern
- 2 - Diagonal 45° hatching pattern
- 3 - Vertical hatching pattern
- 4 - Diagonal 135° hatching pattern
- 5 - Erode hatching pattern

## 6.8 FLIGHTFONTS

The FlightFont control is a drop-down combo box and acts as a container for an array of fonts. This component provides the possibility to use all the Windows TrueType and SingleStroke fonts installed on the PC. Besides these text fonts, the component also provides support for 1D and 2D barcodes.

### 6.8.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	

Enabled	Value	Returns or sets whether the control is enabled.	VARIANT_BOOL 16 bit	Boolean
Visible	Value	Returns or sets whether the control is visible.	VARIANT_BOOL 16 bit	Boolean
ListCount	Read-only	Returns the number of items (Fonts) in the control, i.e. the size of the Font data array.	LONG 32 bit	
ListIndex	Value	Returns or sets the index of the currently selected Font in the control.	LONG 32 bit	-1 indicates no item is currently selected Or index of the currently selected Font from the array.
Data(LONG Index)	Value	Returns or sets Font from the data array where Index indicates the position in the font array. Index = 0 for the first element.	FlightFont	

## 6.8.2 Methods Overview

Method	Description	Return Value
void AddTrueTypeFont()	Adds a True Type font at the end of the data array.	
void AddSingleStrokeFont()	Adds a Single Strike font at the end of the data array.	
void AddBarcodeFont()	Adds a Barcode font at the end of the data array.	
void RemoveFont()	Removes last Font from the the data array.	
void Clear()	Deletes the complete data array.	
LONG Load(BSTR FileName)	Loads the data array from a file (XML). (*1)	-1 if the function was successful;

			Or an error value
LONG Store(BSTR FileName)		Stores the data array to a file (XML). (*1)	-1 if the function was successful; Or an error value
void Refresh()		Causes the control to refresh, i.e. a complete redraw of the control.	
void SetFocus()		Sets focus to the control.	
(*1): FileName is a string expression specifying the path to the desired file. The path can be absolute, relative or a network name.			

### 6.8.3 Events Overview

Event	Description
void Change()	Indicates that the content of the control have changed.
void Click()	User selects an item in the control, either by pressing the arrow keys or by clicking the mouse button.
void DropDown()	The list portion of the control is about to drop down.
void KeyDown(Short KeyCode, Short ShiftState)	User presses a key while the control has the focus.
void KeyPress(Short KeyCode)	User presses and releases a key while control has the focus.
void KeyUp(Short KeyCode, Short ShiftState)	User releases a key while the control has the focus.

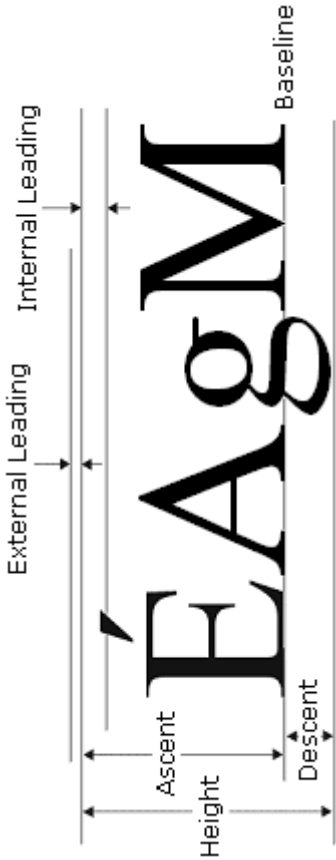
## 6.9 FLIGHTFONT

This object represents a font. It cannot exist without his parent (FlightFonts object).

### 6.9.1 Font Height

The size of a font is a rather imprecise value. FLIGHT SDK uses the Windows font height to determine the metric font height.  
The Windows font height is the sum of the Ascent and Descent values of the font - see figure.

A font in FLIGHT SDK with a Font Height of 3mm is that size where the sum of the Ascent and Descent in Windows is 3mm.



6.9.2 Properties Overview

Property	Access Mode	Description	Type	Range
Type	Read-only	Returns the type of the font.	LONG 32 bit	1,2,3 (*1)
SubType	Read-only	Returns the subtype of a barcode font.	LONG 32 bit	1,2,3 (*2)
FaceName	Value	Returns or sets the typeface name of the font.	BSTR	
Height	Value	Returns or sets the font height, expressed in mm.	DOUBLE 64 bit	
LineSpacing	Value	Returns or sets the line spacing of the font used by multiline text objects, expressed in mm.	DOUBLE 64 bit	
Style	Value	Returns or sets the style of the typeface of a font.	LONG	0,1,2,3 (*3)

					32 bit	
Pen	Value	Returns or sets the reference to the pen used while vectoring the source font. This is the index of the Pen in the Pen data array of the FlightPens object.			LONG 32 bit	
Margin	Value	Returns or sets the leading and trailing margin width in a Code39 (Extended) barcode. Expressed in mm.			DOUBLE 64 bit	
NarrowBarWidth	Value	Returns or sets the width of a narrow bar pattern in a Code39 (Extended) barcode. Expressed in mm.			DOUBLE 64 bit	
WideToNarrow	Value	Returns or sets the wide to narrow ratio in a Code39 (Extended) barcode. Expressed in mm.			DOUBLE 64 bit	
ImageMode	Value	Returns or sets the image mode of a Code39 (Extended) barcode.			LONG 32 bit	0,1 (*4)
CheckDigit	Value	Returns or sets whether a Code39 (Extended) barcode comprises a check digit.			VARIANT_BOOL 16 bit	
Charset	Value	Returns or sets the character set of a TrueType font. Not all fonts support all charsets.			SHORT 16 bit	
SymbolSize	Value	Returns or sets smallest symbol size of Datamatrix barcodes. However if the data encoded creates a larger symbol, the size will automatically increase.			LONG 32 bit	
ToolShape	Value	Returns or sets the tool used to create Datamatrix barcodes.			LONG 32 bit	0,1,2 (*5)
Encoding	Value	Returns or sets the method used to encode data of Datamatrix barcodes.			LONG 32 bit	0,1,2,3,4,5 (*6)

- (\*1)

1 - True Type font

2 - Single Stroke font

3 - Barcode font

1 - Code39 barcode font

2 - Code39 Extended barcode font

3 - DataMatrix ECC 200 barcode font

0 - Regular

1 - Bold

2 - Italic

3 - Bold + Italic

0 - Normal

1 - Inverted

0 - Square tool

1 - Round tool

2 - Pulse tool
- (\*2)

0 - ASCII - preferable to be used for encoding ASCII characters

1 - C40 - preferable to be used for encoding uppercase letters and numbers

2 - TEXT - preferable to be used for encoding lowercase letters and numbers

3 - X12 - encodes X12 EDI data set

4 - EDIFACT - encodes ASCII values 32 - 94

5 - BINARY - encodes bytes of data
- (\*3)
- (\*4)
- (\*5)
- (\*6)

6.9.3 Overview Valid Properties

Not all properties are valid for all object types and subtypes, as shown in following table.

Type	1 - True Type	2 - Single Stroke	3 - Barcode	3 - Barcode	3 - Barcode
Subtype			1	2	3
FaceName	x	x	Code39	Code39 Extended	DataMatrix

Height	x	x	x	x	x	x
LineSpacing	x	x				
Style	x					
Pen	x			x	x	x
Margin				x	x	
NarrowBarWidth				x	x	
WideToNarrow				x	x	
ImageMode				x	x	
CheckDigit				x	x	
CharSet	x					
SymbolSize						x
ToolShape						x
Encoding						x

## 6.10 FLIGHTLAYOUTS

The FlightLayouts control is a drop-down combo box comprising an array of layouts. A layout is a set of position definitions, where a marking image has to be marked. A layout can be a description of a "step and repeat" structure.

### 6.10.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	



Enabled	Value	Returns or sets whether the control is enabled.	VARIANT_BOOL 16 bit	Boolean
Visible	Value	Returns or sets whether the control is visible.	VARIANT_BOOL 16 bit	Boolean
ListCount	Read-only	Returns the number of items (Layouts) in the control, i.e. the size of the Layout data array.	LONG 32 bit	
ListIndex	Value	Returns or sets the index of the currently selected Layout in the control.	LONG 32 bit	-1 indicates no item is currently selected Or index of the currently selected Layout from the array.
Data(LONG Index)	Value	Returns or sets Layout from the data array where Index indicates the position in the Layout array. Index = 0 for the first element.	FlightLayout	

## 6.10.2 Methods Overview

Method	Description	Return Value
void AddLayout()	Adds a Layout at the end of the data array.	
void RemoveLayout()	Removes last Layout from the the data array.	
void Clear()	Deletes all Layouts from the data array.	
LONG Load(BSTR FileName)	Loads the data array from a file (XML). (*1)	-1 if the function was successful; Or an error value
LONG Store(BSTR FileName)	Stores the data array to a file (XML). (*1)	-1 if the function was successful;

			Or an error value
void Refresh()		Causes the control to refresh.	
void SetFocus()		Sets focus to the control.	
(*1): FileName is a string expression specifying the path to the desired file. The path can be absolute, relative or a network name.			

### 6.10.3 Events Overview

Event	Description
void Change()	Indicates that the content of the control have changed.
void Click()	User selects an item in the control, either by pressing the arrow keys or by clicking the mouse button.
void DropDown()	The list portion of the control is about to drop down.
void KeyDown(Short KeyCode, Short ShiftState)	User presses a key while the control has the focus.
void KeyPress(Short KeyCode)	User presses and releases a key while control has the focus.
void KeyUp(Short KeyCode, Short ShiftState)	User releases a key while the control has the focus.

### 6.11 FLIGHTLAYOUT

This object represents a layout. It cannot exist without his parent (FlightLayouts object). A layout defines a set of marking positions inside the co ordinate system of the group.

Property	Access Mode	Description	Type	Range
Positions	Value	Sets/Gets the number of positions defined by the layout. Returns or sets the number of positions defined by the layout, i.e. the size of the X, Y, Rotation and Enabled arrays.	LONG 32 bit	
X[i]	Value	Returns or sets the X co-ordinate of a layout position definition defined by Index i. Expressed in mm.	Double	

				64 bit	
Y[i]	Value	Returns or sets the Y co-ordinate of a layout position definition defined by Index i. Expressed in mm.		Double 64 bit	
Rotation[i]	Value	Returns or sets the Rotation around the X/Y co-ordinate of the layout position definition defined by Index i. Expressed in degrees.		Double 64 bit	
Enabled[i]	Value	Returns or sets whether a layout position is enabled or not. Only enabled position will be marked.	VARIANT_BOOL	16 bit	Boolean

## 6.12 FLIGHTVIEW

The FlightView component represents the physical marking field and shows each vector object added to a FlightTree. The boundaries of the marking field represent the field size depending on the Field definition. Vector objects will be drawn in the color appropriate with the RGB color defined through its color reference.

### 6.12.1 Properties overview tot hier

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	
ViewMode	Value	Gets/Sets the ViewMode of the control.	LONG 32 bit	(*1)
FieldWidth	Value	Gets/Sets the width of the marking field, expressed in mm. This is not the physical field size of the rhothor deflection head itself.	DOUBLE 64 bit	
FieldHeight	Value	Gets/Sets the height of the marking field, expressed in mm. This is not the physical field size	DOUBLE	

		of the rhothor deflection head itself.	64 bit	
BackColor	Value	Gets/Sets background color.	OLE_COLOR 32 bit	
ForeColor	Value	Gets/Sets fore color.	OLE_COLOR 32 bit	
BorderColor	Value	Gets/Sets border color.	OLE_COLOR 32 bit	
PanelColor	Value	Gets/Sets panel color.	OLE_COLOR 32 bit	
MajorGridColor	Value	Gets/Sets color used for drawing major grid lines	OLE_COLOR 32 bit	
MinorGridColor	Value	Gets/Sets color used for drawing minor grid lines	OLE_COLOR 32 bit	
DisabledColor	Value	Gets/Sets the color used for drawing disabled objects	OLE_COLOR 32 bit	
Visible	Value	Gets/Sets a value indicating whether the control is visible	VARIANT_BOOL 16 bit	

(\*)1:

The ViewMode value is one of following constants:

- (0x0000) fvm\_Static view acts as a static window.
- (0x0001) fvm\_Zoom view captures mouse input to zoom in the view through mouse interaction.
- (0x0002) fvm\_ZoomOut view captures mouse input to zoom out the view through mouse interaction.
- (0x0003) fvm\_Pan view captures mouse input to pan the zoom through mouse interaction.
- (0x0004) fvm\_Select view generates event when user clicks on object in the view.

### 6.12.2 Methods Overview

Method	Description	Return Value
DOUBLE ClientToDocX(LONG X)	Convert client X coordinate, expressed in pixels, into a document X co-ordinate.	X co-ordinate value, expressed in mm
DOUBLE ClientToDocY(LONG Y)	Convert client Y coordinate, expressed in pixels, into a document Y co-ordinate.	Y co-ordinate value, expressed in mm
void Select(FlightObject Object)	Selects an object in the view.	
void Refresh()	Causes the control to refresh, i.e. a complete redraw of the control.	
void SetFocus()	Sets focus to the control.	

### 6.12.3 Events Overview

Event	Description
void Click()	User selects an item in the control, either by pressing the arrow keys or by clicking the mouse button.
void MouseMove(Short Button, Short X, Short Y)	Mouse pointer is moved over the control while the control has focus.
void KeyDown(Short KeyCode, Short ShiftState)	User presses a key while the control has the focus.
void KeyPress(Short KeyCode)	User presses and releases a key while control has the focus.
void KeyUp(Short KeyCode, Short ShiftState)	User releases a key while the control has the focus.
void ObjectClick(FlightObject Object)	Occurs when user clicks on FlightObject and the View Mode of the control is fvmc_Select (0x0004).

## 6.13 FLIGHTPROPLIST

The FlightPropList is a property list, being able to show and modify the properties of different objects including FlightObject, FlightPen, FlightColor, FlightFont, FlightLayout. To attach an object to this property list change the "PropSrc" property of the FlightPropList control to this object.

### 6.13.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	
Enabled	Value	Gets/Sets whether the control is enabled.	VARIANT_BOOL 16 bit	
Visible	Value	Gets/Sets a value indicating whether the control is visible	VARIANT_BOOL 16 bit	
PropSrc		Sets the property source of this control to an object. Properties of this object are shown in the property control and can be changed by the user.	Object, can be FlightObject, FlightFont, FlightColor, FlightLayout or FlightPen	

### 6.13.2 Methods Overview

Method	Description	Return Value
void Clear()	Clears the property source of the control.	
void Refresh()	Causes the control to refresh, i.e. a complete redraw of the control.	

SetFocus()	Sets focus to the control.	
------------	----------------------------	--

## 6.14 FLIGHTEDITOR

The FlightEditor is a component, which can show a vector graphics editor to modify an existing object.

### 6.14.1 Properties Overview

Property	Access Mode	Description	Type	Range
Handle	Read-only	Returns a handle to the control.	LONG 32 bit	
Visible	Value	Gets/Sets a value indicating whether the control is visible	VARIANT_BOOL 16 bit	

### 6.14.2 Methods Overview

Method	Description	Return Value
void Edit(FlightObject Object)	Attaches a FlightObject to the control for editing.	
void ShowEditor()	Makes the vector graphics editor dialog visible.	
void HideEditor()	Hides the vector graphics editor dialog.	

## 6.15 FLIGHTMARKER

FlightMarker is the master component. It controls all communication between the components, and controls the marking process and interface to the hardware.

### 6.15.1 Properties Overview

Property	Access Mode	Description	Type	Range
----------	-------------	-------------	------	-------

Visible	Value	Sets/Gets a value indicating whether the control is visible.	VARIANT_BOOL 16 bit	
Serial	Value	Sets/Gets serial number of the rhothor device used during enumeration. Enumeration of the devices is done before the first call to the rhothor device. The serial value will be matched against the rhothor's serial number to find the appropriate deflection head. A value of -1 indicates that the control may choose any available rhothor device.	LONG 32 bit	
Resolution	Value	Sets/Gets the global resolution, expressed in mm. This resolution is used during the vectoring process from bitmaps, text objects using TrueType fonts and struct object with multiple outlines.	DOUBLE 64 bit	[0.001 - 0.050]
HwLibFileName	Value	Sets/Gets the Hardware Library filename. Default the SDK uses the Flight.dll to interface to the rhothor deflection head. In multihead applications this can be used to create extra interfaces.	BSTR	

### 6.15.2 Methods Overview

Method	Description	Return Value
void Attach(Object)	Attaches a Flight ActiveX component to the marker. Every component should be attached before use. Object can be a FlightColors, FlightFonts, FlightPens, FlightLayouts, FlightView, FlightPropList or FlightEditor control.	
void Detach(Object)	Detaches a Flight ActiveX component from the marker. Object can be a FlightColors, FlightFonts, FlightPens, FlightLayouts, FlightView, FlightPropList or FlightEditor control.	
void UpdateData()	Recompiles data and update all attached views.	
LONG Mark()	Starts marking process.	-1 if the function was succesfull; or an error value otherwise.
LONG SetIO(LONG Value, LONG Mask )	Sets the state of the configured OUTPUTS on the deflection	-1 if the function was succesfull; or an error



	head. Can't be called during marking process. Value determines bitwise the output states of the appropriate IO's. Mask determines bitwise which output states of Value are valid and should be set. (*1)	value otherwise.
LONG GetIO(LONG* Value)	Retrieves the state of the configured OUTPUTS on the deflection head. Can't be called during marking process. Value retrieves the output states of the appropriate IO's.	-1 if the function was succesfull; or an error value otherwise.
LONG SetLaser(VARIANT_BOOL LaserOn)	Swaps the laser hardware state between continuous pulsing mode on/off. Can't be called during marking process. LaserOn (Boolean) determines whether the laser should move into the continuously-on mode or return to the normal state.	-1 if the function was succesfull; or an error value otherwise.
LONG JumpTo(DOUBLE X, DOUBLE Y, DOUBLE Z)	Moves the deflection head to a certain position. Can't be called during marking process. For 2D laser deflection systems Z value is ignored.	-1 if the function was succesfull; or an error value otherwise.
LONG Abort()	Aborts the current marking process.	
LONG SetMatrix(DOUBLE a00, DOUBLE a01, DOUBLE a10, DOUBLE a11)	Changes the hardware transformation matrix. Can't be called during marking process.	-1 if the function was succesfull; or an error value otherwise.
LONG SetOffset(DOUBLE X, DOUBLE Y, DOUBLE Z)	Changes the hardware offset. Can't be called during marking process. For 2D laser deflection systems Z value is ignored.	-1 if the function was succesfull; or an error value otherwise.
LONG SelectColor(LONG Color)	Sets the laser parameters of a Color (Index to Color of the attached FlightColors control) into the hardware. Can't be called during marking process.	-1 if the function was succesfull; or an error value otherwise.
LONG LoadCalibrationFile(BSTR FileName)	Loads calibration file into the hardware. The calibration data remains in the deflection head till power down. (*2)	-1 if the function was succesfull; or an error value otherwise.

(\*1) Example: Calling SetIO(4,6) will set output pin 2 low and output pin 3 high.

(\*2): FileName is a string expression specifying the path to the desired file. The path can be absolute, relative or a network name.

6.15.3 Events Overview

Event	Description
void OnMarkingFinished(LONG Error)	Marking process is finished. Error contains -1 if the process was finished successfully; or an error value otherwise.
void OnLaserChanged(VARIANT* Params)	Laser parameters are changed during marking process. To enable these events, change the LaserEvent property of the attached FlightColors object.. (*1)

(\*1) Params will be a variant containing a BYTE array of 16 Bytes presenting following structure:

- LONG Power (4 bytes) - Laser power [%]
- LONG PulseRate (4 bytes) - Qswitch pulse rate [Hz]
- DOUBLE PulseWidth (8 bytes) - Qswitch pulse width [µs]

## 7 APPENDIX: CHARACTER SET CODES

The following table defines the list of possible values for the character set codes.

Code	Value (Codepage)	Alphabet
DIN_66003	20106	IA5 (German)
NS_4551-1	20108	IA5 (Norwegian)
SEN_850200_B	20107	IA5 (Swedish)
_autodetect	50932	Japanese (Auto Select)
_autodetect_kr	50949	Korean (Auto Select)
big5	950	Chinese Traditional (Big5)
csISO2022JP	50221	Japanese (JIS-Allow 1 byte Kana)
euc-kr	51949	Korean (EUC)
gb2312	936	Chinese Simplified (GB2312)
hz-gb-2312	52936	Chinese Simplified (HZ)
ibm852	852	Central European (DOS)
ibm866	866	Cyrillic Alphabet (DOS)
irv	20105	IA5 (IRV)
iso-2022-jp	50220	Japanese (JIS)
iso-2022-jp	50222	Japanese (JIS-Allow 1 byte Kana)
iso-2022-kr	50225	Korean (ISO)
iso-8859-1	1252	Western Alphabet
iso-8859-1	28591	Western Alphabet (ISO)
iso-8859-2	28592	Central European Alphabet (ISO)
iso-8859-3	28593	Latin 3 Alphabet (ISO)
iso-8859-4	28594	Baltic Alphabet (ISO)
iso-8859-5	28595	Cyrillic Alphabet (ISO)
iso-8859-6	28596	Arabic Alphabet (ISO)
iso-8859-7	28597	Greek Alphabet (ISO)
iso-8859-8	28598	Hebrew Alphabet (ISO)
koi8-r	20866	Cyrillic Alphabet (KOI8-R)
ks_c_5601	949	Korean
shift-jis	932	Japanese (Shift-JIS)
unicode	1200	Universal Alphabet
unicodeFEFF	1201	Universal Alphabet (Big-Endian)
utf-7	65000	Universal Alphabet (UTF-7)
utf-8	65001	Universal Alphabet (UTF-8)

windows-1250	1250	Central European Alphabet (Windows)
windows-1251	1251	Cyrillic Alphabet (Windows)
windows-1252	1252	Western Alphabet (Windows)
windows-1253	1253	Greek Alphabet (Windows)
windows-1254	1254	Turkish Alphabet
windows-1255	1255	Hebrew Alphabet (Windows)
windows-1256	1256	Arabic Alphabet (Windows)
windows-1257	1257	Baltic Alphabet (Windows)
windows-1258	1258	Vietnamese Alphabet (Windows)
windows-874	874	Thai (Windows)
x-euc	51932	Japanese (EUC)

## 8 APPENDIX: BARCODE ECC 200 FONT SYMBOL SIZE

Symbol Size	Row x Columns	Max. Data Capacity: Numeric	Max. Data Capacity: Alphanumeric	Max. Data Capacity: Byte
0	AUTO	-	-	-
1	10 x 10	6	3	1
2	12 x 12	10	6	3
3	14 x 14	16	10	6
4	16 x 16	24	16	10
5	18 x 18	36	25	16
6	20 x 20	44	31	20
7	22 x 22	60	43	28
8	24 x 24	72	52	34
9	26 x 26	88	64	42
10	32 x 32	124	91	60
11	36 x 36	172	127	84
12	40 x 40	228	169	112
13	44 x 44	288	214	142
14	48 x 48	348	259	172
15	52 x 52	408	304	202
16	64 x 64	560	418	277
17	72 x 72	736	550	365
18	80 x 80	912	682	453
19	88 x 88	1152	862	573
20	96 x 96	1392	1042	693
21	104 x 104	1632	1222	813
22	120 x 120	2100	1573	1047
23	132 x 132	2608	1954	1301
24	144 x 144	3166	2335	1555
25	8 x 18	10	6	3
26	8 x 32	20	13	8
27	12 x 26	32	22	14
28	12 x 36	44	31	20
29	16 x 36	64	46	30
30	16 x 48	98	72	47

Please list the following information, and use this outline to provide us with your comments about this document.

Would you like a reply?                    Y            N

1. What additions to the document do you think would enhance the subject and/or the structure ?

---

---

---

---

---