TpaCutOEM

TpaCutOEM 库

用户手册

版本 **2.0.0**

xx/xx/2025

T.P.A.Srl Tecnologie e Prodotti per l' Automazione -Via Carducci, 221 - 20099 Sesto S. Giovanni (MI) ITALY
电话+393666507029 - www.tpaspa.com info@tpaspa.it

1. 介绍

TpaCutOEM.dll（又名 TPA_C）是一款专为软件开发人员设计的产品，可以集成为 32 位和 64 位 Windows 体系结构的库。
TPA_C 库针对自动优化线性切割矩形形状而开发，可以开发应用程序，优化各个应用领域的 2D 放置。

库的典型用途与切割机编程相关。

库必须集成为产品的一部分。

1.1. 版本

版本	*发布日期*	*注释*
1.1.0	28.03.2024	
1.2.0	08.07.2024	- *ImportProject* 函数中的新实施
1.3.0	09.12.2024	- 条排序中的新实施 - 增加废料区域管理（残料/废料）
		-
2.0.0	XX-03-2025	- 新的授权管理系统

1.2. 许可证

从 2.0.0 版本开始，TPA_C 的运行需要软件许可证验证。
本文档中，“key（密钥）”一词与“软件许可证”同义。
软件许可证管理由专门的应用程序负责，具体信息请参见该应用程序说明。

在没有许可证的情况下，将无法执行任何项目优化操作。

1.3. 系统要求

安装本应用程序的计算机应满足以下最低要求：

- 操作系统: 64 位 Windows 10 或更高版本
- 处理器: 双核（CPU 越快，优化计算越快）
- 内存: 1 GB 或以上
- 所需 Windows 扩展组件：NET Framework 4.8 或更高版本
- 安装该软件套件需要管理员权限

1.4.所有权和版权

1.5.联系方式

如需更多信息，请联系：

T.P.A. Srl Tecnologie e Prodotti per l'Automazione - Via Carducci, 221 - 20099 Sesto S. Giovanni (MI) - ITALY
电话：+393666507029
电子邮件：info@tpaspa.it

或访问我们的网站：
www.tpaspa.com

2. 用户指南

TPA_C 赋值 ***TpaCutOEM*** 命名空间的 ***TpaRctCut*** 类。

TPA_C 计算矩形放置区域中的矩形形状线性切割。
矩形 是单个实体，可以通过在一个截面区域进行多次切割来实现。
截面区域称为*板材*，往往也是矩形。

赋值*矩形* 可以定义通常（可用数量、放置优先级）或普通技术信息（厚度、材料、纹理、旋转可能性…）。对于每种矩形，定义请求的放置数量，以及填充已使用板材可以额外放置的额外数量。

赋值*板材* 可以定义普通技术设置（厚度、材料、纹理…）。对于每种板材，定义可用数量。

普通技术信息可以对优化过程应用*矩形* 和*板材* 之间的匹配和/或筛选条件。

优化*结果* 为用户提供从*可用板材* 切割*矩形* 的最佳方式的详细信息。优化*结果* 可以赋值一个或多个*板材*：每个*板材* 包含至少一个*矩形* 的切割。

下图显示优化板材的示例，也称为*切割模式*：

2.1. 切割优化

通过在接下来的嵌套操作中交替顺序水平和垂直切割，实现板材切割模式上指示的矩形。
为板材计算的切割模式的结果是交替拆分为水平和垂直切割来切割单个部件，直到实现所有所需分隔。

带来切割求解的计算过程组称为*优化步骤*。
切割矩形可以对同一矩形应用 0° 或 90° 旋转（逆时针）。
优化可以提出最多 5 个求解，用户可进行评估以选择最佳求解。

2.2. 单位和坐标

在 TPA_C 中，在 *Unit* 属性 (0= [mm], 1= [inch]) 中定义为 [mm] 或 [inch] 的单位中表示为公制值（坐标，尺寸）。

TPA_C 采用 2D 笛卡尔坐标系：

- 坐标点 [0;0] 位于板材左下角
- X 是水平轴，向右为正
- Y 是垂直轴，向上为正。

计算的切割坐标始终参考板材左下角。

2.3. 方向和延伸角落

方向赋值板材第一个切割的方向：

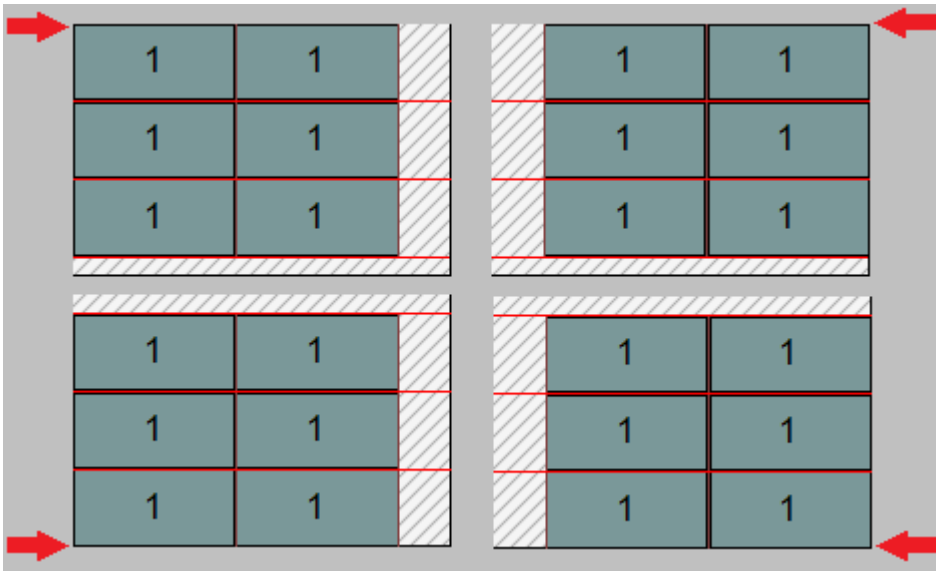
- *水平方向*: 切割将水平开始，初始切割等级为 1
- *垂直方向*：切割将垂直开始，初始切割等级为 0（或者称为：头切）
- *未设置方向*：每个优化板材可以具有垂直或水平方向，取决于哪个方向可以为该板材实现最佳效果。

角落选择赋值板材放置的起始矢量，废料积累在相对角落附近。值有 4 个：

- 0 = 左下
- 1 = 左上
- 2 = 右下
- 3 = 右上

TPA_C 始终使用赋值的角落设置。
信息对应 Direction 和 Corner 项目常规赋值。

下图显示在所有顶点（红色箭头指示）执行的水平方向优化的示例



本文中的“第一个切割”、“最后一个切割”、“第一条”、“最后一条”等词语必须分别解释为最靠近或最远离优化顶点的意思。
除非另有说明，否则这里的所有示例使用顶点 0。

2.4.匹配组和筛选器

矩形 和*板材* 赋值具有通常技术设置，应用*矩形* 和*板材* 之间的匹配和/或筛选条件。

首先看一看有助于应用匹配条件的设置。

受影响的设置对应 OneRect 和 OneSheet 结构的字段：

- *Thickness*：（double 类型）厚度
- *Material*：（string 类型）普通材料赋值
- *Grain*：（integer 类型）赋值纹理或纹理方向。

Thickness 字段：根据设定值相等性评估匹配，差值是否小于对应 0.1 mm 的线性比较的 epsilon。

赋值矩形和板材厚度匹配组的示例：

- 赋值两个矩形（标识符：1, 2），字段 *Thickness* =18.0
- 赋值一个矩形（标识符：3），字段 *Thickness* =25.0
- 赋值一个板材（标识符：1），字段 *Thickness* =0.0
- 赋值一个板材（标识符：2），字段 *Thickness* =18.0

此情况决定赋值 3 个匹配组：

- 组 1，匹配 *Thickness* =18.0
- 组 2，匹配 *Thickness* =25.0
- 组 3，匹配 *Thickness* =0.0。

每个组单独优化。从上面的示例可以看到，只有组 1 可以确定切割求解，实现矩形与板材之间的关联：标识矩形 (1, 2) 可以放置在标识符 2 的板材上。

赋值多个值时（例如材料），矩形和板材之间的匹配关联还可变得更加复杂。示例：

- 组 1，匹配 *Thickness* = 18.0 和 *Material*= “mahogany”
- 组 2，匹配 *Thickness* = 18.0 和 *Material*= “birch”
- 组 3，匹配 *Thickness* = 25.0 和 *Material*= “mahogany”

2.4.1. 纹理控制

OneRect 和 OneSheet 结构中显示字段，对应纹理或纹理方向赋值：

- 赋值发生在结构的 *Grain* 字段中
- 此信息的技术含义取决于板材材料（如：木材、胶合板、金属）。

可赋值三个值：

- *0 (None)*：不赋值纹理
- *1 (X)*：沿水平方向赋值纹理
- *2 (Y)*：沿垂直方向赋值纹理。

纹理赋值不一定确定匹配不同组：*Grain* 字段值相同的矩形可以放在不同组板材中，和/或与 *Grain* 字段值相同或不同的放在一起，

我们来看一下如果允许矩形旋转，应用的标准：

- *Grain=(1, 2)* 的矩形可以在 *Grain=0* 的板材中任意旋转放置
- *Grain=(1, 2)* 的矩形可以放置在 *Grain=(1, 2)* 的板材中，旋转时遵守二者的赋值方向
- *Grain=0* 的矩形可以任意旋转放置，与板材的 *Grain* 字段无关。

Grain=(1, 2) 的矩形只能放置在具有相同 Grain 的板材上，但无法旋转。

如果启用旋转且 *Grain=(1, 2)* 的矩形可以放置在不同纹理的板材中，在赋值相同纹理的板材中保证无特权放置。

2.5. 切割

各种信息有助于确定与待切割矩形和板材边缘相关的切割放置和顺序。

常规信息赋值用于切割的刀具直径：对应 CutterDiameter 设置。
刀具用于切割模式需要的所有切割。

切割有两种：

- **Rip cuts**：切割沿 X 方向穿过面板。
- **Cross cuts**：切割沿 Y 方向穿过面板。

切割模式中赋值的切割根据特定代码区分，以下称*切割等级*：

- **Head cut**：生成 *panel*（0 级）的横切类型：head cut 只能来自*垂直方向*优化
- **Rip cut**：沿面板整个长度生成*条*的纵切（1 级）
  - **Cross cut**：通过截断条生成元素的横切（2 级）
  - **Z**：通过截断以 2 级切割获得面板一部分生成元素的纵切（3 级）
  - **W**：通过截断以 3 级切割获得面板一部分生成元素的横切（4 级）
  - **Cut #5**：通过截断以 4 级切割获得面板一部分生成元素的纵切（5 级）
  - **Cut #6**：通过截断以 5 级切割获得面板一部分生成元素的横切（6 级）。

等级序列匹配基材从纵切转为横切和从横切转为纵切以完成工件的翻转次数。每次翻转面板或条将增加 1 级。可以如下解释，以*水平方向*为例：

- 第一个切割方向为 1 级（纵切）
- 获得的材料条将需要以另一方向切割（2 级，横切）
- 如果需要，可以进一步纵切，划分获得的一部分（3 级）
- 以此类推，直到最多 6 级切割。

2.5.1. 切割等级

从优化获得的模式中的切割序列遵循 MaxCutLevels 属性中赋值的最大切割等级：

- **1 级**：仅 head cut（如果*垂直方向*优化）或 rip cut（如果*水平方向*优化）插入模式

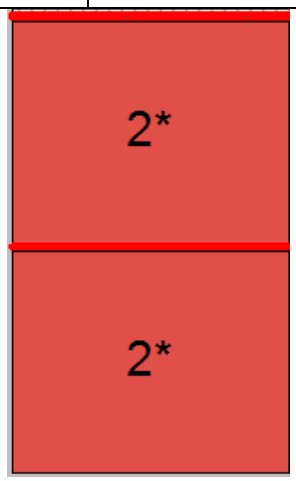
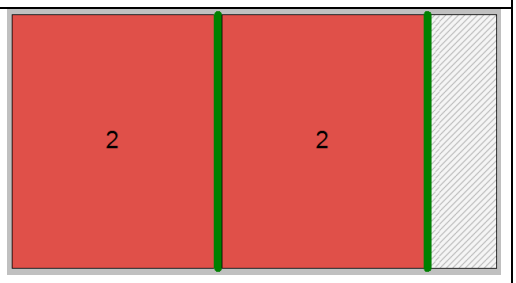
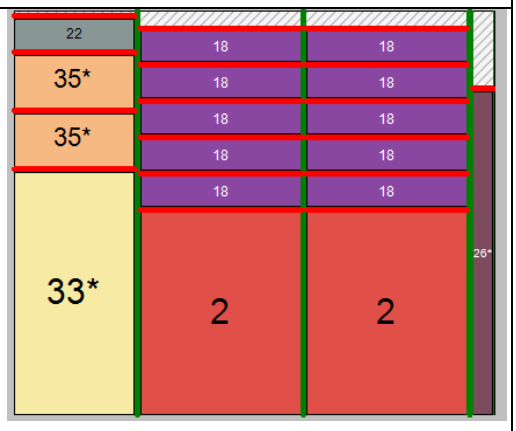
- **2 级**：仅横切插入条
- **3 级**：仅 Z 切插入交叉元素
- **4 级**：仅 W 切插入 z 元素
- **5 级**：W 元素中仅存在 5 级切割
- **6 级**：优化器中不插入超过 6 级的切割

最大切割等级为 6。5 级和 6 级切割没有具体名称。

允许的等级将决定切割模式的复杂程度：

- 限制切割数量将简化切割模式
- 简化切割模式将缩短切割操作时间
- 限制等级数量可导致更多浪费材料。

以下图片对应 MaxCutLevels 属性的不同值：

MaxCutLevels	*Direction*= 水平	*Direction*= 垂直
1		
2		

不同颜色表示不同切割等级：

- **Head cuts**：绿色
- **Rip cuts**（1 级）：红色
- **Cross cuts**（2 级：棕色
- **Z cuts**（3 级）：橄榄绿
- **W cuts**（4 级）：紫色。

2.5.2. 修剪

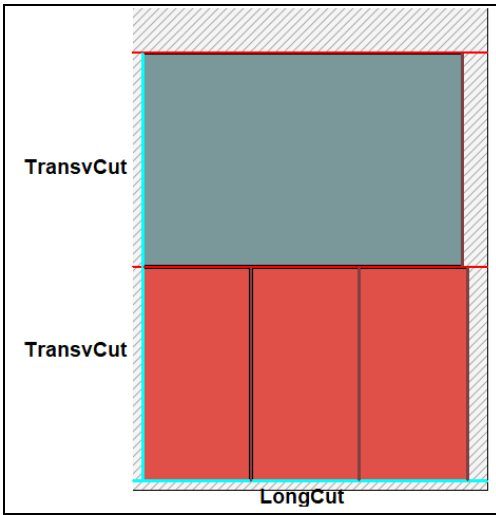
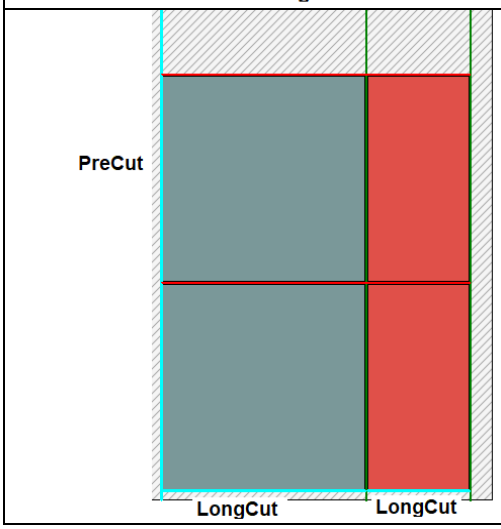
您可以使用以下 *double* 类型属性，根据切割等级添加间隔（报告为修边）：

- *头部修边 (PreCut)*：第一个 0 级切割之前的修边尺寸
- *纵切修边 (LongCut)*：板材第一个纵切之前或预切元素内的修边尺寸
- *横切修边 (TransvCut)*：纵切元素内的第一个横切之前的修边尺寸
- *Z 和 W 修边 (ZCut)*：横切或 Z 切元素内的第一个 Z 或 W 切割之前的修边尺寸。

没有为最后两个切割等级（5 和 6）赋值修边。

字段应用 MaxDimCut 中赋值的最大值。

下图显示前两个切割等级的应用示例，切割对应浅蓝色显示的修边：

	（水平方向；角 0） - 位于板材下部的纵向修边（第一个纵切之前） - 左侧部分有两个横向修边：每个修边在其所属的条的第一个纵切之前 - 在左右部分，应用修边不产生新切割，仅减小赋值数量可使用板材的尺寸：*LongCut* 用于高度，*TransvCut* 用于长度
	（垂直方向；角 0） - 位于板材左侧的预切修边（第一个预切之前） - 位于板材下部的两个纵向修边：每个修边在其所属的面板的第一个纵切之前 - 在左右部分，应用修边减小赋值数量可使用板材的尺寸：*LongCut* 用于高度，*PreCut* 用于长度

在显示的示例中，向材料边缘赋值修边：可使用性对应板材粗糙部分的减少。在此情况下，赋值修边的替代方法是使用边距（参见：板材边距）。

在剩余情况下，将在板材内赋值修边：可使用性对应一般技术理由。

2.5.3. 无张力切割

无张力切割（TFC）功能可以在为板材赋值的第一个切割等级添加额外切割，去除因板材张力导致的“香蕉”效应。

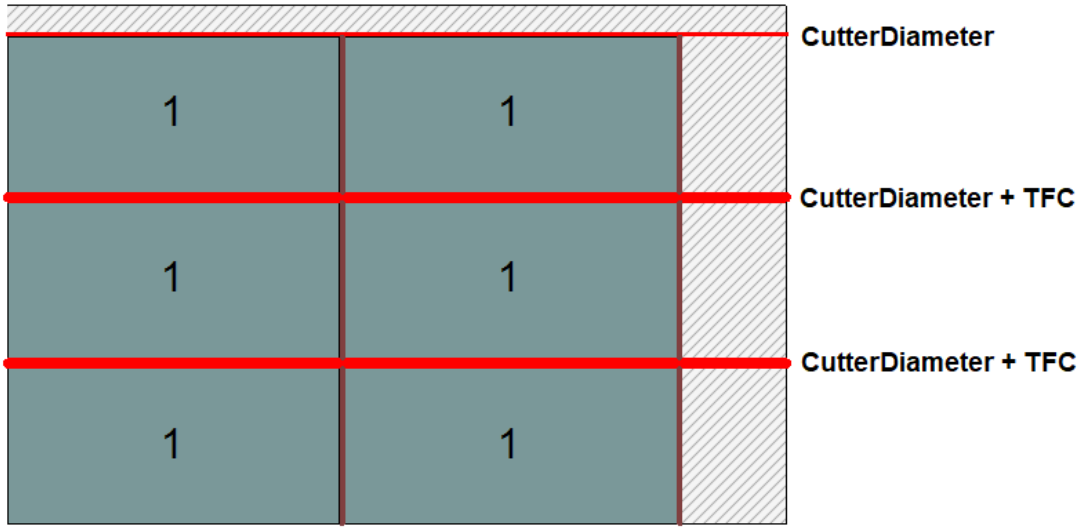
赋值对应 *double* 类型属性 *TensionGap*：值添加到切割直径。

可以根据应用的优化，区分两种情况：

- 如果用 *Horizontal Direction* 执行优化：在纵切添加
- 如果用 *Vertical Direction* 执行优化：在预切添加。

但是，始终为最后一个切割执行添加。

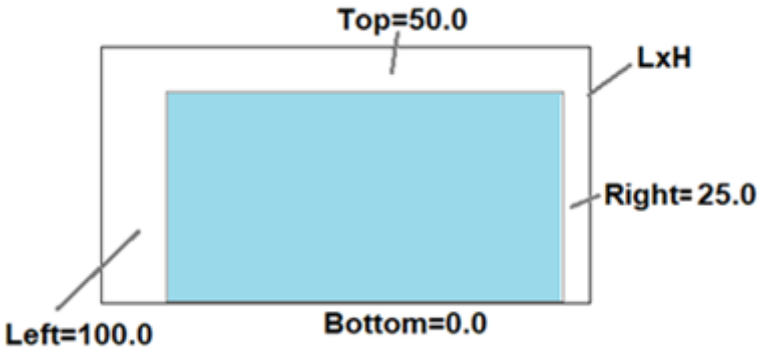
下图显示 TFC 应用示例（水平方向；角 0）：



- 前 2 个切割的厚度相等（切割直径 + TFC）
- 最后一个切割的厚度等于切割直径。

2.5.4. 板材边距

给出*板材*尺寸，可以赋值对放置用途无用的侧边区域，通过侧边区分。下图显示尺寸 LxH 并赋值 4 个不同外边距的板材：对放置有用的区域是内部有颜色的区域。



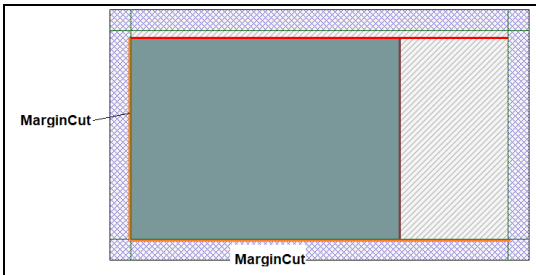
外边距设置对应 OneSheet 结构字段：BorderLeft、BorderRight、BorderTop 和 BorderBottom。
对于所有字段，可以赋值空或正值。
设置边距可以代替管理外部修边。如果二者都赋值，边距和修边管理相加。

2.5.4.1. 切割板材边距

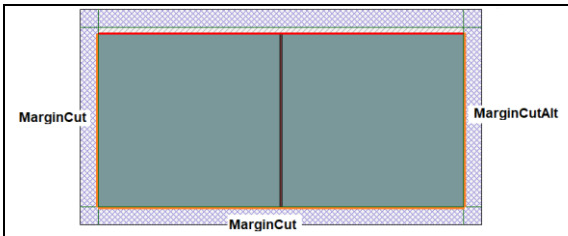
特别选项需要生成边距切割（参见：BorderCut）。

为分离边距产生的切割遵循和板材相同的切割模式，部分代替修边切割。

下图显示赋值所有边距的板材示例，相关切割显示为橙色（水平方向；角 0）：



- 显示的两个切割对应靠近优化顶点（左下角）的边距
  - 水平切割在纵切（红色）之前
  - 垂直切割在横切（棕色）之前
- 不为板材两个相对边生成切割，因为它们由相应主切割（纵向和横向）定义

	相比前一个情况，在板材右侧添加切割： • 切割指示为 *MarginCutAlt* • 生成切割，因为板材右侧现在不由纵切定义

2.5.5. 识别残料和废料

不匹配矩形放置或板材边距的材料部分对应废料区域。TPA_C 可以管理此类区域，并区分可以回收进行后续放置和无法回收的部分：

- 可重复使用区域识别为*残料*，可以成为废料板材
- 其余区域识别为*废料*，丢失用于切割优化器。

可以选择要求激活此具体管理（参见：EnableRestScrap）。

接下来的设置用于赋值将区域识别为残料的标准：

- (RestLength, RestHeight)：要视为残留，废料区域必须具有的最小长度和高度尺寸
- EnableRestArea：允许考虑废料区域的属性
  - RestDim：废料必须具有的最小面积值
  - RestArea：废料必须具有的最小大小（两个尺寸）

根据所有赋值的设置，如果满足以下标准，可以将废料识别为可回收区域：

- 废料大于等于 (*RestLength*, *RestHeight*)
- 如果 *EnableRestArea*=true：两个尺寸大于等于 *RestDim*，废料面积大于等于 *RestArea*。

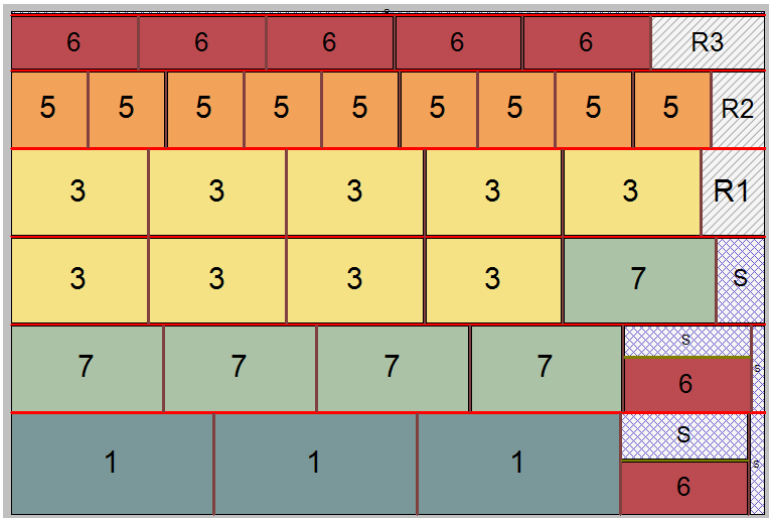
合适方法允许识别废料区域的位置和大小。

如果 *EnableRestScrap=false*：所有废料区域作为废料处理。

客户决定如何对待、分类和利用 TPA_C 识别的废料区域，无论识别为残料还是废料。具体来说：库和/或客户识别为可重复使用的区域可以作为废料板材添加到项目板材列表（参见 OneSheet 结构的 *Scrap* 字段）。

图片显示一个具有一些废料区域的板材：

- (R1, R2, R3) 对应可重复使用区域（残料）
- 标记有字母 (S) 的区域对应废料



2.6. TPA_C 中的优化

TPA_C 优化的主要目的是最佳整体利用具有请求放置（*矩形*）和设定条件（*常规设置*）的可用材料（*板材*）。

优化运行一定数量的*尝试*，选择提供最佳利用即*最佳求解* 的尝试结果。
尝试序列修改特定*参数*，生成不同放置情况。
我们通常定义为*参数* 的性质在于：

- 更改板材的填充逻辑
- 更改矩形的放置顺序
- 更改矩形旋转
- 更改切割方向（水平或垂直；仅当启用 Direction 属性时）。

但是如何判断更好的求解？

我们来看一些非常通用的标准：

- 最大的放置区域优先。工件占据 93.0% 板材的求解优于 88.00% 填充的求解；
- “最多可重复使用”求解优先（意味着足以重复使用的单个废料区域）；
- 还可以应用工件排列，放置工件数量和尺寸，内部废料等其他评估标准。

优化过程得出可重复确定的求解定义，保持可影响其形成的所有边界设置不变。

2.6.1. 渐进优化

优化器可以计算并提供更多求解，最多 5 个。
所有提出的求解使用相同数量的板材。

所有计算的优化保持可用直到：

- 新总体优化（参见函数：Compute）
- 总初始化请求（参见函数：ClearAll）。

2.6.2. 优化标准和优先级

提出优化请求后，TPA_C 开始分析赋值列表和下一个优化步骤。
列表分析可能遇到错误，可导致取消优化。

我们来看一看分析的第一步，区分矩形和板材。

检查*矩形* 列表：

- 未启用的矩形不优化（*OneRect* 结构中的 *Enable* 字段）
- 具有空请求数量（*N* 字段）和额外（*Extra* 字段）的矩形不优化
- 一个或两个尺寸赋值 < 1.0 mm 的矩形不优化
- 不匹配任何有效*板材* 的矩形不优化（示例：*Material=*”abc”的矩形，没有具有相同设置的板材）

检查*板材* 列表：

- 未启用的板材不优化（*OneSheet* 结构中的 *Enable* 字段）
- 具有空可用数量（*N* 字段）的板材不优化
- 一个或两个尺寸 < 1.0 mm 的板材不优化
- 不匹配任何有效*矩形* 的板材不优化（示例：*Material=*”abc”的矩形，没有具有相同设置的矩形）。

报告的结果必须能够为每个矩形和板材列表使用至少一个元素。
开始优化过程，应用特定标准，赋值使用矩形和板材的顺序。

2.6.2.1. 使用板材

板材使用遵循一个顺序，可以考虑不同情况。

现在我们看一看应用于板材初始排序的主要标准。根据给定顺序应用点，如果无法在上一个点之上选择，则移动至下一个点：

- 优先排序标记为废料的板材（*OneSheet* 结构中的 *IsScrap* 字段）
- 升序排序（0 除外，最后运行）
- *OrderSheetDim* 常规设置允许按尺寸排序板材（降序面积排序）
- 升序类型排序（*OneSheet* 结构中的 *ID* 字段），方式和之前点的其他条件相同。

UseBeforeScrap 常规设置允许应用 *sheet* 量化字段应用为废料。

2.6.2.2. 使用矩形

现在我们看一看应用于矩形初始排序的主要标准。根据给定顺序应用点，如果无法在上一个点之上选择，则移动至下一个点：

- 升序排序（0 除外，最后运行）
- 赋值请求数量（正）的矩形。即：仅需要额外放置的矩形后排序
- 降序面积排序
- 无法旋转的矩形
- 更大请求数量的矩形
- 升序类型的部件（*OneRect* 结构的 *ID* 字段）

额外放置

可以为矩形赋值额外放置，作为额外或实际请求的替代。也就是说，对于矩形，可以额外赋值或仅赋值额外放置。
信息在 OneRect 结构的 *Extra* 字段赋值：正值直接赋值额外放置数量。

仅当检查无法放置请求的矩形后，在板材上使用额外放置。因此很明显，在任何情况下，无法使用板材仅应用额外放置。

2.7. 管理切割项目的支持

切割项目理解为赋值给 TPA_C 库的所有信息集：
- 整体启用设置
- 矩形、板材列表。

TPA_C 公开项目序列化方法。这些方法尤其对以下有用：
- 测试情况
- TPA_C 集成情况，此处列表赋值结构中提供的信息足够。

2.8. 库的已知限制

赋值矩形
- 最多可以赋值 500 个不同*矩形*
- 对于每个*矩形*，可以赋值最大可放置数量 999。

赋值板材
- 最多可以赋值 100 个不同*板材*
- 对于每个*板材*，可以赋值的最大可使用数量为 999。

3. 库函数指南

本章详细介绍 TPA_C 的属性和函数。

3.1. 许可证管理

3.1.1. ExpirationDateString

字符串属性，用于返回许可证的到期日期。日期格式依据计算机的区域文化设置进行格式化。
当识别出的是有时限的许可证时，该属性的返回值具有意义（例如：不为空字符串）。
如果返回的是空字符串，可能表示许可证尚未验证，或为无限期许可证。

3.1.2. IsValidLicense

布尔类型属性，用于检测许可证的存在与有效性。

数值
如果启用模块则为 *True*，否则为 *False*。

注释
属性查询作为实际密钥读取运行，但仅当没有切割优化运行时。
如果密钥无效，则无法执行任何优化。

3.2. 常量

TpaCutOEM.TpaRctCut 类提供以下常量：

MAX_ROW_ITEMS	500	矩形类型最大数量
MAX_ROW_N	999	为一个矩形请求的放置最大值
MAX_SHEET_ITEMS	100	板材类型最大数量
MAX_SHEET_N	999	一个板材可用的放置最大值
MAX_CUSTOM_SETTINGS	10	项目常规设置最大数量
MAX_LL_SETTINGS	250	项目单个常规设置最大长度（字符串长度）
MAX_ROW_PARAMS	15	单个矩形中的常规设置最大数量
MAX_LL_PARAMS	50	矩形单个常规设置最大长度（字符串长度）
MAX_CUT_LEVELS	6	切割等级最大数量

3.3. 枚举

TpaCutOEM 命名空间提供以下枚举。

3.3.1. ModeOrder

赋值分类切割方案中的条或条部分的不同标准

- *ModeHeight*：应遵守的条顺序
  - 条高度降序值
  - 条末端废材料的升序值（相同高度）
- *ModeScrap*：应遵守的条顺序
  - 部分末端的废材料升序值
  - 部分内的废材料升序值（以前的值相等）。

3.3.2. CutError

库管理的错误赋值：

- *ErrorNone*：无错误
- *ErrorKey*：未验证许可证
- *ErrorMemory*：内存错误
- *ErrorRectEmpty*：空矩形列表或没有启用或可放置的矩形
- *ErrorSheetEmpty*：空板材列表或没有启用板材
- *ErrorNoMatch*：没有板材列表相应匹配
- *ErrorIOfile*：管理项目文件出错（路径和/或文件访问错误…）
- *ErrorFileNotValid*：管理项目文件出错（非有效格式）
- *ErrorNoSolutions*：没有赋值求解
- *ErrorUnexpected*：常规或未管理错误（例如，无效赋值索引）。

3.4.结构

TpaCutOEM 命名空间提供以下结构。

3.4.1. OneRect

矩形 常规赋值。结构公开将字段初始化为默认值的方法。

- *int ID*：*矩形* 的数字标识符（单义，严格为正）{*默认* = 0}
- *bool Enable*：　启用矩形使用（false = 不放置矩形）{*默认* = true}
- *string Label*：　矩形的描述名称（可以 =""；最多 50 个字符）{*默认* =""}
- *double Length*：　长度 (>= 0.0) {*默认* = 0.0}
- *double Height*：　高度 (>= 0.0) {*默认* = 0.0}
  - 值采用以下单位：Unit
- *double Thickness*：　厚度 (>= 0.0) {*默认* = 0.0}。
  - 如果需要匹配相应 *sheet* 字段，为字段赋值不同值
  - 值采用以下单位：Unit
- *int N*：　放置请求数量 (>= 0) {*默认* = 0；最大值 = 999}
- *int Extra*：　额外可用数量（如果 > 0 则有效） {*默认* = 0；最大值 = 999}
  - *N* 字段设置为矩形请求的数量。*Extra* 字段设置额外数量。仅当尝试放置所有 *rectangle* 类型请求数量后，插入额外工件。
- *string Material*：　材料 {*默认* = “”}
  - 如果需要匹配相应 *sheet* 字段，为字段赋值不同值
  - 字段的真实含义取决于外部应用
- *int Grain*：　纹理方向（0 = 无；1 = 水平；2 = 垂直）{*默认* = 0}
  - 如果需要匹配相应 *sheet* 字段，为字段赋值 1 或 2。仅当可以观察纹理本身方向（可应用旋转），可以在带纹理的 *板材* 上放置具有字段 (1, 2) 的矩形；不带纹理的矩形适用于具有任何 *Grain* 字段的板材
- *int Priority*：　使用优先级 (>= 0) {*默认* = 0}
  - 具有更低优先级的 *矩形* 在切割模式求解中具有放置优先级
  - 具有优先级 0 的 *矩形* 最后插入
- *bool Rotate*：　旋转（*false* = 不允许；*true* = 允许）{*默认* = *true*}
  - *true* 值启用 90° 逆时针旋转
- *string MatEdge1, MatEdge2, MatEdge3, MatEdge4*：边的材料，在矩形 4 边区分（分别上下左右）{*默认* = “”}
- *double ThickEdge1, ThickEdge2, ThickEdge3, ThickEdge4*：边的厚度，在矩形 4 边区分 {*默认* = 0}
- *bool BoolEdge1, BoolEdge2, BoolEdge3, BoolEdge4*：启用矩形减小边对应的厚度，在矩形 4 边区分，用于切割优化 {*默认* = false}
  - 切割优化器使用边信息对从切割模式获得的矩形尺寸应用可能减小。减小可在矩形每边应用，验证对应信息 (*ThickEdge >0.0, BoolEdge =true)*
  - 边相关字段将在切割优化后封边站使用
  - 仅当 EnableRectEdge=true 时，边相关字段有意义。
- *string Param1, Param2…, Param15*：　不影响优化的 15 个通用参数 {*默认* = “”}

- 字段可用于赋值管理或通常技术信息。示例：订单信息（客户、订单号、注释），材料添加信息。

3.4.2. OneSheet

板材 常规赋值。结构公开将字段初始化为默认值的方法。

- *int ID*：　　*板材* 数字标识符（单义，严格为正）{*默认* = 0}
- *bool Enable*：　　启用*板材* 使用（false = 不放置板材）{*默认* = true}
- *string Label*：　　*板材* 的描述名称（可以 =""；最多 50 个字符）{*默认* =""}
- *bool Scrap*：　　将板材标识为找回（例如以前优化的板材废料）{*默认* = false}
  - 此类型*板材* 优先使用
  - *OrderBeforeScrap* 设置赋值字段应用

- *double Length*：　长度 (>= 0.0) {*默认* = 0.0}
- *double Height*：　高度 (>= 0.0) {*默认* = 0.0}
  - 值采用以下单位：Unit
- *double Thickness*：　厚度 (>= 0.0) {*默认* = 0.0}。
  - 如果需要匹配相应 *rectangle* 字段，为字段赋值不同值
  - 值采用以下单位：Unit

- *int N*：　　可用数量 (>= 0) {*默认* = 0；最大值 = 100}

- *string Material*：　材料 {*默认* = “ ” }
  - 如果需要匹配相应 *rectangle* 字段，为字段赋值不同值
  - 字段的真实含义取决于外部应用
- *int Grain*：　　纹理方向（0 = 无；1 = x = 水平；2 = y = 垂直）{*默认* = 0}
  - 如果需要匹配相应 *rectangle* 字段，为字段赋值 1 或 2

- *int Priority*：　　使用优先级 (>= 0) {*默认* = 0}
  - 较低优先级*板材* 优先使用
  - 优先级 =0 的*板材* 最后使用

- *double Cost*：　　每单位面积的板材成本 {*默认* = 0}
  - 客户端应赋值信息单位，因为其使用委派给客户端
  - 典型单位可以是：€/sq.m.（欧元/平方米），$/ft2（美元/平方英尺）

- *double BorderBottom, BorderTop, BorderRight, BorderLeft*：板材边距 {*默认* = 0}
  - 值采用以下单位：Unit

3.5. 函数定义

我们来看看如何赋值 Tpa_C 的所有函数：对于函数，也使用设置一词。
初始化 TPA_C 时，所有设置赋值为默认值，以下指示其位置 {*默认* = ⋯}。

为了提供更综合的框架，赋值分为四组：

- *常规函数*：常规使用赋值
- *对应切割优化常规设置的函数*：指项目常规优化和切割优化的赋值。这种信息通常保持不变，不序列化为项目对应文件
- *切割项目常规赋值相关函数*：指项目特定赋值的赋值。这种信息通常序列化为项目对应文件。

3.5.1. 常规函数

3.5.1.1. Version

String 类型属性，返回库版本。
字符串报告库的主要、次要、版本和修订编号。

3.5.1.2. LastError

CutError 类型属性，返回上次遇到的错误。在可以诊断异常情况的所有过程中更新该值。

3.5.1.3. ErrorMessage

函数返回所指示错误的赋值消息。

string ErrorMessage (CutError Error)

参数

- *Error*： 枚举值

返回值

如果为 *Error = CutError.ErrorNone*，函数返回为 ""（空字符串）。
内部消息采用英语赋值。

3.5.2. 对应切割优化常规设置的函数

此信息不会随项目更改而更改：与通常归属于应用常规操作选项的设置有关。

3.5.2.1. NumberCustom

Integer 类型属性，赋值/返回项目常规设置 {*默认* = MAX_CUSTOM_SETTINGS, min=0, max = MAX_CUSTOM_SET-TINGS}。
值可以限制 TPA_C 管理的项目常规设置数量（例如：向文件读取和/或记录项目时）。

3.5.2.2. NumberRectParam

Integer 类型属性，赋值/返回每个 OneRect 的参数数量 {*默认* = MAX_ROW_PARAMS, min=0, max = MAX_ROW_PARAMS}。
值可以限制 TPA_C 管理的项目常规设置数量（例如：向文件读取和/或记录项目时）。

3.5.2.3. EnableRectEdge

Boolean 类型属性，赋值/返回激活以赋值矩形边 {*默认* = true}。
具体来说：

- *true*：由于应用边，优化器可以使用小矩形
- *false*：不序列化矩形边相关字段（项目文件上）。

3.5.2.4. OrderSheetDim

Boolean 类型属性，赋值/返回启用以按尺寸排序板材 {*默认* = false}。
值 =*true*：启用优化，按面积降序排序板材。

3.5.2.5. OrderBeforeScrap

Boolean 类型属性，赋值/返回启用先使用废料标记板材 {*默认* = false}。

值 =*true*：启用优化，在其他板材之前使用废料板材。

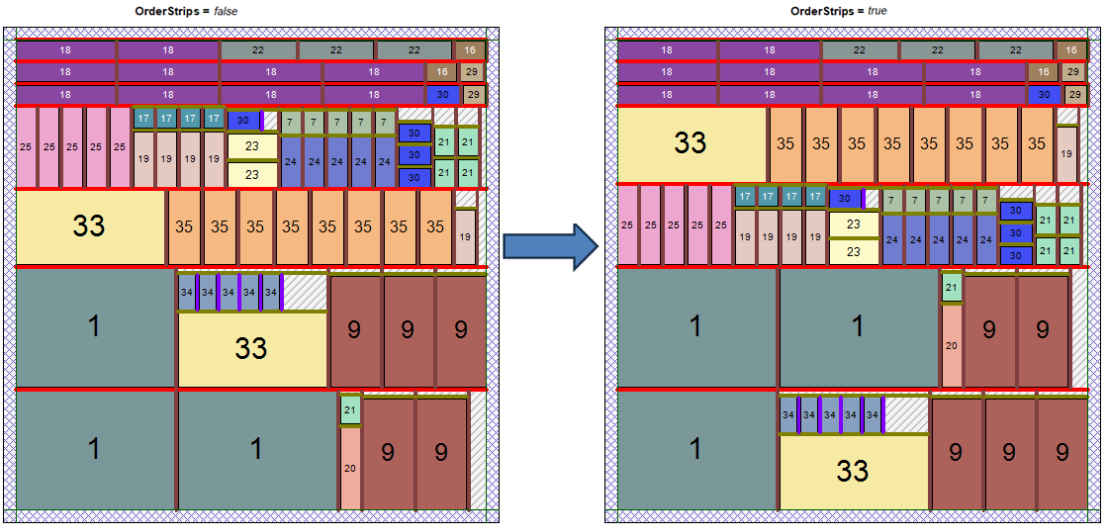
3.5.2.6. OrderStrips

Boolean 类型属性，赋值/返回启用排序切割模式中的条 {*默认* = true}。

如果启用，条排序应用 *ModeOrderStrips* 属性中赋值的标准。

请注意，条对应纵切获得的面板的一部分（切割等级 1）。

下图显示两种情况下获得的同一切割模式：



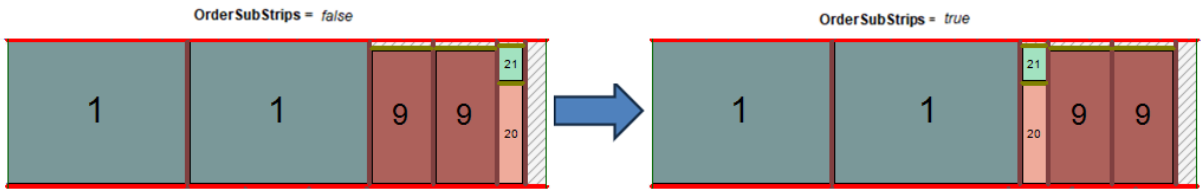
- *false*：（左）以计算顺序报告条
- *true*：（右）条顺序应遵守对应 *ModeOrderStrips*=*ModeOrder.ModeHeight* 的标准：
  - 将值减小条高度
  - 增加条末端浪费材料的值（相同高度）。

3.5.2.7. OrderSubStrips

Boolean 类型属性，赋值/返回启用排序条或切割模式条部分内的工件 {*默认* = true}.

如果启用，排序将应用 *ModeOrderSubStrips* 属性中赋值的标准。

下图显示两种情况下获得的同一条：



- *false*：（左）以计算顺序列出条切割
- *true*：（右）条每个部分的顺序应遵守对应 *ModeOrderSubStrips* =*ModeOrder.ModeScrap* 的标准：
  - 增加部分末端浪费材料的值
  - 增加部分内浪费材料的值（之前相同值）。

3.5.2.8. ModeOrderStrips

ModeOrder 类型属性赋值/返回切割方案中的条排序标准 {*default* = *ModeOrder.ModeHeight*}。

3.5.2.9. ModeOrderSubStrips

ModeOrder 类型属性赋值/返回切割方案的条或条部分内的工件排序标准 {*default* = *ModeOrder.ModeScrap*}。

3.5.2.10. MaxDimCut

Double 类型属性，赋值/返回可应用为修边切割或张力释放（TFC）尺寸的最大值 {*单位*: mm；*默认* = 100.0；*有效范围*：>= 0.0}。

3.5.2.11. BorderCut

Boolean 类型属性，赋值/返回在板材边缘生成切割的启用 {*默认* = false}。
对于值 *true*：仅生成必要切割，无法被其他切割类型设为无用（例如修边、头切、纵切…）。

3.5.2.12. EnableRestScrap

Boolean 类型属性赋值/返回激活识别废料区域的启用 {*default* = false}。
对于 *true* 值：可以获得与可重复使用区域（残料）和丢失区域（废料）相关的信息。
以下属性指出应用于识别可重复区域的标准：*RestLength*、*RestHeight*、*EnableRestArea*、*RestDim*、*RestArea*。
对于 *false* 值：所有废料区域作为丢失（废料）处理。

3.5.2.13. RestLength

Double 类型属性赋值/返回废料识别为可回收区域（残料）必须具有的最小长度 {*unit*: mm; *default* = 50.0; *validity range*: >= 0.0}。

3.5.2.14. RestHeight

Double 类型属性赋值/返回废料识别为可回收区域（残料）必须具有的最小高度 {*unit*: mm; *default* = 50.0; *validity range*: >= 0.0}。

3.5.2.15. EnableRestArea

Boolean 类型属性赋值/返回使用废料区域信息的启用 {*default* = false}。
对于 *true* 变量：使用 *RestDim*、*RestArea* 属性。

3.5.2.16. RestDim

Double 类型属性赋值/返回废料识别为可回收区域（残料）必须具有的最小长度和高度 {*unit*: mm; *default* = 50.0; *validity range*: >= 0.0}。

3.5.2.17. RestArea

Double 类型属性赋值/返回废料识别为可回收区域（残料）必须具有的最小面积 {*unit*: mm^2; *default* = 2500.0; *validity range*: >= 0.0}。

3.5.3. 项目常规赋值相关函数

此信息可随着项目更改而更改。

3.5.3.1. Unit

Integer 类型属性，返回函数赋值的单位 {*默认* = 0}。

3.5.3.2. CutterDiameter

Double 类型属性，赋值/返回切割刀具直径 {*单位*：Unit；默认 = 0.0；*有效范围*：>= 0.0}。
矩形 之间放置应用匹配设定值的最小间距。

3.5.3.3. Direction

Short 类型属性，赋值/返回第一个应用于板材的切割等级的方向 {*默认* = 0；*有效范围*：0/1/2}：

- 0 = 水平方向
- 1 = 垂直方向
- 2 = 优化器确定的方向。

对于值 2，优化器将始终为每个板材类型搜索最佳切割模式。在此情况下：作为优化的结果，一些板材的第一个切割方向匹配长度尺寸，其他板材的第一个切割方向匹配高度尺寸。

3.5.3.4. Corner

Short 类型属性，赋值/返回有效值中的放置起始顶点 {*默认* = 0； *有效范围*：0/3}

- 0 = 左下
- 1 = 左上
- 2 = 右下
- 3 = 右上。

3.5.3.5. MaxCutLevels

Integer 类型属性，赋值/返回可在优化中使用的最大切割等级 {*默认* = MAX_CUT_LEVELS; min = 1; *max* = MAX_CUT_LEVELS}。

值 1 对应只能执行一个切割等级的可能：

- 值 0 (head cut)，垂直方向优化情况
  - 需要和原始板材相同高度的矩形
- 值 1（纵切），水平方向优化情况
  - 需要和原始板材相同长度的矩形。

3.5.3.6. TensionGap

Double 类型属性，赋值/返回为板材赋值的第一个切割等级的额外尺寸，以释放板材张力 {*单位*：Unit；默认 = 0.0； *有效范围*：>= 0.0, <=MaxDimCut}。

3.5.3.7. PreCut

Double 类型属性，赋值/返回第一个头切分离面板前添加的垂直修边（0 级切割）{*单位*：Unit；默认 = 0.0； *有效范围*：>= 0.0, <= MaxDimCut}。

3.5.3.8. LongCut

Double 类型属性，赋值/返回第一个 1 级切割标识的第一个条之前添加的水平修边 {*单位*：Unit；默认 = 0.0； *有效范围*：>= 0.0, <= MaxDimCut}。

3.5.3.9. TransvCut

Double 类型属性，赋值/返回第一个 2 级切割分离部分之前添加的垂直修边 {*单位*：Unit；默认 = 0.0； *有效范围*：>= 0.0, <= MaxDimCut}。

3.5.3.10. Zcut

Double 类型属性，赋值/返回第一个 3 或 4 级切割分离部分之前添加的修边 {*单位*：Unit；默认 = 0.0； *有效范围*：>= 0.0, <= MaxDimCut}。

3.5.3.11. Custom1, Custom2, …, Custom10

String 类型属性，赋值/返回不影响优化过程的普通项目信息 {默认 = “”；属性最大数量 = NumberCustom；参数最大长度 = MAX_LL_SETTINGS}。

3.5.3.12. ClearAll

函数清除项目列表（矩形，板材）和任何计算的求解。

bool ClearAll ()

返回值

如果结果为正，则为 *true*。

当前没有任何情况可以确定返回为 *false*。

3.6.矩形定义

3.6.1. AddRct

函数将矩形添加到列表。

bool AddRct (OneRect Item)

参数

- *Item*： 矩形赋值结构

返回值

如果结果为正，则为 *true*，如果未插入矩形，则为 *false*

注释

返回 *false* 对应以下错误情况之一：

- 矩形列表已经达到最大允许（500 个元素）
- 矩形赋值了一个无效标识符（***Item.ID*** 必须严格为正）
- 已经赋值具有 ***Item.ID*** 数字标识符的矩形
- 矩形具有无效尺寸或无效边尺寸。具体来说：从矩形尺寸减去边厚度，将一个或两个尺寸减小至 < 1.0 mm 的值。

3.6.2. ClearRct

该函数清除矩形列表。

bool ClearRct ()

返回值

如果正确删除列表则为 True

3.6.3. CountRect

Int 类型属性，获取列表中的矩形数量。

3.6.4. ReadRect

该函数搜索匹配指定 ID 的矩形。

CutError ReadRect (*int ItemID, ref *OneRect Item)

参数

- *ItemId*： 矩形标识符 (> 0)
- *Item*： 矩形赋值结构

返回值

如果结果为正，则为 *CutError.ErrorNone*。

注释

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorUnexpected*) 赋值无效标识符（***ItemID*** 必须严格为正），或没有赋值具有 ***ItemID*** 数字标识符的矩形。

3.6.5. ReadRectIndex

该函数搜索匹配指定索引的矩形。

CutError ReadRectIndex (*int Index, ref *OneRect Item)

参数

- *Index*： 矩形列表上的索引（从 0 开始）(>= 0)

- *Item*：　矩形赋值结构

返回值
如果结果为正，则为 *CutError.ErrorNone*

注释
该函数的主要用途是执行 ***LoadProject*** 函数后获取矩形。

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：
- (*CutError.ErrorUnexpected*) 赋值了无效索引。

3.7.板材定义

3.7.1. AddSheet

函数将板材添加到列表。

bool AddSheet (OneSheet Item)

参数
- *Item*：　板材赋值结构

返回值
如果结果为正，则为 *true*

注释
返回 *true* 以外的值对应以下错误情况之一：
- 板材列表已经达到允许最大值（100 个元素），或结构中请求的数量超过允许最大值 (999)
- 板材赋值了一个无效标识符（***Item.ID*** 必须严格为正）
- 已经赋值具有 ***Item.ID*** 数字标识符的板材
- 板材赋值无效边距。具体来说：从板材尺寸减去边距，将一个或两个尺寸减小至 < 1.0 mm 的值

3.7.2. ClearSheet

该函数清除板材列表。

bool ClearSheet ()

返回值
如果结果为正，则为 *true*

3.7.3. CountSheet

Int 类型属性，获取列表中的板材数量。

3.7.4. ReadSheet

该函数搜索匹配指定 ID 的板材。

CutError ReadSheet (*int ItemID, ref *OneSheet Item)

参数
- *ItemId*：　板材标识符 (> 0)
- *Item*：　板材赋值结构

返回值
如果结果为正，则为 *CutError.ErrorNone*。

注释

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorUnexpected*) 赋值无效标识符（***ItemID*** 必须严格为正），或没有赋值具有 ***ItemID*** 数字标识符的板材

3.7.5. ReadSheetIndex

该函数搜索匹配指定索引的板材。

CutError ReadSheetIndex (*int Index, ref *OneSheet Item)

参数

- *Index*： 板材列表上的索引（从 0 开始）(>= 0)
- *Item*： 板材赋值结构

返回值

如果结果为正，则为 *CutError.ErrorNone*。

注释

该函数的主要用途是执行 ***LoadProject*** 函数后获取板材。

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorUnexpected*) 赋值了无效索引。

3.8. 切割求解

3.8.1. Compute

该函数启动优化过程

bool Compute ()

返回值

如果结果为正，则为 *true*。

注释

如果返回 *false*，可以查询 *LastError* 属性，了解阻碍优化的原因：

- (*CutError.ErrorKey*) 密钥存在和状态检查失败
- (*CutError.ErrorRectEmpty*) 矩形列表为空，或未赋值启用项
- (*CutError.ErrorSheetsEmpty*) 板材列表为空，或未赋值启用项
- (CutError.ErrorNoMatch) 部件和板材列表没有公共匹配组

启动时，函数：

- 运行初步检查（如上所述）
- 检查矩形和板材列表。

如果执行列出的检查导致错误，函数仍返回，不执行优化。

3.9. 优化结果

如果未计算有效优化，此处列出的所有属性和函数将失败。

3.9.1. NumberOfSolutions

Integer 类型属性，返回计算求解总数：

- 0： 未计算任何求解
- >0： 计算求解数量

3.9.2. SelectSolution

Integer 类型属性，赋值/返回 *正确求解* 编号：

- -1：　　未计算任何求解
- >=0：　当前求解匹配用 ***Compute()*** 函数调用计算的求解

赋值后：

- 属性允许确认或更改 *当前求解*
- 如果值无效，则不赋值。

3.9.3. SolutionSheets

Integer 类型属性，返回 *当前求解* 的板材数量：

- 0：求解为空或不存在
- >0：　　求解有效

求解板材也称：*切割模式*。

返回值不考虑任何重复板材。

3.9.4. GetSheetId

函数返回板材 ID，给定其在 *当前求解* 中的索引。

Int GetSheetID (int idx)

参数

- *idx：*　　板材索引 (>= 0)

注释

如果索引或求解无效，函数返回 0。

3.9.5. NumberOfRepetitions

函数返回特定切割模式在 *当前求解* 中重复的次数。

int NumberOfRepetitions (int idx)

参数

- *idx：*　　板材索引 (>= 0)

注释

运行优化后，属性值有效。
如果索引或求解无效，函数返回 0。
如果仅赋值一次切割方案（无模式），函数返回 1。

3.9.6. SheetDirection

函数返回优化板材中应用的方向，给出当前求解中的索引。

int SheetDirection (int idx)

参数

- *idx:*　　板材索引 (>= 0)

注释

对于无效索引或求解，函数返回 -1

函数的使用性在未赋值优化方向情况下 (Direction = 2)。

3.9.7. UsedSheets

函数读取*当前求解*的板材数量（带放置的板材）。

int UsedSheets (int ID_Sheet)

参数

- *ID_Sheet*：　　板材标识符 (>0)

注释

函数返回当前求解或板材类型的数量：

ID_Sheet	
=0	获得为求解计算的板材总数
>0	获得为求解计算的板材类型数量 (*ID_Sheet*)

3.9.8. UsedRects

函数读取当前求解板材中或整个当前求解的放置数量，可以选择仅匹配一种矩形。

int UsedRects (int Index, int ID_Part)

参数

- *Index*：　从 0 开始的求解板材索引
- *ID_Part*：　　矩形标识符 (> 0)

注释

函数读取对应赋值参数的放置数量：

索引	*ID_Part*	
-1	0	获得为求解计算的放置总数
-1	>0	获得所有求解板材中匹配矩形 *(ID_Part*) 的放置数量
>=0	0	获得求解的索引板材 *(Index*) 中的放置总数
>=0	>0	获得匹配索引板材 *(Index*) 中矩形 (*ID_Part*) 的放置数量

对于负 *Index* (-1) 的操作，返回值考虑相同重复板材。
对于 *Index* >= 0 的操作，返回值考虑单个板材放置，不考虑板材的任何重复。

3.9.9. FitnessSheet

该函数返回*当前求解*中的板材效率信息。

double FitnessSheet (int idx)

参数

- *idx*：　　板材索引 (>= 0)

注释

运行优化后，属性值有效。
总效率计算为用于放置的面积与所使用板材总面积的比值 %。如果求解生成多个板材，上一个板材的效率求解可以适当限制可使用的放置区域
该函数返回作为一个板材或所有当前求解百分比的健康程度：

idx	
= -1	获得整个求解的健康程度（考虑重复）
>=0	获得 idx 索引板材的健康程度

3.9.10. ReadResolRect

该函数返回*当前求解*中的板材放置信息

int ReadResolRect (int idx, int idxrct, ref double qx, ref double qy, ref bool rotate)

参数

- *idx*： 求解板材的索引（从 0 开始）
- *idxrct*： 求解板材上的放置索引（从 0 开始）
- *qx*： 矩形左下角的 x 坐标
- *qy*： 矩形左下角的 y 坐标
- *rotate*： 指示是否旋转矩形（*true* = 旋转）

返回值

正结果情况下读取其数据的矩形标识，否则为 0。

注释

如果返回 0，可以查询 LastError 属性，找出错误：

- (*CutError.ErrorKey*) 密钥存在和状态检查失败
- (*CutError.ErrorNoSolutions*) 找不到求解
- (*CutError.ErrorUnexpected*) 一个或两个索引 (*idx, idxrct)* 无效

3.9.11. NumberOfCuts

该函数返回为匹配指定索引的*当前求解*板材计算的切割数量

int NumberOfCuts(int idx)

参数

- *idx*： 求解板材的索引（从 0 开始）

返回值

切割模式中的切割数量。如果索引或求解无效，函数返回 0。

3.9.12. ReadResolCut

该函数返回*当前求解*中的切割放置信息

int ReadResolCut (int idx, int idxcut, ref double xStart, ref double yStart, ref double xEnd, ref double yEnd, ref int orientation, ref double thickness)

参数

- *idx*： 求解板材的索引（从 0 开始）
- *idxcut*： 求解板材上的切割索引（从 0 开始）
- *xStart*： 切割起点的 x 坐标
- *yStart*： 切割起点的 y 坐标
- *xEnd*： 切割终点的 x 坐标
- *yEnd*： 切割终点的 y 坐标
- *orientation*： 切割方向（0 = 垂直，1 = 水平）
- *Thickness*： 切割厚度

返回值

切割类型（出错时为 -1）：

- 0：头切
- 1, 2, …：纵切，横切, …（最高切割水平）
- 10：头部修边
- 11：纵切修边
- 12：横切修边
- 13：Z 修边
- 14：W 修边
- 20：预切边缘切割

- 21：撕裂边际削减
- 22：横切边距
- 30：头切边距的最终切口
- 31：裁边
- 32：最后切割横边

注释

如果返回 -1，可以查询 *LastError* 属性，找出错误：

- (*CutError.ErrorKey*) 密钥存在和状态检查失败
- (*CutError.ErrorNoSolutions*) 找不到求解
- (*CutError.ErrorUnexpected*) 一个或两个索引 (*idx, idxcut*) 无效

函数返回指示 *切割等级*，范围在 0 到 6。
切割坐标始终采用纯笛卡尔坐标系（原点位于左下角）。
thickness 参数可以赋值与切割刀具厚度不同的厚度：

- *更小的值*，如果切割部分在材料可使用区域外
- *更大的值*，如果重复切割（例如：对于添加 TFC 类型切割）。

切割序列遵循将板材分隔为条，将条分隔为更多部件：序列遵循在原始面板上排列切割的顺序，从用于放置的顶点开始向相对顶点。

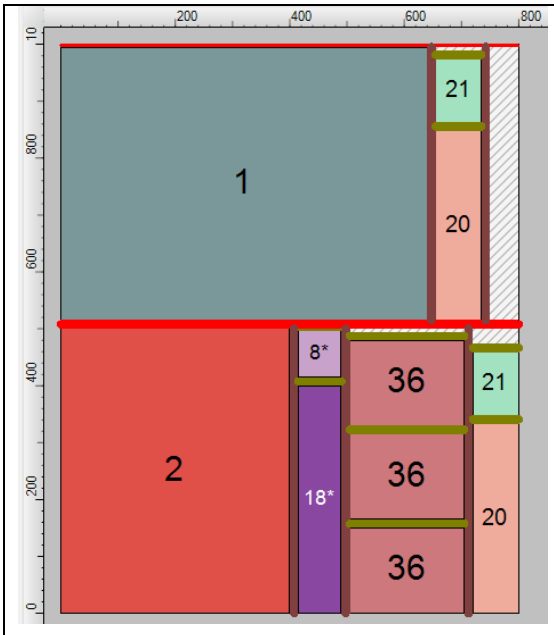
可以赋值修边。

如果 BorderCut =*true*，赋值分离板材边缘的切割。

下面是水平方向优化获得的切割模式示例：

- 用于放置的顶点是 *left-bottom*
- 有 2 个纵切（颜色切割：红色）
- 最大切割等级为 3（Z 切割）。

返回的切割序列对应图片右侧的序列。
板材侧面的标尺有助于在图上定位切割。
原始板材的尺寸如下：800 x 1000 mm。



纵切：[0;500] →[800;500]
横切：[400;0] →[400;500]
横切：[490;0] →[490;500]
Z 切: [415;400] →[490;400]
Z 切: [415;497] →[490;497]
横切：[705;0] →[705;500]
Z 切: [505;150] →[705;150]
Z 切: [505;315] →[705;315]
Z 切: [505;480] →[705;480]
横切：⋯ 未赋值
Z 切: [720;333] →[800;333]
Z 切: [720;459] →[800;459]

纵切：[0;995] →[800;995]
横切：[640;515] →[640;995]
横切：[735;515] →[735;995]
Z 切: [655;848] →[735;848]
Z 切: [655;974] →[735;974]

为第一个*纵切* 报告的最后一个*横切* 指示为“未赋值”，因为其完全在板材右边缘以外。
板材顶部的*纵切* 明显比第一个薄，因为与板材顶边重叠。

3.9.13. CutLinear

该函数返回为指定索引的*当前求解* 板材计算的切割数量对应的线性延伸

double CutLinear(int idx)

参数

- *Idx*：　　求解板材的索引（从 0 开始）

返回值

该函数对于无效索引或求解返回 0；否则，返回切割的整体线性延伸。

3.9.14. CutArea

该函数返回为指定索引的*当前求解*板材计算的切割数量对应的面积

double CutArea(int idx)

参数

- *idx*：　　求解板材的索引（从 0 开始）

返回值

该函数对于无效索引或求解返回 0；否则，返回切割占据的面积。
应用 ReadResolCut 函数返回的切割厚度，计算面积。

3.9.15. NumberOfRemains

函数返回*当前求解*指定索引板材找到的残料数量

int NumberOfRemains (int idx)

参数

- *idx*：　求解板材的索引（从零开始）

返回值

切割方案中的残料数量。如果索引或求解无效，函数返回 0。

3.9.16. ReadResolRemain

函数返回*当前求解 板材上的残料区域信息*

bool ReadResolRemain (int idx, int idxrst, ref double length, ref double height, ref double qx, ref double qy)

参数

- *idx*：　　求解板材的索引（从零开始）
- *idxrct*：　求解板材上的区域索引（从零开始）
- *length*：　区域长度
- *height*：　区域高度
- *qx*：　　区域左下角的 x 坐标
- *qy*：　　区域左下角的 y 坐标

返回值

如果结果为正，则为 true，否则为 *false*。

注释

如果返回 =*false*， 可以查询 *LastError* 属性查找错误：

- *(CutError.ErrorKey)* 密钥存在和状态检查失败
- *(CutError.ErrorNoSolutions)* 找不到求解
- *(CutError.ErrorUnexpected)* 一个或两个索引 *(idx, idxrst)* 无效

3.9.17. NumberOfScraps

函数返回*当前求解*指定索引板材找到的废料区域数量

int NumberOfScraps (int idx)

参数

- *idx*：　　求解板材的索引（从零开始）

返回值

切割方案中的废料区域数量。如果索引或求解无效，函数返回 0。

3.9.18. ReadResolScrap

函数返回*当前求解板材上的废料区域信息*

bool ReadResolScrap (int idx, int idxscrap, ref double length, ref double height, ref double qx, ref double qy)

参数

- *idx*：　　求解板材的索引（从零开始）
- *idxscrap*：　求解板材上的区域索引（从零开始）
- *length*：　区域长度
- *height*：　区域高度
- *qx*：　区域左下角的 x 坐标
- *qy*：　区域左下角的 y 坐标

返回值

如果结果为正，则为 true，否则为 *false*。

注释

如果返回 *=false*，可以查询 *LastError* 属性查找错误：

- (*CutError.ErrorKey*) 密钥存在和状态检查失败
- *(CutError.ErrorNoSolutions)* 找不到求解
- *(CutError.ErrorUnexpected)* 一个或两个索引 (*idx*, *idxscrap)* 无效

3.9.19. MarginArea

该函数返回为指定索引的*当前求解*板材赋值的边距对应的面积

double MarginArea(int idx)

参数

- *idx*：　　求解板材的索引（从 0 开始）

返回值

该函数对于无效索引或求解或空赋值边距返回 0；否则，返回边距对应的面积。
边距在板材的 OneSheet 结构中赋值。

3.9.20. GetWaste

该函数返回指定索引的当前求解板材的未使用面积

double GetWaste (int idx)

参数

- *idx*：　　求解板材的索引（从 0 开始）

返回值

如果索引或求解无效，函数返回 0；否则：板材的未使用面积。

应从板材原始面积减去，计算板材的未使用面积：

- 边距对应的面积
- 切割矩形对应的面积
- 切割面积。

3.9.21. EstimatedTime

赋值客户端预计执行指定索引*当前求解*切割模式的时间

public bool EstimatedTime(int idx, long valueH, long valueM, long valueS)

参数

- *idx*： 板材索引。如果 -1：赋值求解总时间。
- *ValueH*： 小时数 (>=0)
- *ValueM*： 分钟数 (>=0)
- *ValueS*： 秒数 (>=0)

返回值

如果结果为正，则为 *true*。

注释

返回 *true* 以外的值对应以下错误情况之一：

- 未计算任何求解
- 索引无效 (idx)
- 时间值的负值。

分配时间（小时，分钟，秒）可以实现更大灵活性：例如，可以分配总时间秒数。
函数允许赋值只能根据特定应用的计算确定的信息。

3.9.22. EstimatedCost

赋值客户端预计执行指定索引*当前求解*切割模式的成本

public bool EstimatedCost(int idx, double value)

参数

- *idx*： 板材索引。如果 -1：赋值求解总成本
- *value*： 板材或求解）成本

返回值

如果结果为正，则为 *true*。

注释

返回 *true* 以外的值对应以下错误情况之一：

- 未计算求解
- 索引无效 (idx)
- 成本负值

函数允许赋值只能根据特定应用的计算确定的信息。

3.9.23. Export

函数将*当前求解*保存到一个或多个文件（XML 格式）。

CutError Export (string pathName)

参数

- *pathName*： 文件路径

返回值

如果结果为正，则为 *CutError.ErrorNone*。

注释

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorNoSolutions*) 未计算求解
- (CutError.ErrorKey) 许可证或密钥无效或找不到
- (*CutError.ErrorIOfile*) 访问待写入文件时出错
- (CutError.ErrorUnexpected) 发生意外错误。

外部应用程序可以将保存的文件解释为查询公开函数的替代。具体来说：

- 文件具有“XML”格式：如果需要，将任何扩展名“.XML”添加到 *pathName*
- 为每个计算的切割方案保存一个文件
- 每个文件包含切割模式任何重复的信息
- 第一个文件保存为 *pathName+”_001”*
- 第二个文件保存为 *pathName+”_002”*
- 直到切割模式用尽
- 覆盖现有文件。

3.9.23.1. 切割模式对应的文件结构

主代码是 **MAIN** 节点，包含嵌套元件中分布的常规信息：

```
<MAIN>
 <GENERALSETTINGS Info1="Client ABC" Info2="ABC12345" />
 <TECHSETTINGS Unit="0" Direction="0" Corner="0" MaxCutLevel="3" />
 <DIM Code="3.0" L="800.0" H="1000.0" T="0.0" G="N" Scrap="0" Priority="0" />
 <DIMTRIMS BladeThickness="15.0" PreCut="0.0" LongCut="0.0" TransvCut="0.0" ZCut="0.0" />
 <DATA Rep="2" />
 <PIECESLIST>
  <PIECE Code="1" L="640.0" H="480.0" T="0.0" Used="1" Rotation="1" G="N" Priority="0" ⋯/>
  <PIECE Code="2" L="400.0" H="500.0" T="0.0" Used="1" Rotation="1" G="N" Priority="1" ⋯ />
  ⋯
 </PIECESLIST>
 <DRAW>
  ⋯
 </DRAW>
</MAIN>
```

<GENERALSETTINGS> 节点

提供常规项目信息（参见：Custom1, Custom2, ⋯, Custom10）

- 属性：*Info1*,⋯ , *Info10*

<TECHSETTINGS> 节点

列出一些项目赋值

- *Unit* 属性：项目测量单位（参见：Unit）
- *Direction* 属性：为切割方案应用的放置的向前方向
- *Corner* 属性：放置的起始顶点（参见：Corner）
- *MaxCutLevel* 属性：可以用于优化的最大切割等级（参见 MaxCutLevels）

<DIM> 节点

提供常规板材信息（参见结构：OneSheet）

- *Code* 属性：结构的 *ID* 字段
- *L* 属性：结构的 *Length* 字段
- *H* 属性：结构的 *Height* 字段
- *T* 属性：结构的 *Thickness* 字段
- *Descr* 属性：结构的 *Label* 字段
- *Mat* 属性：结构的 *Material* 字段
- *G* 属性：结构的 *Grain* 字段（值：“N”=无纹理，“X”=x 纹理，“Y”=y 纹理）
- *Scrap* 属性：结构的 *Scrap* 字段
- *Priority* 属性：结构的 *Priority* 字段

- *TopMargin* 属性：结构的 *BorderTop* 字段
- *BottomMargin* 属性：结构的 *BorderBottom* 字段
- *RightMargin* 属性：结构的 *BorderRight* 字段
- *LeftMargin* 属性：结构的 *BorderLeft* 字段
- *Cost* 属性：结构的 *Cost* 字段

<DIMTRIMS> 节点

报告项目修整的赋值

- *BladeThickness* 属性：切割刀具的厚度（参见：CutterDiameter）
- *PreCut* 属性：修边添加到头且（参见：PreCut）
- *LongCut* 属性：添加到纵切的修整（参见：LongCut）
- *TransvCut* 属性：添加到横切的修整（参见：TransvCut）
- *ZCut* 属性：添加到 Z 或 W 切的修整（参见：Zcut）

<DATA> 节点

提供切割模式的常规信息

- *Rep* 属性：板材重复
- *Cost* 属性：预计执行成本（客户端赋值）
- *Time* 属性：预计执行时间（单位：秒）-（客户端赋值）

<PIECESLIST> 节点

列出当前切割模式切割的矩形。

每个具有一个 **<PIECE>** 节点，以及矩形的常规信息（参见结构：OneRect）：

- *Code* 属性：结构的 *ID* 字段
- *L* 属性：结构的 *Length* 字段
- *H* 属性：结构的 *Height* 字段
- *T* 属性：结构的 *Thickness* 字段
- *Descr* 属性：结构的 *Label* 字段
- *Mat* 属性：结构的 *Material* 字段
- *G* 属性：结构的 *Grain* 字段（值：“N”=无纹理，“X”=x 纹理，“Y”=y 纹理）
- *Rotation* 属性：结构的 *Rotate* 字段
- *Priority* 属性：结构的 *Priority* 字段
- *MatEdge1, ThickEdge1, EnableEdge1* 属性：结构的 *MatEdge1, ThickEdge1, BoolEdge1* 字段
- *MatEdge2, ThickEdge2, EnableEdge2* 属性：结构的 *MatEdge2, ThickEdge2, BoolEdge2* 字段
- *MatEdge3, ThickEdge3, EnableEdge3* 属性：结构的 *MatEdge3, ThickEdge3, BoolEdge3* 字段
- *MatEdge4, ThickEdge4, EnableEdge4* 属性：结构的 *MatEdge4, ThickEdge4, BoolEdge4* 字段
- *Param1,⋯,Param15* 属性：结构的字段 *(Param1,⋯,Param15)*

此外：

- *Used* 属性：当前板材中的放置数量

<DRAW> 节点

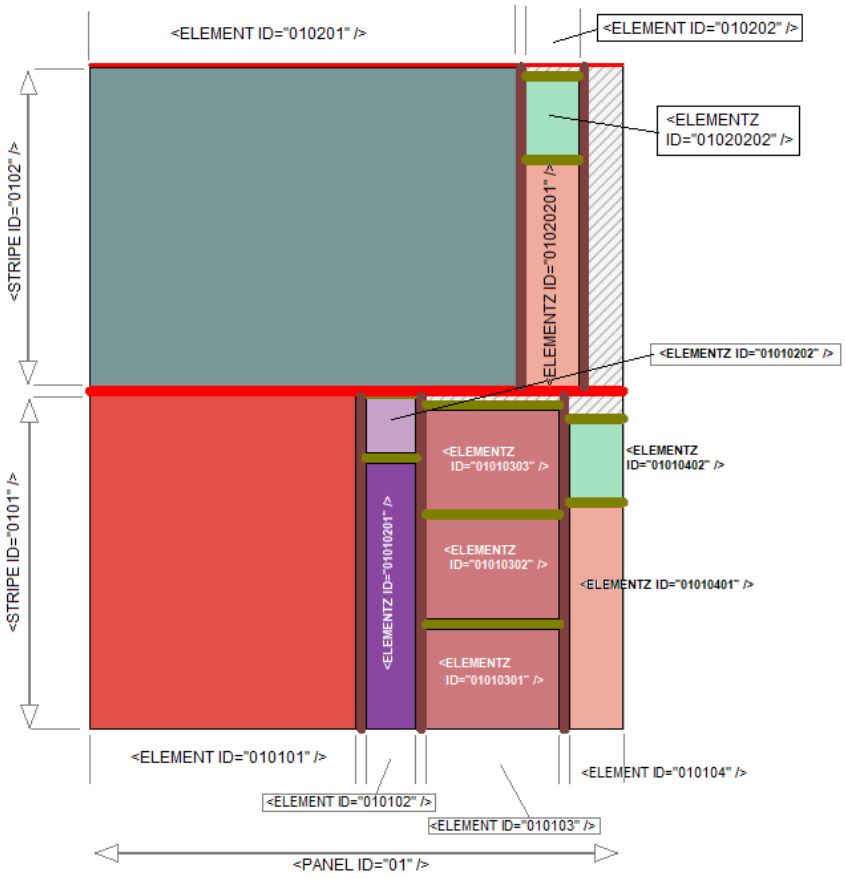
返回板材对应的切割模式，但方式和之前看到的不同。

下面是段落 ReadResolCut 的示例对应的切割模式

```
<DRAW>
 <PANEL ID="01" REP="1" L="800.00" H="1000.000">
  <STRIPE ID="0101" REP="1" L="800.000" H="500.000">
   <ELEMENT ID="010101" REP="1" L="400.000" H="500.000">
    <LABEL Code="2" Rep="1" Rotated="0" />
   </ELEMENT>
   < ELEMENT ID="010102" REP="1" L="75.000" H="500.000">
    <ELEMENTZ ID="01010201" REP="1" L="75.000" H="400.000">
     <LABEL Code="18" Rep="1" Rotated="1" />
    </ELEMENTZ>
    <ELEMENTZ ID="01010202" REP="1" L="75.000" H="82.000">
```

```
      <LABEL Code="8" Rep="1" Rotated="1" />
     </ELEMENTZ>
    </ELEMENT>
    <ELEMENT ID="010103" REP="1" L="200.000" H="500.000">
     <ELEMENTZ ID="01010301" REP="1" L="200.000" H="150.000">
      <LABEL Code="36" Rep="1" Rotated="0" />
     </ELEMENTZ>
     <ELEMENTZ ID="01010302" REP="1" L="200.000" H="150.000">
      <LABEL Code="36" Rep="1" Rotated="0" />
     </ELEMENTZ>
     <ELEMENTZ ID="01010303" REP="1" L="200.000" H="150.000">
      <LABEL Code="36" Rep="1" Rotated="0" />
     </ELEMENTZ>
    </ELEMENT>
    <ELEMENT ID="010104" REP="1" L="80.000" H="500.000">
     <ELEMENTZ ID="01010401" REP="1" L="80.000" H="333.000">
      <LABEL Code="20" Rep="1" Rotated="0" />
     </ELEMENTZ>
     <ELEMENTZ ID="01010402" REP="1" L="80.000" H="111.000">
      <LABEL Code="21" Rep="1" Rotated="0" />
     </ELEMENTZ>
    </ELEMENT>
   </STRIPE>
   <STRIPE ID="0102" REP="1" L="800.000" H="480.000">
    <ELEMENT ID="010201" REP="1" L="640.000" H="480.000">
     <LABEL Code="1" Rep="1" Rotated="0" />
    </ELEMENT>
    <ELEMENT ID="010202" REP="1" L="80.000" H="480.000">
     <ELEMENTZ ID="01020201" REP="1" L="80.000" H="333.000">
      <LABEL Code="20" Rep="1" Rotated="0" />
     </ELEMENTZ>
     <ELEMENTZ ID="01020202" REP="1" L="80.000" H="111.000">
      <LABEL Code="21" Rep="1" Rotated="0" />
     </ELEMENTZ>
    </ELEMENT>
   </STRIPE>
  </PANEL>
</DRAW>
```

下图突出显示“xml”架构中每个节点的含义：



<PANEL> 节点

给定原始面板，可以通过 head cut（垂直方向切割，覆盖原始面板高度）为多个面板：已经提到，这是应用*垂直方向*的优化面板情况。

这样获得的每个元素对应一部分 **<PANEL>** 类型：对于应用*水平方向*优化面板的情况，该部分始终一个（如示例中）。**<PANEL>** 类型部分的顺序遵循在原始面板上的放置顺序，从用于放置的顶点开始，向相反顶点（在示例中：从*左下*顶点到*右上*顶点）。

节点属性对所有后续节点类型部分通用，名称和含义。

<PANEL ID="01" REP="1" L="800.00" H="1000.000">

- *ID* 属性：单义标识符
- *REP* 属性：元素重复（默认 = 1）
- *L* 属性：元素水平尺寸（切割净）
- *H* 属性：元素垂直尺寸（切割净）

节点的具体属性：

- *PreCut* 属性：报告头切修边值（参见：PreCut），如果元素需要执行相应修边，则赋值
- *MarginCut* 属性：报告靠近优化角（左/右）的边距值，如果元素需要执行相应切割并且不赋值 *PreCut* 属性，则赋值
- *MarginCutAlt* 属性：报告优化角（左/右）相对边距的值，如果元素需要执行相应切割，并且其在板材上的位置不需要执行头切，则赋值
- *TFC* 属性：报告 TFC 类型切割值（参见：TensionGap），如果元素需要执行相应切割，则赋值。请记住，切割尺寸添加到刀具尺寸进行的通常切割

在我们的示例中：元素尺寸和原始面板相同 (800 x 1000 mm)。

<STRIPE> 节点

<PANEL> 节点中插入的节点，描述纵切（水平方向切割，覆盖属于 **<PANEL>** 的元素长度）生成的*条*。**<STRIPE>** 类型部分的顺序遵循 **<PANEL>** 元素上排列的顺序，符合用于放置的顶点展示的方向（在示例中：从下到上）。

节点的常规属性和为可添加具体节点属性的父节点检查的属性相同 (ID, REP, L, H)：

- *LongCut* 属性：报告边切修边值 (vedi: LongCut），如果元素要求执行相应修边，则赋值
- *MarginCut* 属性：报告靠近优化角（下/上）的边距值，如果元素需要执行相应切割并且不赋值 *LongCut* 属性，则赋值
- *MarginCutAlt* 属性：报告优化角（上/下）相对边距的值，如果元素需要执行相应切割，并且其在板材上的位置不需要执行纵切，则赋值
- *TFC* 属性：报告 TFC 类型切割值（参见：TensionGap)，如果元素需要执行相应切割，则赋值。请记住，切割尺寸添加到刀具尺寸进行的正常切割。

在此示例中，赋值 2 个 **<STRIPE>** 部分。

```
<DRAW>
 <PANEL ID="01" REP="1" L="800.00" H="1000.000">
  <STRIPE ID="0101" REP="1" L="800.000" H="500.000">
   …

  </STRIPE>
  <STRIPE ID="0102" REP="1" L="800.000" H="480.000">
   …
  </STRIPE>
 </PANEL>
</DRAW>
```

<ELEMENT> 节点

<STRIPE> 节点中插入的节点，描述横切（垂直方向切割，覆盖属于 **<STRIPE>** 的元素高度）生成的元素。**<ELEMENT>** 类型部分的顺序遵循 **<STRIPE>** 元素上放置的顺序，符合用于放置的顶点展示的方向（在示例中：从左到右）。

节点的常规属性和为可添加具体节点属性的父节点检查的属性相同 (ID, REP, L, H)：

- *TransvCut* 属性：报告交叉切割修边值（参见：TransvCut）如果元素要求执行相应修边，则赋值
- *MarginCut* 属性：报告靠近优化角（左/右）的边距值，如果元素需要执行相应切割并且不赋值 *TransvCut* 属性，则赋值
- *MarginCutAlt* 属性：报告优化角（右/左）相对边距的值，如果元素需要执行相应切割，并且其在板材上的位置不需要执行纵切，则赋值。
- *Type* 属性：如果元素对应废料区域，则存在。属性值赋值其类型：1=残料，2=废料。

节点属性和为父节点检查的相同。
在此示例中，赋值如下：

- 第一个条的 4 个部分
- 第二个条的 2 个部分。

```
<PANEL ID="01" REP="1" L="800.00" H="1000.000">
   <STRIPE ID="0101" REP="1" L="800.000" H="500.000">
    <ELEMENT ID="010101" REP="1" L="400.000" H="500.000">
     <LABEL Code="2" Rep="1" Rotated="0" />
    </ELEMENT>
    <ELEMENT ID="010102" REP="1" L="75.000" H="500.000">
     …
   </ELEMENT>
   <ELEMENT ID="010103" REP="1" L="200.000" H="500.000">
     …
    </ELEMENT>
    <ELEMENT ID="010104" REP="1" L="80.000" H="500.000">
     …
    </ELEMENT>
   </STRIPE>
   <STRIPE ID="0102" REP="1" L="800.000" H="480.000">
    <ELEMENT ID="010201" REP="1" L="640.000" H="480.000">
     …
    </ELEMENT>
    <ELEMENT ID="010202" REP="1" L="80.000" H="480.000">
     …
    </ELEMENT>
   </STRIPE>
  </PANEL>
```

<ELEMENTZ> 节点

<ELEMENT> 节点中插入的节点，描述 Z 切（水平方向切割，覆盖属于 **<ELEMENT>** 的元素长度）生成的元素。
<ELEMENTZ> 类型部分的顺序遵循 **<ELEMENT>** 元素上放置的顺序，符合用于放置的顶点展示的方向（在示例中：从下到上）。
节点的常规属性和为可添加具体节点属性的父节点检查的属性相同 (ID, REP, L, H)：

- *Zcut* 属性：报告 Z 修边值（参见：Zcut）如果元素要求执行相应修边，则赋值。
- *Type* 属性：如果元素对应废料区域，则存在。属性值赋值其类型：1=残料，2=废料。

<ELEMENTW> 节点

<ELEMENTZ> 节点中插入的节点，描述 W 切（垂直方向切割，覆盖属于 **<ELEMENTZ>** 的元素高度）生成的元素。
<ELEMENTW> 类型部分的顺序遵循 **<ELEMENTZ>** 元素上放置的顺序，符合用于放置的顶点展示的方向（在示例中：从左到右）。
在此示例中，不赋值节点。
节点属性和为节点 <ELEMENTZ> 检查的属性相同。

<ELEMENT5> 节点

<ELEMENTW> 节点中插入的节点，描述 cuts #5（水平方向切割，覆盖属于 **<ELEMENTW>** 的元素长度）生成的元素。
<ELEMENT5> 类型部分的顺序遵循 **<ELEMENTW>** 元素上放置的顺序，符合用于放置的顶点展示的方向（在示例中：从下到上）。
在此示例中，不赋值节点。
节点属性是为属性 (TYPE) 添加到的父节点 (ID, REP, L, H) 检查的属性。

<ELEMENT6> 节点

<ELEMENT5> 节点中插入的节点，描述 cuts #6（垂直方向切割，覆盖属于 **<ELEMENT5>** 的元素高度）生成的元素。
<ELEMENT6> 类型部分的顺序遵循 **<ELEMENT6>** 元素上放置的顺序，符合用于放置的顶点展示的方向（在示例中：从左到右）。
在此示例中，不赋值节点。
节点属性是为属性 (TYPE) 添加到的父节点 (ID, REP, L, H) 检查的属性。

<LABEL> 节点

此类型节点可以插入类型 (PANEL, STRIPE, ELEMENT, ELEMENTZ, ELEMENTW, ELEMENT5, ELEMENT6) 的每个节点，按照相应元素中放置的矩形定义。
<LABEL> 节点终止切割分支。
*如果类型 (ELEMENT, ELEMENTZ, ELEMENTW, ELEMENT5, ELEMENT6) 的节点赋值 TYPE 类型值 1 或 2，直接终止切割分支，则不赋值 **<LABEL>** 节点。*

```
<PANEL ID="01" REP="1" L="800.00" H="1000.000">
    < STRIPE ID="0101" REP="1" L="800.000" H="500.000">
      <ELEMENT ID="010101" REP="1" L="400.000" H="500.000">
        <LABEL Code="2" Rep="1" Rotated="0" />
      </ELEMENT>
      …
    </STRIPE>
    < STRIPE ID="0102" REP="1" L="800.000" H="480.000">
      …
    </STRIPE>
  </PANEL>
```

节点赋值以下属性：

- *Code* 属性：矩形标识符（OneRect 结构中的 ID 字段）
- *REP* 属性：元素重复（默认 = 1）
- *Rotated* 属性：指示是否旋转放置矩形。

3.10. 项目序列化函数

函数处理与文件之间的序列化。

3.10.1. SaveProject

函数将*矩形*和*板材*列表赋值保存到文件（XML 格式）。

CutError SaveProject (string pathName, bool bMode)

参数

- *pathName*： 文件路径
- *bMode*： *true* 需要保存常规项目设置

返回值

如果结果为正，则为 *CutError.ErrorNone*。

注释

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorIOFile*) 访问待写入文件时出错。

还利用 *bMode=true* 保存常规项目设置。

3.10.2. LoadProject

函数从文件（XML 格式）读取*矩形*和*板材*列表赋值。

CutError LoadProject (string pathName, bool bMode)

参数

- *pathName*： 文件路径
- *bMode*： *true* 支持读取常规项目设置

返回值

如果结果为正，则为 *CutError.ErrorNone*。

注释

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorIOFile*) 访问文件时出错。

还使用 *bMode=true* 读取常规项目设置：文件未读取的设置初始化为默认值。
如果 *bMode=false*：设置保持不变。

执行该函数不更改常规优化设置。

3.10.3. ImportProject

该函数从 CSV 文件导入*板材和矩形*列表赋值

CutError ImportProject (string pathName, string stFormatSheet, string stFormatRect)

CutError ImportProject (string pathName, string stFormatSheet, string stFormatRect, int countHeader, string stSeparator)

参数

- *pathName*： 文件路径
- *stFormatSheet*： 板材赋值行的格式化字符串（CSV 文件中）
- *stFormatRect*： 矩形赋值行的格式化字符串（CSV 文件中）
- *countHeader*： 标题行数 (>= 0)-（可包含元素格式设置行）
- *stSeparator*： 赋值可以用作 CSV 文件中字段分隔符的字符。

返回值

如果结果为正，则为 *CutError.ErrorNone*。

注释

返回 *CutError.ErrorNone* 以外的值对应以下错误情况之一：

- (*CutError.ErrorIOFile*) 访问文件时出错

- (*CutError.ErrorFileNotValid*) 数据读取不一致

函数赋值 CSV 文件中的*板材和矩形* 列表，保留其余常规项目设置不变，初始化当前项目，澄清一点：

- *矩形* 列表在函数调用时重置
- 仅当需要相应类型的第一个元素时，*板材* 列表重置。
  这样可以赋值 CSV 文件仅获得矩形列表，板材原始分配不变。

CSV 文件是用于赋值数据表的文本文件。文件的每一行匹配一个表格行，反过来通过分隔符细分为字段（单个列）。*stSeparator* 列出识别为有效的字符：如果赋值字符串为空，则使用 =”;,”。下面假定使用的分隔符字符为：分号。

函数赋值两个字符串，格式化每个项类型的赋值行，如果同一 CSV 文件不再赋值时使用。对于格式设置字符串，使用分号作为分隔符。

countHeader 参数指定解释为标题行，不参与板材或矩形对应元素解释的最小行数。*countHeader* 值可以包含元素的格式赋值行。

CSV 文件的赋值规则：

- 对应 *countHeader 的数字中的标题行数*
  - 不强制行的格式
  - 可以包含元素的格式赋值行数
  - 不包含板材或矩形类型元素解释

- 元素的格式赋值行
  - **“***panel;…”，对于*板材*
  - **“***part;…”，对于*矩形*
  - 是可选行，可以在文件前几行之外赋值 (*countHeader)*
  - 如果存在（一个或两个），将替换函数参数 (*stFormatSheet, stFormatRect*)，必须在板材或矩形类型元素每个赋值行之前赋值

- 板材或矩形类型元素的赋值行
  - **“panel;…”** 解释 *sheet* 类型元素。已经提到：此类型的第一次采集清除板材列表
  - **“part;…”** 解释 *rectangle* 类型元素。行不可包含标题部分 **(part;)**

3.10.3.1. 格式化字符串

如果赋值对应 length、quantity 的字段，则格式化字符串视为有效。每个字段标记一个最多 2 个字符的单义字符串。
我们看一看完整列表：

- ***e*** 启用行（解释：0/1、yes/no、on/off、true/false）–（默认=1）
- ***d*** 元素说明 – (默认=””)
- ***l*** 元素长度尺寸
- ***h*** 元素高度尺寸
- ***s*** 元素厚度尺寸
- ***n*** 元素数量
- ***x*** 额外数量（仅针对矩形）
- ***m*** 材料
- ***g*** 纹理（解释：1/2, x/y) –（默认=0）
- ***r*** 旋转可能（仅针对矩形）（解释：0/1, yes/no, on/off, true/false) –（默认=0）
- ***p*** 优先级
- ***et*** 矩形上边（最大格式：“thickness|name|cut flag”。例如“3.0|abc456|0”）
- ***eb*** 矩形下边（格式：参见上一个字段）
- ***er*** 矩形右边（格式：参见上一个字段）
- ***el*** 矩形左边（格式：参见上一个字段）
- ***a1*** 矩形的 Param1 字段
- ***a2;a3;a4;a5;a6;a7;a8:a9;a10;a11;a12;a13;a14;a15***
  矩形的字段 *(Param2,…,Param15)*
- ***mt*** 板材上边距
- ***mb*** 板材下边距

- ***mr*** 板材右边距
- ***ml*** 板材左边距
- ***y*** 废料板材信息（解释：0/1, yes/no, on/off, true/false）。

3.10.3.2. CSV 文件示例 (1)

***panel;d;l;h;n;m;g;p ***part;d;l;h;n;x;m;g;p;r	元素格式化行 • 可选 • 不一定同时存在 • 突出显示识别的行标题： - “***panel;”，用于板材 - “***part;”，用于矩形
panel;MDF;2000;1000;1; panel;MDF2;2000;1500;1;	板材赋值行 - 突出显示识别的行标题：“panel;” - 后续字段按照文件第一行赋值的格式解释
part;A1;1000;300;30;0;;;1 part;A2;750;250;30;0;;;1 part;A3;300;250;30;0;;;1 part;A4;150;130;50;10;;;1 part;A5;180;150;0;20;;;0	矩形赋值行 - 突出显示识别的行标题：“part;” - 后续字段按照文件第二行赋值的格式解释

- 板材的格式化字符串赋值“d;l;h;n;m;g;p”：
  - “d”：赋值 OneSheet 结构的 *Label* 字段
  - “l”：赋值结构的 *Length* 字段
  - “h”：赋值结构的 *Height* 字段
  - “n”：赋值结构的 *N* 字段
  - “m”：赋值结构的 *Material* 字段
  - “g”：赋值结构的 *Grain* 字段
  - “p”：赋值结构的 *Priority* 字段

- 矩形的格式化字符串赋值“d;l;h;n;x;m;g;p;r”：
  - “d”：赋值 OneRect 结构的 *Label* 字段
  - “l”：赋值结构的 *Length* 字段
  - “h”：赋值结构的 *Height* 字段
  - “n”：赋值结构的 *N* 字段
  - “x”：赋值结构的 *Extra* 字段
  - “m”：赋值结构的 *Material* 字段
  - “g”：赋值结构的 *Grain* 字段
  - “p”：赋值结构的 *Priority* 字段
  - “r”：赋值结构的 *Rotate* 字段。

3.10.3.3. CSV 文件示例 (2)

name;length;height;quantity;extra;rotation	标题行 - 赋值 *countHeader=1* 执行其解释
A1;1000;300;30;0;1 A2;750;250;30;0;1 A3;300;250;30;0;1 A4;150;130;50;0;1 A5;180;150;0;20;0	矩形赋值行 - 根据 *stFormatRect* 参数中赋值的格式解释字段

示例文件不赋值 *sheet* 类型元素：函数保持板材列表不变。

4. 库使用指南

本章介绍将 TPA_C 集成到应用程序所需的信息。

TpaCutOEM 作为 32 位或 64 位类库提供。

4.1. 安装程序

安装程序可通过正常模式或静默模式启动。

4.1.1. 正常安装

正常安装将在所选文件夹下创建一个完整的演示环境。默认安装路径为：“C:\TpaCutOEM”。

安装程序会创建以下子文件夹：

- “\bin”：用于存放演示可执行程序（TpaCutDEMO.exe）及所需库文件
- “\help”：用于存放 TPA_N 库的教程
- “\CutOemCfg”：辅助配置文件夹
- “\Examples”：用于排样的示例项目
- “\HyLicenses”：软件许可证管理相关的程序文件夹

建议使用正常安装模式，尤其适用于对库进行初步评估的场景。演示应用程序提供了 TPA_C 集成的示例：该环境以简洁直观的方式开发，几乎包含所有可用选项，可直观评估各项功能的实际应用效果。

在 “\bin” 文件夹中，您将找到演示程序的配置文件（TpaCutDEMO.exe.config），其中包含许可证识别所需的关键信息，可作为创建您自己应用程序对应配置文件的参考。

在 “\HyLicenses” 文件夹中包含可执行文件 HyLicenses.Exe：该程序用于许可证的安装、更新与迁移操作。关于该程序的使用方法，请参阅专用文档。

4.1.2. 静默安装

静默安装（又称隐藏安装）仅安装将库集成至您应用程序所需的必要文件。*静默安装*过程中不会弹出任何提示，例如目标文件夹选择或许可协议确认，所有过程均在后台自动执行，对用户完全不可见。

静默安装通常用于自定义安装程序中的自动调用场景。

在命令行模式下启动安装时，系统会自动识别所输入的命令。

命令	*含义*
/PATH /INSTALLDIR	指定产品的完整安装路径（示例：/INSTALLDIR=c:\abc） 所有集成库所需文件将直接复制到该路径中
/HYLIC	要求安装 “\HyLicenses” 文件夹（该文件夹用于软件许可证的管理） 该文件夹将从 /PATH 创建
/S	要求以静默模式安装（在无其他命令时调用该命令）

4.2. 典型流程图

下面介绍库的基本使用：

1. 创建 *TpaCutOEM.TpaRctCut()* 类的实例
2. 执行常规操作的初步检查（密钥存在检查）
3. 赋值常规操作设置（参见：*NumberCustom*, *NumberRectParam*, …）
4. 赋值常规项目设置（参见：*Unit*, *Direction*, …）
5. 用 *AddRct()* 函数调用赋值*矩形*
6. 用 *AddSheet()* 函数调用赋值*板材*
7. 用 *Compute()* 函数调用启动优化过程
8. 获得调用函数的结果：NumberOfSolutions, SelectSolution, …, …
9. 自主采集/处理类优化和赋值的效率信息 (EstimatedTime, EstimatedCost)
10. 保存优化报告 (Export)。

赋值*设置*、*矩形* 和*板材* 的顺序相同：提出的顺序仅供参考。
常规设置、矩形和板材集构成*切割模式*，标识执行优化需要的数据集。

4.3.初步检查

初步检查密钥存在可能有用，是调整应用程序功能必要的：

- 查询 *IsValidLicense* 检查密钥存在和有效性

4.4.赋值常规操作设置

通常需要对常规操作设置执行初步赋值。此信息不会随项目更改而更改：通常与应用程序常规操作选项有关。

4.5.赋值常规项目设置

必须对常规项目设置进行初步赋值，每个项目不同。
还可以从以前读取的文件初步赋值设置：参见 *SaveProject()*, *LoadProject()* 函数。

4.6.赋值矩形

继续赋值矩形列表。
赋值矩形时需要注意一些要点：

- 每个矩形具有单义严格为正的数字标识符 (> 0)：*OneRect* 结构的 *ID* 字段
- 因此无法为多个矩形赋值相同 *ID*
- 可以按照任何顺序赋值标识符，不一定在一行
- 注意，ID 在排序矩形时起到重要作用：和其他重要设置（例如优先级、矩形面积）一样，ID 决定哪个矩形在优化过程中具有最高放置优先级。

要添加矩形，调用 *AddRct()* 函数。
要删除矩形，调用 *ClearRct()* 函数。

4.7.赋值板材

继续赋值板材列表。
和矩形一样，赋值板材时需要注意一些要点：

- 每个板材具有单义严格为正的数字标识符 (> 0)：*OneSheet* 结构的 *ID* 字段
- 因此，无法向更多群集赋值同一 *ID*

- 可以按照任何顺序赋值标识符，不一定在一行
- 注意，ID 在排序板材时起到重要作用：和其他重要设置（例如优先级、废料信息、尺寸）一样，ID 决定哪个板材在优化过程中具有最高放置优先级。

要添加板材，调用 *AddSheet()* 函数。
要删除板材列表，调用 *ClearSheet()* 函数。

4.8.执行切割优化

要开始项目优化，需要调用 Compute 函数。
矩形和板材检查在优化前执行，还可导致函数返回错误并取消优化：

- 为放置和板材请求的矩形必须可用
- 检查匹配筛选器（厚度、材料）必须能够匹配至少一个矩形和一个板材。

4.8.1. 获得求解结果

优化后，可以获得切割求解的结果。

一组函数采集优化求解的所有相关信息：

- *NumberOfSolutions*：优化器计算的求解数量
- *FitnessSheet*：求解效率
- *SolutionSheets*：求解切割模式数量信息
- *UsedRects*：求解放置数量或板材和/或矩形类型信息
- *ReadResolRect*：求解单个板材的矩形信息
- *ReadResolCut*：求解单个板材的切割信息
- *Export*：将求解保存到文件（XML 格式）

4.8.1.1. 示例代码：如何获取求解板材的常规信息

```
TpaCutOEM.TpaRctCut optimizeObj=new TpaCutOEM.TpaRctCut ();

...
//组件查询赋值字段:
OneRect OneRect = new OneRect();

//求解板材查询循环
int IndexSheet = 0; //板材索引
for (IndexSheet = 0; IndexSheet < optimizeObj.SolutionSheets; IndexSheet++)
{
    //index sheet ID (IndexSheet)
    int IdSheet = optimizeObj.GetSheetID(IndexSheet);
    //板材重复
    int sheetRepetition = optimizeObj.NumberOfRepetitions(IndexSheet);
    // ···
    //板材放置采集循环
    //·····
}
```

4.8.1.2. 示例代码：板材放置的获取循环

```
// IndexSheet = 板材索引（参见：上一个循环）
int IndexRct = 0; //板材放置索引
OneRect oneRect=new OneRect();

for (IndexRct = 0; IndexRct < UsedRects(IndexSheet, 0); IndexRct ++)
{
        double qx = 0.0, qy = 0.0;
        bool rotate = false;
```

```
            //读取插入工件的位置和旋转
            int ID = optimizeObj.ReadResolRect(IndexSheet, IndexRct, ref qx, ref qy, ref rotate);
            //读取插入工件的尺寸和特征
            OptimizeObj.ReadRect(ID, ref OneRect);
            //…
    }
```

4.8.1.3. 示例代码：板材可重复使用区域的采集循环

```
// IndexSheet = 板材索引（参见：上一个循环）
int IndexArea= 0; //板材上的区域索引
OneRect oneRect=new OneRect();

for (IndexArea = 0; IndexArea < NumberOfRemains (IndexSheet, 0); IndexArea ++)
{
            double dl = 0.0, dh = 0.0;
            double qx = 0.0, qy = 0.0;
            //读取区域位置和尺寸
            int Id = optimizeObj.ReadResolRemain (IndexSheet, IndexArea, ref dl, ref dh, ref qx, ref qy);
            //…
}
```

4.8.1.4. 示例代码：板材切割的获取循环

```
    // IndexSheet = 板材索引（参见：上一个循环）
    for (idxP = 0; idxP < optimizeObj.NumberOfCuts(IndexSheet); idxP++)
   {
        double xStart = 0, yStart = 0, xEnd = 0, yEnd = 0, 厚度 = 0;
        int orientation = 0;
        int levelCut = optimizeObj.ReadResolCut(i, idxP, ref xStart, ref yStart, ref xEnd, ref yEnd, ref orientation, ref 厚度);
          //…
    }
```

4.8.2. 管理多个求解

优化器可以找到更多求解，最多 5 个。
执行优化时，计算求解中的第一个求解自动设置为*当前求解*。
所有计算求解都是可用的，可以浏览，也可以将当前求解更改为特定求解。
可以调用 Export 函数，将当前求解保存到文件。

4.8.2.1. 示例代码：导航多个求解

```
    TpaCutOEM.TpaRctCut optimizeObj=new TpaCutOEM.TpaRctCut();

    // GoToNextSolution: 移动到下一个计算的求解
    bool GoToNextSolution()
    {
        If (optimizeObj.SelectSolution < optimizeObj.NumberOfSolution)
        {
            optimizeObj.SelectSolution = optimizeObj.SelectSolution+1;
            return true;
        }
        return false;
    }

    // GoToPrevSolution: 移动到上一个计算的求解
    bool GoToPrevSolution()
    {
        If (optimizeObj.SelectSolution >0)
        {
            optimizeObj.SelectSolution = optimizeObj.SelectSolution-1;
```

```
            return true;
        }
        return false;
    }
```

4.9. 保存和读取 TPA_C 项目

可以使用 SaveProject 和 LoadProject 函数，保存和找回项目。
这两个函数还支持保存和找回常规项目设置。
执行 *LoadProject* 后，外部应用程序需要从 TPA_C 读取信息，更新项目设置、矩形和板材。

4.9.1. 示例代码

```
TpaCutOEM.TpaRctCut optimizeObj=new TpaCutOEM.TpaRctCut();

//结构
OneRect ItemRect=new OneRect();
OneSheet ItemSheet=new OneSheet();

// 读取项目
if (optimizeObj.LoadProject ("C:\projects\one.xml", true) == CutError.ErrorNone)
{
  //----------项目设置
  myUnit= optimizeObj.Unit;
  myDirection= optimizeObj.Direction;
  myCorner= optimizeObj.Corner;
  myFiTool= optimizeObj.CutterDiameter;
  myTensionGap= optimizeObj.TensionGap;
  myMaxCutLevels= optimizeObj.MaxCutLevels;
  myPreCut= optimizeObj.PreCut;
  myLongCut= optimizeObj.LongCut;
  myTransvCut= optimizeObj.TransvCut;
  myZCut= optimizeObj.ZCut;
  myInfoCst[ni = 0]= optimizeObj.Custom1;
  myInfoCst[++ni]= optimizeObj.Custom2;
  ….
  myInfoCst[++ni]= optimizeObj.Custom10;

  //----------读取矩形
  for (int idxElement=0; idxElement< optimizeObj.CountRect; idxElement++)
    {
        optimizeObj.ReadRectIndex (idxElement, ref ItemRect);
        // …
        // 读取和处理数据
       //…
    }

    //---------- 读取板材
    for (int idxElement=0; idxElement< optimizeObj.CountSheet; idxElement++)
    {
        optimizeObj.ReadSheetIndex (idxElement, ref ItemSheet);
        // …
        // 读取和处理数据
       //…
    }
}
```