­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­

Лабораторна робота

з курсу «Теоретичні основи САПР»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 10

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2014

## Мета роботи

Розробити программу та її графічний інтерфейс для автоматизованої побудови деталі в SolidWorks за допомогою його API.

## Основні теоретичні відомості

Останнім часом відзначається постійне зростання інтересу до САПР SolidWorks не тільки з точки зору 3D-моделювання, але і в плані розробки прикладного програмного забезпечення, що автоматизує різні етапи проектно-конструкторських робіт.

SolidWorks API - це інтерфейс, що дозволяє розробляти користувальницькі додатки для системи SolidWorks. API-інтерфейс містить сотні функцій, які можна викликати з програм Microsoft Visual Basic, VBA (Microsoft Excel, Word, Access і т.д.), Microsoft Visual C + +, C# або з файлів-макросів SolidWorks. Ці функції надають програмісту прямий доступ до функціональних можливостей SolidWorks.

Пакет SolidWorks надає користувачеві широкий спектр функціональних можливостей, що дозволяють створювати і модифікувати як 2D-, так і 3D-геометрію. Завдяки цьому в даний момент він є однією з найпопулярніших систем проектування як в Росії, так і в усьому світі. Однак при всьому багатстві функцій SolidWorks робота через стандартний користувальницький інтерфейс забезпечує інженеру доступ приблизно до 80% функціональних можливостей, закладених розробниками в SolidWorks. Це ні в якому разі не є утиском прав користувачів. Просто деякі дії зручніше і набагато швидше виконуються саме за допомогою програмного інтерфейсу.

Наведемо ще один аргумент на користь SolidWorks API. Пакет SolidWorks як універсальна система призначений не тільки для загального машинобудування, а й для цілого ряду інших галузей. А як відомо, будь-яка універсальність має свої плюси і мінуси: з одного боку, програму легко освоїти, з іншого - в ній немає, наприклад, кнопки, що дозволяє легким рухом миші за одну дію побудувати модель стільникового телефону або кар'єрного екскаватора. Тільки завдяки SolidWorks API ви можете самі створити цю кнопку і автоматизувати таким чином процес побудови моделі.

## Завдання

Написати программу для автоматизованої побудови мікросхеми DIP, використовуючи API САПР SolidWorks. Розробити базу данних параметрів мікросхем.

## Реалізація

Лістінг

﻿namespace SolidDip.Solid

{

using System;

using System.Linq;

using Helpers;

using SolidDip.Model;

using SolidWorks.Interop.sldworks;

using SolidWorks.Interop.swconst;

public static class SwController

{

public static void BuildDip(DipCorpus corpus)

{

var sw = new SldWorks();

sw.Visible = true;

var pinCount = corpus.PinCount;

var pinDistance = 0.00254;

var pinRadius = 0.0004;

var firstPin = 0.00109;

var width = corpus.CorpusWidthMm / 1000; // 0.00648;

var length = (((pinCount / 2) - 1) \* pinDistance) + (firstPin \* 2.0);

var height = 0.00368;

var bevel = 0.0005;

var bottomBevel = 0.0004;

var pinSurfHeight = 0.0006;

var halfHeight = height / 2.0;

var halfWidth = width / 2.0;

var halfLength = length / 2.0;

var widthWithPins = width + (pinRadius \* 2); // 0.00762;

var pinThickness = 0.0003;

var pinWidth = 0.0005;

var pinWidthBase = 0.0015;

var bottomPinHeight = 0.0032;

var bevelPinHeight = 0.00051;

var pinHeight = bottomPinHeight + bevelPinHeight + halfHeight;

ModelDoc2 doc = sw.NewPart();

// Body

doc.Extension.SelectByID2("Top Plane", "PLANE", 0, 0, 0, false, 0, null, 0);

var bodySketchSegments = ((dynamic[])doc.SketchManager.CreateCenterRectangle(0, 0, 0, halfWidth, halfLength, 0)).Cast<SketchSegment>();

doc.SketchManager.InsertSketch(true);

doc.SelectSegments(bodySketchSegments);

var bodyFeature = doc.FeatureManager.FeatureExtrusion2(false, false, false,

0, 0, height / 2.0, height / 2.0, false, false, false, false,

0, 0, false, false, false, false, true, true, true, 0, 0, false);

bodyFeature.Name = "Body Feature";

// Make it black

foreach (var face in (dynamic[])bodyFeature.GetFaces())

{

var body = face.GetBody();

body.MaterialPropertyValues2 = new double[9] { 0, 0, 0, 1, 1, 1, 0.3, 0, 0 }; // Black

}

doc.ClearSelection2(true);

// Bevel

doc.Extension.SelectByID2("Front Plane", "PLANE", 0, 0, 0, false, 0, null, 0);

doc.SketchManager.InsertSketch(true);

var leftBevelSketchSegments = doc.SketchManager.CreateLinesByPoints(

-halfWidth, +pinSurfHeight / 2.0, .0, // A

-halfWidth + bevel, +halfHeight, .0, // B

halfWidth - bevel, +halfHeight, .0, // C

halfWidth, +pinSurfHeight / 2.0, .0, // D

halfWidth, -pinSurfHeight / 2.0, .0, // E

halfWidth - bottomBevel, -halfHeight, .0, // F

-halfWidth + bottomBevel, -halfHeight, .0, // G

-halfWidth, -pinSurfHeight / 2.0, .0, // H

-halfWidth, +pinSurfHeight / 2.0, .0 // A

);

doc.SketchManager.InsertSketch(true);

doc.SelectSegments(leftBevelSketchSegments);

var leftBevelFeature = doc.FeatureManager.FeatureCutThin2(false, false, false, 0, 0, length, length, false, false, false, false,

1.74532925199433E-02, 1.74532925199433E-02, false, false, false, false, 0.01, 0.01, 0.01, 0, 0, false, 0.005, true, true, 0, 0, false);

leftBevelFeature.Name = "Bevel Feature";

doc.ClearSelection2(true);

// Cover Plane

doc.Extension.SelectByID2("Top Plane", "PLANE", 0, 0, 0, true, 0, null, 0);

var coverPlane = doc.FeatureManager.InsertRefPlane(8, halfHeight, 0, 0, 0, 0);

coverPlane.Name = "Cover Plane";

// Sign

doc.Extension.SelectByID2("Cover Plane", "PLANE", 0, 0, 0, true, 0, null, 0);

var signSketchSegments = ((dynamic[])doc.SketchManager.CreateCenterRectangle(0, halfLength, .0, .0005, halfLength + .001, .0)).Cast<SketchSegment>();

doc.RoundCorners(0.0002, signSketchSegments);

doc.SketchManager.InsertSketch(true);

doc.SelectSegments(signSketchSegments);

var signFeature = doc.FeatureManager.FeatureCut3(true, false, false, 0, 0, halfHeight / 4.0, .0, false, false, false, false,

1.74532925199433E-02, 1.74532925199433E-02, false, false, false, false, false, true, true, true, true, false, 0, .0, false);

signFeature.Name = "Sign Feature";

doc.ClearSelection2(true);

// Pin Plane

doc.Extension.SelectByID2("Front Plane", "PLANE", 0, 0, 0, true, 0, null, 0);

var pinPlane = doc.FeatureManager.InsertRefPlane(264, (halfLength - firstPin), 0, 0, 0, 0);

pinPlane.Name = "Pin Plane";

// First and Last Pin

doc.Extension.SelectByID2("Pin Plane", "PLANE", 0, 0, 0, false, 0, null, 0);

doc.SketchManager.InsertSketch(true);

var pinSketchSegments = doc.SketchManager.CreateLinesByPoints(

-widthWithPins / 2.0, -pinHeight, .0, // A

-(halfWidth + pinRadius), +pinSurfHeight / 2.0, .0, // B

+(halfWidth + pinRadius), +pinSurfHeight / 2.0, .0, // C

+widthWithPins / 2.0, -pinHeight, .0 // D

);

doc.SketchManager.InsertSketch(true);

var customBendAllowanceData = doc.FeatureManager.CreateCustomBendAllowance();

customBendAllowanceData.KFactor = 0.5;

var pinFeature = doc.FeatureManager.InsertSheetMetalBaseFlange2(pinThickness, false, pinRadius, pinWidthBase / 2.0, pinWidthBase / 2.0, false, 0, 0, 0,

customBendAllowanceData, false, 0, 0.0001, 0.0001, 0.5, true, false, true, true);

pinFeature.Name = "First and Last Pin Feature";

// Pin profile

doc.Extension.SelectByID2("Right Plane", "PLANE", 0, 0, 0, true, 0, null, 0);

doc.SketchManager.InsertSketch(true);

var dPin = halfLength - firstPin;

var pinProfileSketchSegments = doc.SketchManager.CreateLinesByPoints(

-(pinWidth / 2.0) + dPin, -pinHeight, .0, // A

-(pinWidth / 2.0) + dPin, -(halfHeight + bevelPinHeight), .0, // B

-(pinWidthBase / 2.0) + dPin, -halfHeight, .0, // C

-(pinWidthBase / 2.0) + dPin, -(pinHeight + 0.001), .0,

(pinWidthBase / 2.0) + dPin, -(pinHeight + 0.001), .0,

(pinWidthBase / 2.0) + dPin, -halfHeight, .0,

(pinWidth / 2.0) + dPin, -(halfHeight + bevelPinHeight), .0,

(pinWidth / 2.0) + dPin, -pinHeight, .0,

-(pinWidth / 2.0) + dPin, -pinHeight, .0 // A

);

doc.SketchManager.InsertSketch(true);

doc.SelectSegments(pinProfileSketchSegments);

var pinProfileFeature = doc.FeatureManager.FeatureCut3(false, false, false, 0, 0, 0.01, 0.01, false, false, false, false,

1.74532925199433E-02, 1.74532925199433E-02, false, false, false, false, false, true, true, true, true, false, 0, .0, false);

pinProfileFeature.Name = "Pin Profile Feature";

doc.ClearSelection2(true);

// Create other pins

var directionVertex = ((dynamic[])bodyFeature.GetFaces())

.SelectMany(f => ((object[])f.GetEdges()).Cast<IEdge>())

.Select(e => new

{

Start = (double[])e.GetStartVertex().GetPoint(),

End = (double[])e.GetEndVertex().GetPoint()

})

.Where(v =>

{

var edgeWidth = Math.Abs(v.Start[2] - v.End[2]);

return (Math.Abs(edgeWidth - length) < 0.001 && v.Start[2] > v.End[2]);

})

.First();

doc.Extension.SelectByID2("", "EDGE", directionVertex.Start[0], directionVertex.Start[1], (directionVertex.Start[2] + directionVertex.End[2]) / 2.0, false, 1, null, 0);

doc.Extension.SelectByID2("Pin Profile Feature", "SOLIDBODY", 3.68237484639167E-03, -1.38341410382736E-03, -3.5620114174435E-03, true, 256, null, 0);

var otherPinsFeature = doc.FeatureManager.FeatureLinearPattern3(pinCount / 2, pinDistance, 1, 0.01, true, false, "NULL", "NULL", false, false);

otherPinsFeature.Name = "Other Pins Feature";

}

}

}

Повний код проекта - <https://github.com/chaliy/studies-octo-adventure/tree/master/lp/c3_1/sapr/src> . Використовується WPF, MVVM паттерн(MVVM Light), Visual Studio 14, Sqlite3. Протестовано на Solidwokrs 2014.

## Результат

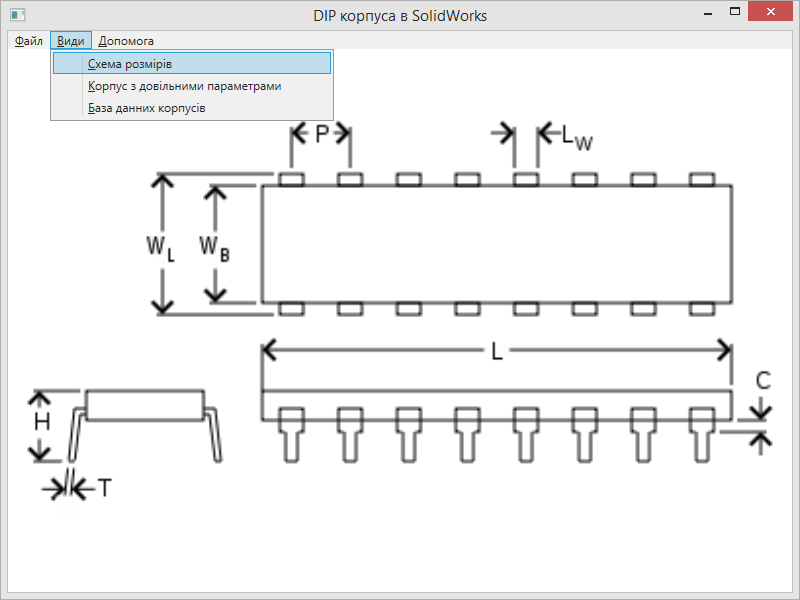


Рисунок 1. Головний екран програми

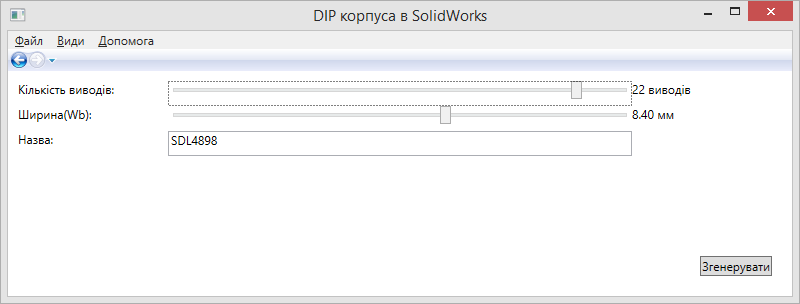


Рисунок 2. Екран генерації копрупуса з довільними параметрами

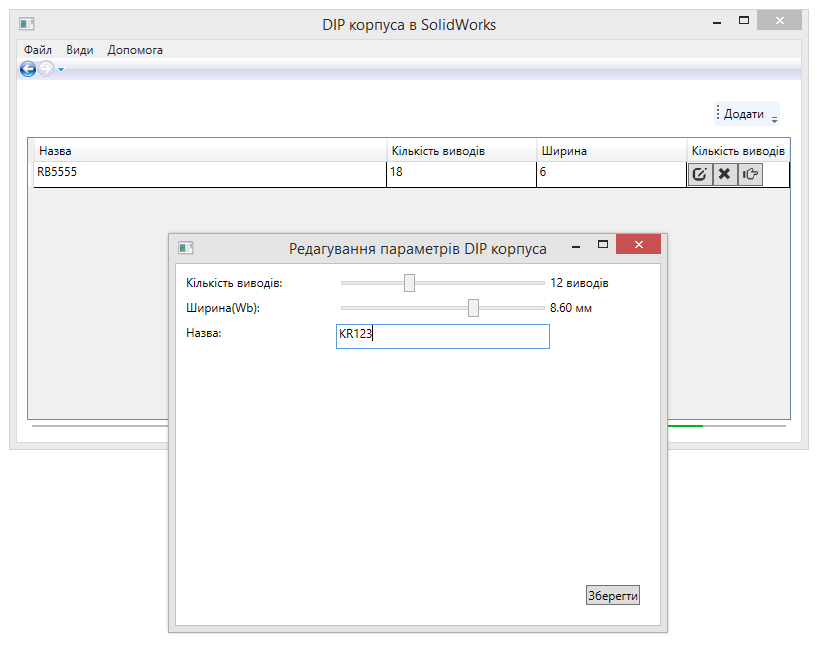


Рисунок 3. Екран бази данних корпусів

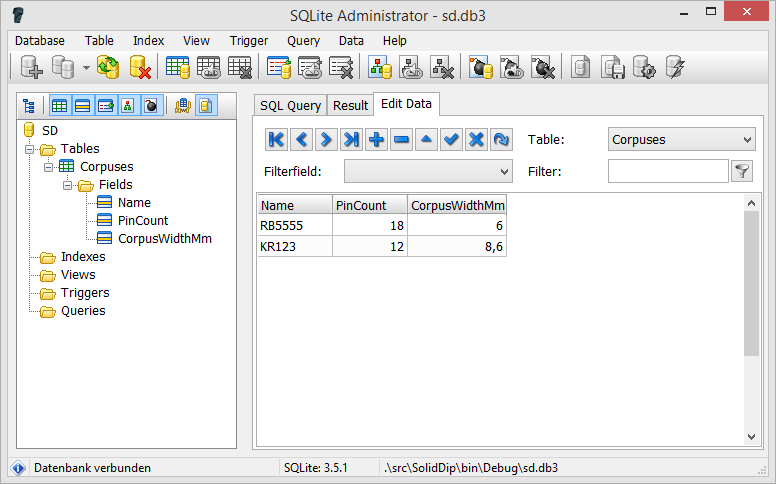


Рисунок 4. Данні збережені в базі данних

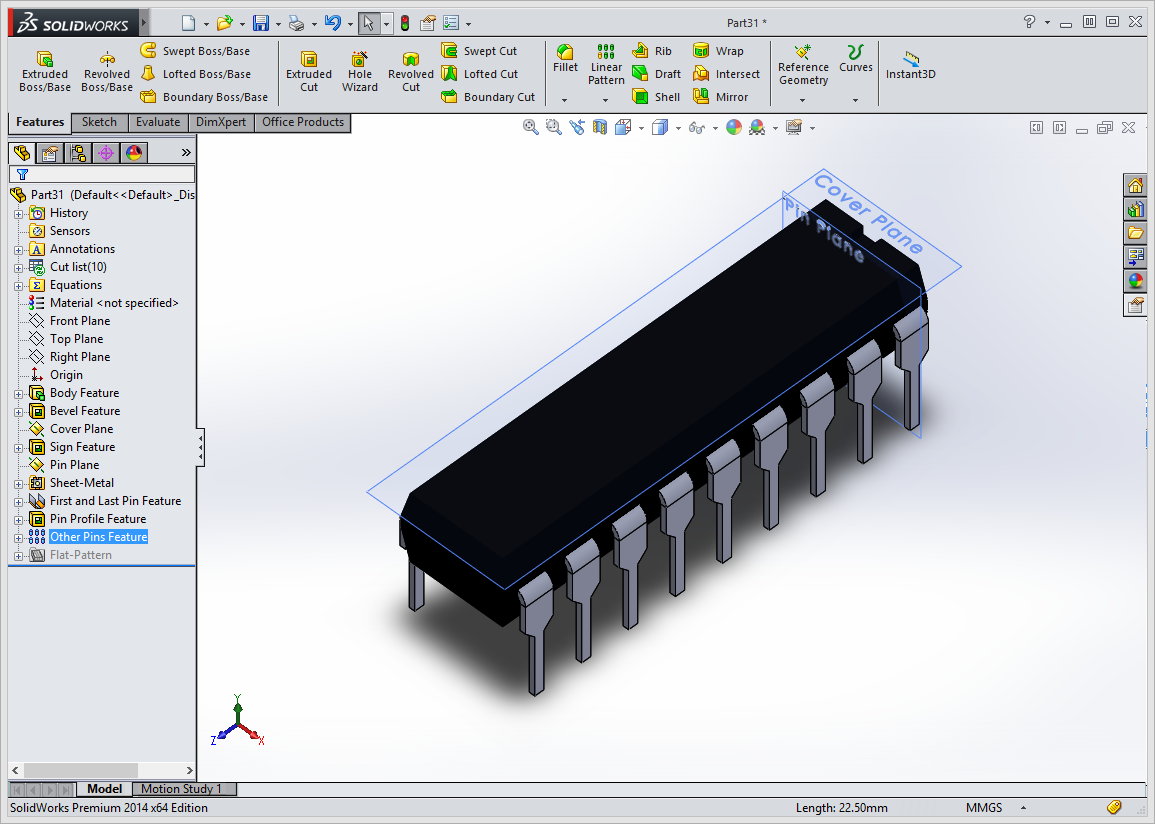


Рисунок 5. Результат роботи програми

## Висновки

Виконуючи лабораторну роботу я ознайомився та здобув базові навички проектування 3D моделей в системі SolidWorks. Крім того, я ознайомився з функціями SolidWorks API для .Net.