愛知教育大学研究報告，55（芸術・保健体育・家政・技術科学・創作編），pp. 35～42， March, 2006

リアル4サイクルエンジンのシミュレーションプログラムの開発

大西研治*　仁木いずみ**

Kenji OHNISHI　　Izumi NIKI

*技術教育講座
**静岡大学教育学部附属島田中学校

1．緒　言

最近の指導要領の改訂により，中学校技術·家庭科におけるエネルギー変換に関わる重要な要素の一つであるエンジンに関する事項が選択となり，教える中学校が減少する傾向にある。

本研究においては，エネルギー変換の教育内容の充実を図るため，その内容の一つであるエンジンのシミュレーションプログラムを作成することとした。第1弾目としてリアル4サイクルエンジン・シミュレーションプログラムを作成した。

プログラミングの記述には，構造化プログラミングの手法を用い，言語はVisual Basic 5.0と6.0とした。

2．プログラム開発に当たり考慮した事項

2.1　基本構成

・ディスプレイの解像度を上げるとform画面の大きさが小さくなるため，本研究ではform画面の大きさを常にディスプレの大きさの80％（Formモジュール上で変更可能）となる標準モジュールを作成する。
・以前のBasicではあったPaintステートメントが無いため，API関数のFloodFillを使いペイント用標準モジュールを作成する。

2.2　プログラム記述の書式

プログラムの記述の書式として，以下の点を考慮し記述することとした。

・グローバル変数はGeneralですべて変数型を定義する。
・プログラムの管理と理解を助けるため，グローバル変数名の前部に変数の型（Int，Dbl,…，但しSingleは除く）を付置する。
・本プログラム上，定まった数値などを持つ変数は，fromでその値などを定義する。
・記述には構造化プログラミングの手法を用い，さらにプログラムの手続きがより認識しやすいよう手続きを説明するレム文を書くなど，プログラムの解釈や修正が容易となるような記述法をとる。
・各プロシージャには，関係したGlobal変数をレム文で記述し，次にプロシージャで使用するLocal変数の型を定義する。
・PicturesとCommandはIndexを用いて記述する。
・エンジンのパーツごとにサブルーチンプロシージャを作り，引数は描画オブジェクト名，描画座標と色などとし，パーツ表示などにも応用できるようにする。
・フォームモジュールのプロパティや，From画面，Picture画面，CommandおよびLabelなどの種々の設定は，Fromに記述し，プログラムの理解しやすくする。
・シミュレーション開始の第1面は，timerに入る前とし，Pictureの2画面の不動パーツ及び開始画面の移動パーツなどはFormオブジェクトに記述する。

2.3　付加すべき機能

・タイミングダイアグラムを表示する。
・シミュレーションの表示切り替え速度を制御する。
・シミュレーションのコマ送りができるようにする。

2.4　エンジンの描画

エンジン本体部を描画する際，実現する点を以下に述べる。

・混合気や燃焼ガスの変化の状態が理解しやすいように数種類の色を用いる。
・エンジンシミュレーションのバルブの開閉が解りやすくするため，全開，全閉と双方の中間の状態を作る。
・実際はカム等で押されて開閉するバルブを矢印を使って押されている状態を描画する。
・オーバーラップ時の混合気や燃焼ガスの流れが解りやすいように描画する。
・混合気のシリンダ内への流入や燃焼ガスのシリンダ

外への排出を各々のマニホールドの横に大きめの矢印を用い描画する。
・エンジンの各パーツ等の色は，シルバーに近い灰色とした。
・描画時の色指定は，ラインにはQBColorを使い，ペインティングの色には，RGB（）を使う。
・燃焼時の点火の炎の広がる過程を解りやすくするために，描画時に配慮する。

2.5 タイミングダイアグラムの描画

タイミングダイアグラムを描画する際，実現する点を以下に述べる。
・タイミングダイアグラムの描画は，一つの円で示す方法もあるが，リアルエンジンシミュレーションとしたため，2重螺旋の表示形式とし，エンジンの状態が解るように小さな円のマーカを螺旋に沿って移動させる。
・タイミングダイアグラムにおいて，吸排気のオーバーラップや点火時の炎の広がり方などが分かりやすいように表示する。

2.6 表示画面の構成

図1示すように，Form画面の最上部の制御コマンドを表示し，残りをPicture画面とした。また，Picture画面を3つの部分に分け，左上部にプログラム名を，左下部にエンジン・タイミングダイアグラムを，右側にエンジン・シミュレーションを表示することとした。

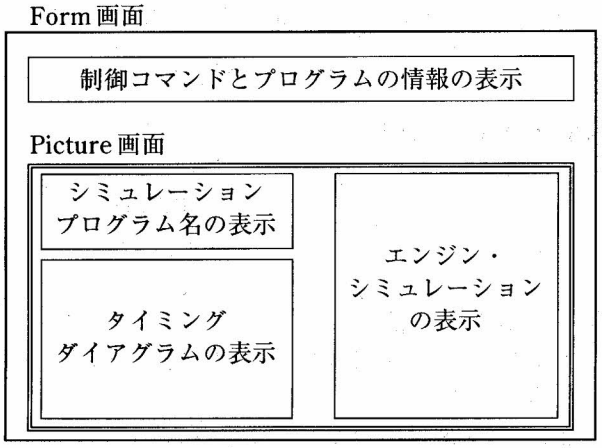

図1 Form画面とPicture画面の配置

3．プログラムの作成

本プログラムは，2つの標準モジュールと1つのフォームモジュールから成り立っている。

3.1 標準モジュール

a．フォームのクライアント領域のサイズを画面のサイズ比より設定するモジュール

ディスプレイの解像度によりFormの大きさが変わらないように，ディスプレイ画面のサイズ比でFormの大きさが設定できる標準モジュールを作成した。これを組み込んだプログラムを動かしたところ，いくつかのディスプレイではシリンダブロックの下部のシリンダ壁とクランク室壁の交点に隙間が発生し色漏れが生じた。起因する要因としては，Circleステートメントによる円と同じ半径の計算で求めた円弧とが画面の上下方向のずれることによるものであった。簡単な解決策としてはCircleステートメントの円と計算で描く円弧が重なるようにForm画面の縦横比を取れば解決されるが，本格的な解決策については，今後も検討していきたい。本プログラムでは，Form画面の縦横比を1.283とし，多くのディスプレイで確認したところ色漏れの発生はなかった。

b．画面の領域を現在のブラシで塗りつぶす標準モジュール

サブプロシージャを作る方法もあり種々の書籍にも掲載されているが，今後のプログラム開発のツールとするため標準モジュールとした。さらに，標準モジュール1を使用するため，色塗り時に生じる座標のずれを補正するScaleXとScaleYの行を組み入れた。

標準モジュールは，画面の領域を現在のブラシで塗りつぶすAPI関数FloodFill（又はExtFloodFill）の宣言と指定されたオブジェクトの領域の指定された色の境界内を塗りつぶすFuncPaintObject関数の定義がされている。

3.2 フォームモジュールの基本構成

General，From，Command，Label，PictureとTimerの6つのオブジェクトとパーツ描画用などの31のサブプロシージャからなる。

3.3 作成するサブプロシージャ

3.3.1 計算用プロシージャ

往復スライダ・クランク機構の連接棒などの座標計算するSub RensetuCalculateとクランクのバランシング・ウエイト描画のための座標計算するSub CrankCalculateからなる。

3.3.2 エンジンパーツ描画用プロシージャ

不動パーツとして，シリンダヘッド，シリンダブロック，クランクケース及びスパークプラグなどと，移動パーツとして，ピストン，連接棒，クランクバルブなどからなる。

3.3.3 再描画のための移動パーツなど消去用プロシージャ

移動パーツと混合気，燃焼ガスなどを消去（バック

カラーで塗りつぶす）するSub Brackからなる。
描画を消去する手順は，以下の通りである。
①クランク中心座標を中心にしてRinkakuの色で円を描く。
②クランク中心座標を塗りつぶしの開始点としてLngBcolorで塗りつぶし，消去する。
③①で描いた円の輪郭をLngBcolorで上から描き直す。
④シリンダブロック左下座標を始点，右上を終点として長方形を描く。
⑤④で描いた長方形をLngBcolorで塗りつぶす。
⑥④で描いた長方形の輪郭をLngBcolorで上から描き直す。

3.3.4 エンジンタイミングダイアグラム描画用プロシージャ

エンジンタイミングダイアグラムの文字を表示するSub VTMojiHyoujiとエンジンタイミングダイアグラムを描くSub ValveTimmingDispからなる。また，マーカーについては，Sub VTMarkerにおいて現在の行程を表示し，さらに吸排気のオーバーラップ時にはマーカーを二つ同時に表示した。

3.4 プログラムの手続きの流れ

2つの標準モジュール，General，Formオブジェクトと Timerオブジェクトプログラムの手続きの概略を以下に示す。

3.4.1 標準モジュール

a. Function FuncSetFormSizeの構成は，

1. ローカルな変数の宣言。
2. 非クライアント領域の幅，高さを求める（Twip）。
3. クライアント領域を求める（Twip）。
4. クライアント領域をTwipsPerPicの倍数に設定する（Twip）。
5. フォームサイズを設定する 。

以上の5つの手続きで構成されている。プログラムリストをList 1に示す。

b. FuncPaintObjecの構成は，

0. API関数FloodFillの宣言。
1. FuncPaintObject関数の定義。
2. ローカル変数の宣言。
3. デバイスコンテキストのハンドルを取得。
4. 塗りつぶしのスタイルを設定。
5. 塗りつぶしのスタイルを設定解除。
6. 再描画を実行。

以上の7つの手続きとAPI関数の宣言とで構成されている。プログラムリストをList 2に示す。

3.4.2 Generalの書式

FormモジュールのGeneralには，以下のものを記述する。

1. 変数名とそのデータ型の設定（配列）
   - 描画の開始角度と描画回数などの変数名と型の設定
   - 主要パーツの大きさ及び描画座標等の変数名と型の設定
   - VTDに関わる変数名と型の設定
   - プログラムで用いる文字やペインティング用の色名の変数名と型の設定
2. 定数の設定
   - 円周率

3.4.3 フォームオブジェクトの構成

フォームオブジェクトに記述する手続きは，以下の通りです。

1. プログラムに用いる変数の値の設定。
2. 標準モジュールFuncSetFormSize関数の呼び出し，表示画面の大きさを設定。
3. 各オブジェクトのバックカラーを同一色にするためにLngBcolorを設定。
4. Formに関わる設定。
   Formの座標をユーザ定義で設定
   FormをScreenの中央に表示する
   FormのBackColorをLngBcolorに設定し，背景を統一。
5. プログラムの多重起動の防止。
6. Picturesオブジェクトの設定（Index使用）。
   オブジェクトの幅と高さ，配置するXY座標，描画線の太さ
   グラフィックが重なっても表示するように再描写の設定
7. Command0オブジェクトの設定（index使用）。
8. Labelオブジェクトの設定。
9. 各種変数の値の設定。
   本体とガスと炎の色の設定
   インテイクバルブとエキゾーストバルブの全開，全閉，中間の三つのXY座標の設定
   クランクケース内のオイルラインを描画する位置などの変数の値の設定
10. 不動パーツをPictures(0)，Pictures(1)に描写。
11. 移動パーツをPictures(0)，Pictures(1)に描写（バルブ）。
12. タイマーの反復回数の初期値の設定（Intn）。
13. 吸気と排気ガスの入れ替わりを描画するために必要なラインの座標値の設定。
14. 移動パーツの初期画面をPictures(0)描画。
15. 初期画面の表示で一時停止。

3.4.4 タイマーオブジェクトの構成

タイマーオブジェクトに記述する手続きは，以下の通りです。

1. 描画回数の奇数と偶数により，Pictureオブジェクトを切り替える。
2. 描画回数に合わせ，クランク角度の加算。
3. クランク・スライダー機構の基本計算とクランク描画用座標計算。
4. クランク，連接棒，ピストンとピストンリングの移動パーツを描画。
5. Case文を用い描画回数の進行に合わせ，インテイクバルブ，エキゾーストバルブのOPEN, MIDDLE, CLOSE状態，バルブ矢印，吸気・排気ガス矢印を描画，及び点火の制御の描画などの制御。
6. Case文を用い描画回数の進行に合わせ，吸気·排気とオーバーラップ時の吸排気のミックスの部分の描画の制御。
7. ZOrderを用い，描画画面を切り替える。
8. 行程を繰り返し描画するため，描写回数の初期化。
9. コマ送りフラッグによるTimerの制御。

3.4.5 その他の事項

プログラムを試行しながら作成したが，移動パーツの消去時に接触する不動パール部のラインの一部が消去されることが数カ所で発生した。

1. バルブの開閉時シリンダヘッドのバルブとの接触部に生じる色漏れ。
2. 点火描画時のプラグ頭部に生じる色漏れ。
3. シリンダ内の消去時シリンダ壁線の線幅の変化。

これらについては個々に対応し，プログラム中にレム文でその旨を記載した。

また，ラインの太さを1ドットから2ドットに変更し，シリンダ径を1ドット大きくした結果，吸排気等の色塗り時に色漏れが生じたが，関連するラインの座標修正で解決した。

4 画面の説明

画面の進行に合わせて，順次説明をする。

4.1 初期画面

実行した初期画面を図2に示す。2.6で構成を考えた表示画面の通りに実現されている。また，本プログラムではフォームの大きさは，ディスプレイの大きさの約80％とした。

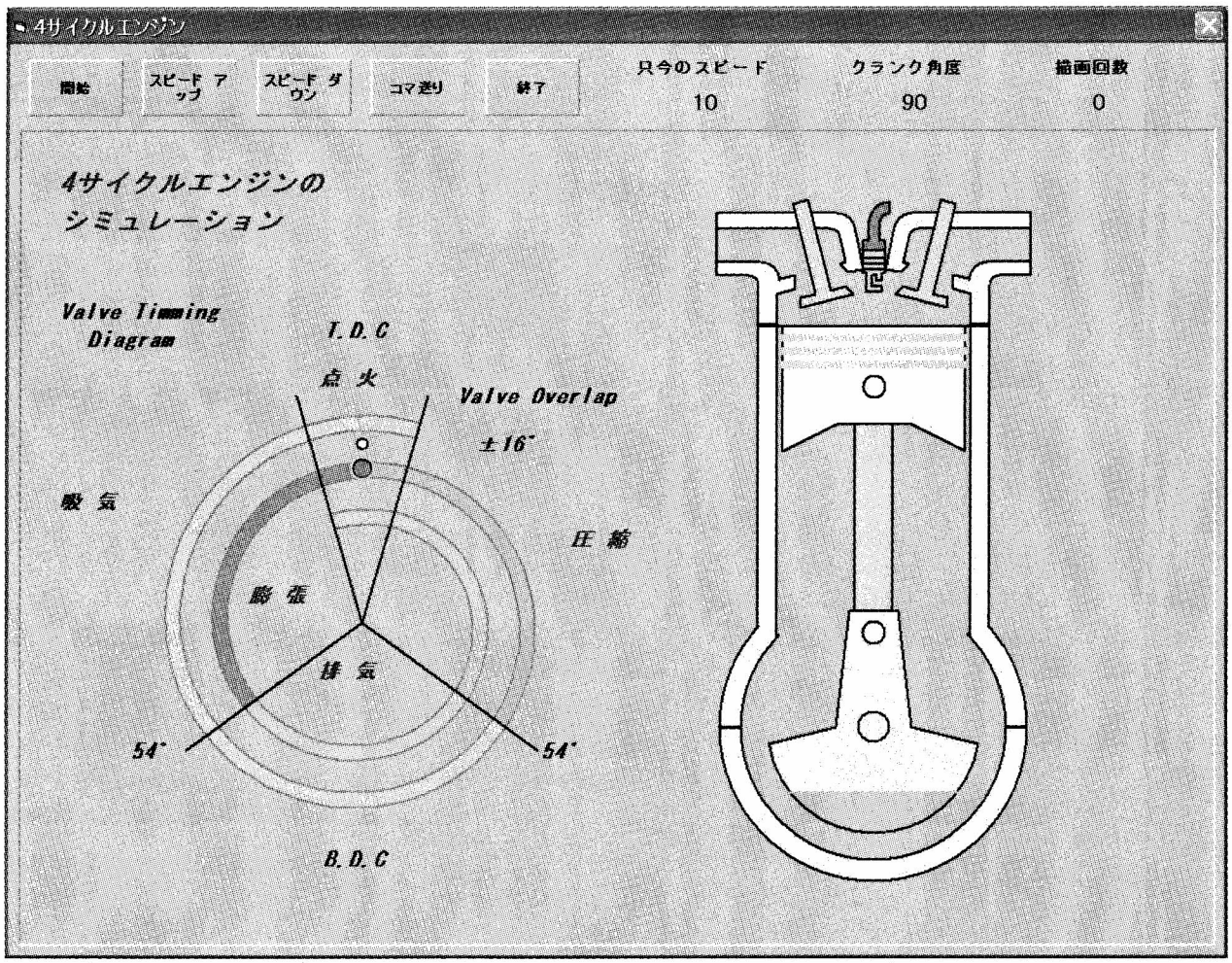

図2 シミュレーションの初期画面

4.2 点火の描画

スパークプラグによる点火の炎の広がりかたと点火が上死点より前で点火することを示すため，図3と図4に示すように2画面を用い表示した。

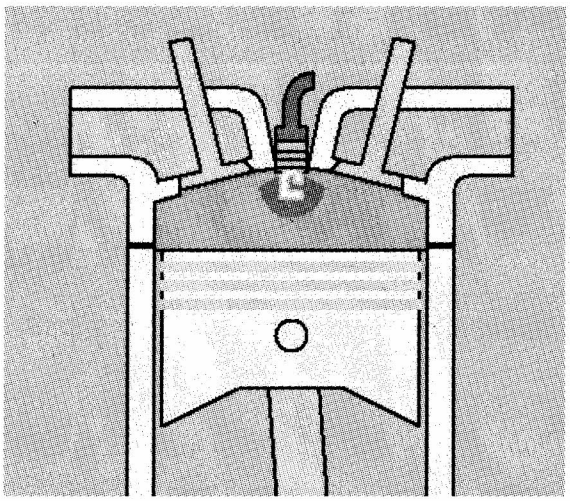

図3　点火描画1

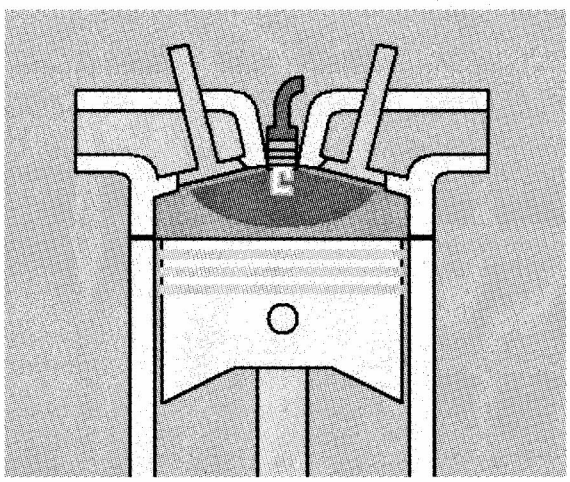

図4　点火描画2

4.3 吸気と排気の入れ替わりの描画

吸気と排気の入れ替わる過程を図5～8までの4画面を使い描画した。吸気，排気及び吸気と排気のミックスされた部分の3種の色を使い，吸気と排気の入れ替わる過程が視覚的に理解しやすいものとした。これらを実現するため，図9に示す6本のガス境界ラインと図10に示す9カ所のガスのペイントポイントを使用した。同時に図11に示すようにオーバーラップ時にバルブタイミングダイヤグラムにおいて吸気と排気の行程に2つのマーカーを同時に表示させ，吸気と排気の過程を理解しやすくした。

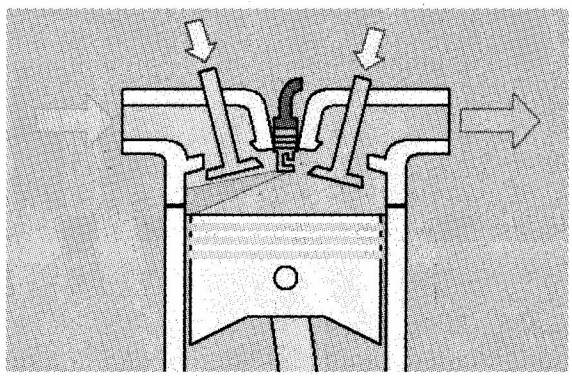

図5　入れ替わり1

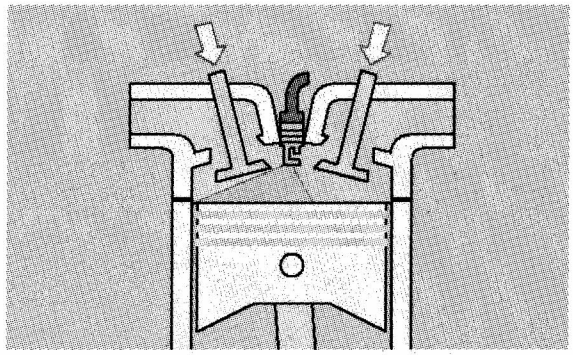

図6　入れ替わり2

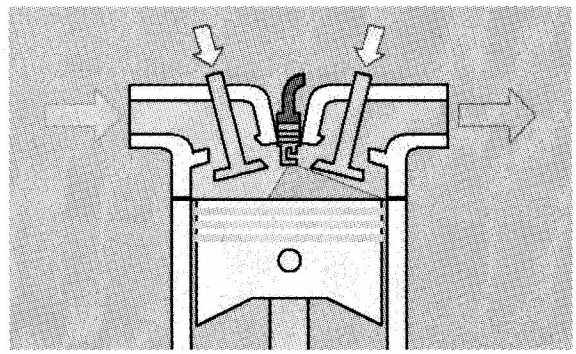

図7　入れ替わり3

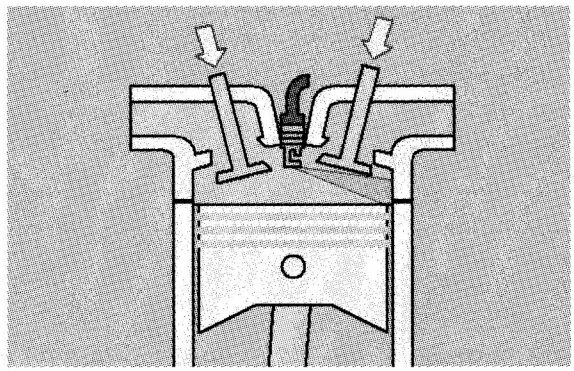

図8　入れ替わり4

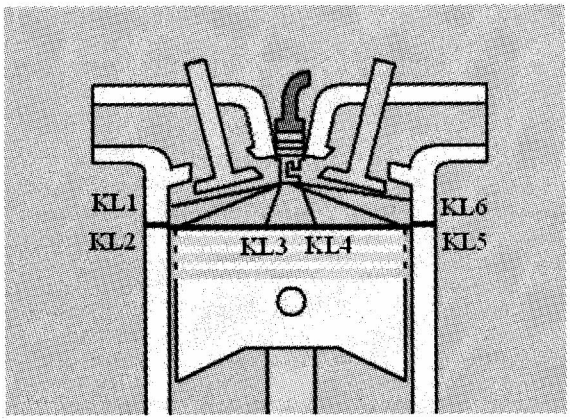

図9　吸排気の描画用ライン

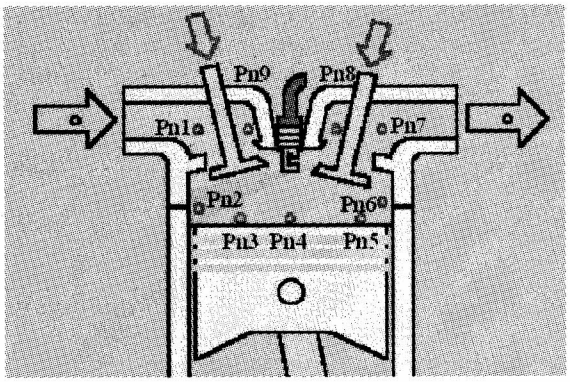

図10　吸排気の色塗りポイント

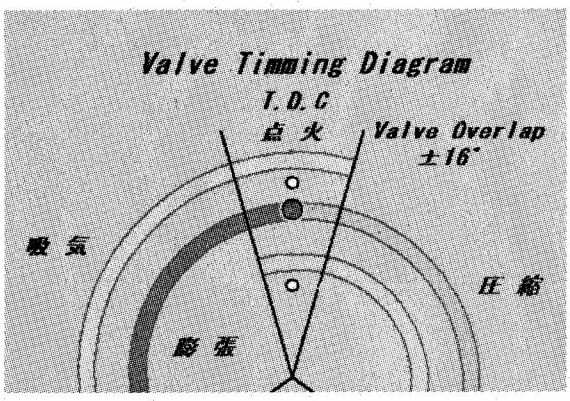

図11　Overlap 時のマーカーの同時表示

4.4　バルブタイミングダイアグラムの描画

バルブタイミングダイアグラムの描画方法として幾つか考えられるが，本研究においてはオーバーラップ等の描画をすることとしたため，図12に示すような螺旋状のダイアグラムを選択した。結果，オーバーラップの理解と説明が行いやすいものとなった。

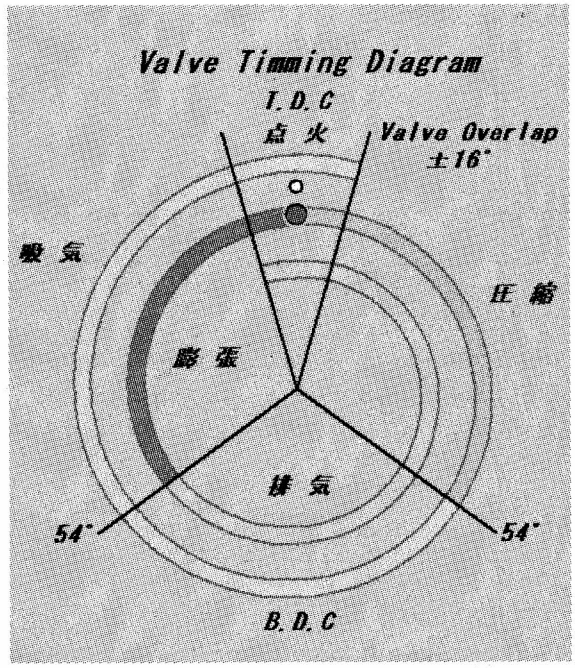

図12　バルブタイミングダイヤグラム

5　考　察

2.4で述べたエンジンの描画に関して，８つの事項については，ほぼプログラム上で実現されている。2.5のタイミングダイヤグラムの描画では２つの事項が見やすく実現されている。

また，構造化プログラミングの手法とレム文による手続きの記載，変数宣言，Global変数の型が解るように変数型を示すため変数の前部にInt, Lngなどを付加した等の処置によりプログラムの解釈，修正が容易になった。

今後の課題として，新たにFormモジュールを作成し，プログラム名やプログラムの内容等を表示することや，一つ一つのエンジンパーツの表示ができるようにしたい。

色抜けの原因となった標準モジュールFuncSetFormSizeで生じた問題を解決する方法について今後研究していきたい。なお，標準モジュールFuncSetFormSizeで生じた問題を簡単に解決するため，Circleステートメントと計算による円弧を描画し，円と円弧が重なるように調整するためのプログラミンググラムを作成したので，本プログラムを配布する時には添付する。

本プログラムは，プログラムリストと共にホームページ上で公開する予定である。

（平成17年９月９日受理）

List 1　標準モジュールFuncSetFormSizeのリスト

Formサイズをディスプレイの大きさの80％（任意の大きさに変更できる）に指定できるようにしたが，ディスプレイの種類により円弧とラインの接点に隙間が生じ色漏れが生じたため，ディスプレイの幅を基準としてFormサイズを定めるようにした。しかしワイドディスプレイでは高さがディスプレイよりはみ出すこともあるので，ディスプレイの高さを基準にFormサイズを定めることとした。以下のリストには双方を表示した。

①-- ～ --①はディスプレイの高さを基準にFromサイズを決める場合，②-- ～ --②はディスプレイの幅を基準にFromサイズを決める場合である。

```
Function FuncSetFormSize _
    (ByVal ObjFormName As Object, _
    ByVal LngRequestRate As Long)

    'ローカルな変数の宣言
    Dim IntNonClientWidth As Integer        '非クライアント領域の幅
    Dim IntNonClientHeight As Integer       '非クライアント領域の高さ
    Dim IntClientWidth As Integer           'クライアント領域の幅
    Dim IntClientHeight As Integer          'クライアント領域の高さ
    Dim IntRemain As Integer                '余りを格納
    Dim IntObjHeight As Integer             'formの幅を，高さに対する比で算出するための変数
①--
    '高さを基準にした場合
    '非クライアント領域の高さを求める（Twip）
    IntNonClientHeight = ObjFormName.Height - Form1.ScaleHeight
    'クライアント領域の高さを求める（Twip）
    IntClientHeight = Screen.Height * LngRequestRate / 100
    'クライアント領域の高さをTwipsPerPicの倍数に設定する（Twip）
    IntRemain = IntClientHeight Mod Screen.TwipsPerPixelY
    IntClientHeight = IntClientHeight - IntRemain
    'フォームサイズの高さを設定する
    ObjFormName.Height = IntNonClientHeight + IntClientHeight
    'フォームサイズの幅を高さの比で求める
    IntObjWidth = ObjFormName.Height * 1.285
    'フォームサイズの幅をTwipsPerPicの倍数に設定する（Twip）
    IntRemain = IntObjWidth Mod Screen.TwipsPerPixelY
    ObjFormName.Width = IntObjWidth - IntRemain
--①
②--
    '幅を基準した場合
    '非クライアント領域の幅を求める（Twip）
    IntNonClientWidth = ObjFormName.Width - Form1.ScaleWidth
    'クライアント領域を求める（Twip）
    IntClientWidth = Screen.Width * LngRequestRate / 100
    'クライアント領域をTwipsPerPicの倍数に設定する（Twip）
    IntRemain = IntClientWidth Mod Screen.TwipsPerPixelX
    IntClientWidth = IntClientWidth - IntRemain
    'フォームサイズを設定する
    ObjFormName.Width = IntNonClientWidth + IntClientWidth
    IntObjHeight = ObjFormName.Width / 1.283
    IntRemain = IntObjHeight Mod Screen.TwipsPerPixelY
    ObjFormName.Height = IntObjHeight - IntRemain
---②
End Function
```

List 2　標準モジュールFuncPaintObjectのリスト

標準モジュールFuncPaintObjectは，API関数の宣言とFuncPaintObject関数の定義からなる。

```
Option Explicit
'       /////       API関数の宣言       /////
'画面の領域を現在のブラシで塗りつぶすAPI関数FloodFillの宣言
'第1パラメータ：デバイスコンテキストのハンドル
```

```
'第2パラメータ：塗りつぶしの開始点X座標（Pixel）
'第3パラメータ：塗りつぶしの開始点Y座標（Pixel）
'第4パラメータ：塗りつぶす領域の境界の色（RGB）
'第5パラメータ：塗りつぶすの方法（0,1）
Private Declare Function ExtFloodFill Lib "gdi32" _
    (ByVal hdc As Long, _
    ByVal x As Long, _
    ByVal y As Long, _
    ByVal crColor As Long, _
    ByVal wFillType As Long) As Long

'       /////       API関数で用いる変数の宣言       /////
'API関数で用いる定数の宣言
Private Const FLOODFILLBORDER = 0       '指定された色の境界内の塗りつぶしを行うための定数

'指定されたオブジェクトの領域の指定された色の境界内を塗りつぶす
'  FuncPaintObject関数の定義
'第1パラメータ：塗りつぶしを行うオブジェクト名
'第2パラメータ：塗りつぶしの色（RGB）
'第3パラメータ：塗りつぶしのスタイル（FillStyle）
'第4パラメータ：塗りつぶしの開始点のX座標
'第5パラメータ：塗りつぶしの開始点のY座標
'第6パラメータ：塗りつぶす領域の境界の色（RGB）
Function FuncPaintObject _
    (ByVal ObjPaintObject As Object, _
    ByVal LngFillColor As Long, _
    ByVal IntFillStyle As Integer, _
    ByVal IntPointX As Integer, _
    ByVal IntPointY As Integer, _
    ByVal IntBorderColor As Long)

    'ローカル変数の宣言
    Dim LngPixelPointX As Long              'Pixelで表現した塗りつぶしのX座標
    Dim LngPixelPointY As Long              'Pixelで表現した塗りつぶしのY座標
    Dim LngReturn As Long                   'API関数の戻り値
    Dim LngPaintObjectHdc As Long           '指定されたオブジェクトのデバイスコンテキストハンドル

    'デバイスコンテキストのハンドルを取得
    LngPaintObjectHdc = ObjPaintObject.hdc

    '塗りつぶしのスタイルを設定
    ObjPaintObject.FillColor = LngFillColor
    ObjPaintObject.FillStyle = IntFillStyle     '0

    '座標をPixel単位に変更
    LngPixelPointX = ObjPaintObject.ScaleX (IntPointX, 0, 3)
    LngPixelPointY = ObjPaintObject.ScaleY (IntPointY, 0, 3)

    'FloodFill関数を呼び出して塗りつぶしを実行
    LngReturn = ExtFloodFill (LngPaintObjectHdc, _
                              LngPixelPointX, _
                              LngPixelPointY, _
                              IntBorderColor, _
                              FLOODFILLBORDER)

    '塗りつぶしのスタイルを設定解除
  ObjPaintObject.FillStyle = 1   'IntFillStyleの初期値
  '再描画を実行
  ObjPaintObject.Refresh
End Function
```