3.2　基礎制御プログラミング

3.2.1　デジタル出力

簡単なデジタル出力を試します。コンパイル時には、入出力関数のためのライブラリを指定します。

```
// iotest0.c
//    bcc32 iotest0.c cdiobld.lib
#include<stdio.h>
#include<windows.h>
#include"cdio.h"

void main()
{
          short     id;
          int       r,n;
          unsigned char x;
          int i;

          r = DioInit("DIO000",&id);
          printf("ret %d ¥n", r);
          n =0x11;
          for(i=0;i<100;i++){
                    r = DioOutByte(id, 0, ~n);
                    printf("ret %d out %x ¥n", r, n );
                    n =  n << 1 ;
                    if( n == 0x110 )
                              n = 0x11;
                    Sleep(250);
          }
}
```

3.2.2 デジタル入出力

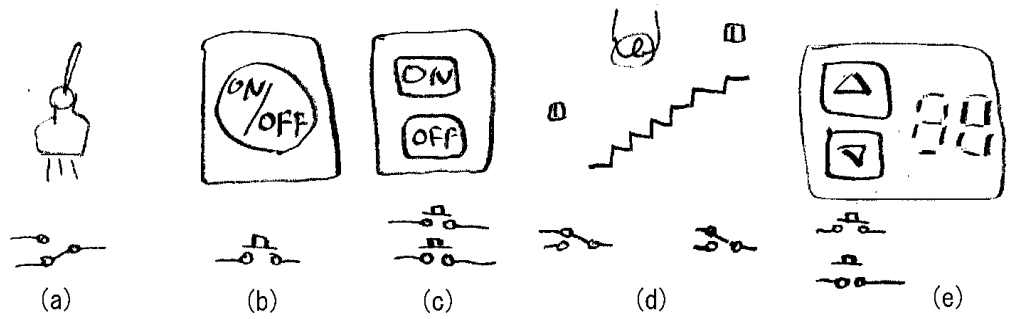
続いて入力を試します。プログラムでは入力したスイッチの内容を LED にエコーするようにします。

```
//
// iotest1.c
//
#include<stdio.h>
#include<windows.h>
#include"cdio.h"

void main()
{
        short    id;
        int      r,n;
        unsigned char x;

        r = DioInit("DIO000",&id);
        printf("ret %d ¥n", r);
        while(1){
                DioInpByte( id, 0, &x );
                n = ~(x & 0x0f);
                DioOutByte(id, 0, n);
                printf("in %x out %x ¥n", x, n );
                Sleep(100);
        }

}
```

3.2.3 スイッチ入力

簡単そうに見えて意外に複雑なのがスイッチの入力です。例えば１つの押しボタンスイッチでも押すタイミングによって複数の機能を持たせることができます。例えばデジタル時計の針送りボタンやマウスのシングルクリック/ダブルクリックなど少し注意すれば多くの事例が見つかります。

ここでは、カウント動作を試します。カウント動作のためにはエッジセンスが必要になります。




```
//
// iotest2.c
//
#include<stdio.h>
#include<windows.h>
#include"cdio.h"

void main()
{
        short    id;
        int      r,n;
        unsigned char x,x0;

        r = DioInit("DIO000",&id);
        printf("ret %d ¥n", r);
        n = 0;
        while(1){
                DioInpByte( id, 0, &x );
                x = x & 1;
                if( x0 != x  && x == 1){
                        n++;
                        printf("in %x n = %d ¥n", x, n );

                }
                x0 = x;
                Sleep(100);
        }

}
```

3.2.4　速度制御

スイッチ入力とモータ出力を使って簡単な速度制御を試みます。

```
//
// iotest3.c
//
#include<stdio.h>
#include<windows.h>
#include"cdio.h"

void main()
{
        short    id;
        int      r,n,m,t;
        unsigned char x;

        r = DioInit("DIO000",&id);
        printf("ret %d ¥n", r);
        m = 0x11;
        while(1){
                DioInpByte( id, 0, &x );
                printf("x = %x¥n",x);
                n = x & 0x0f;
                if( n == 0 )
                        Sleep(100);
                else{
                        t = 1000 / n;
                        DioOutByte(id, 0, m);
                        m = m << 1;
                        if( m == 0x110 )
                                m = 0x11;
                        printf("in %d time %d ¥n", x, t );
                        Sleep( t );
                }
        }

}
```

3.2.5　台形駆動

モータの駆動で使われる台形駆動をプログラムします。

```
//
// iotest4.c       台形駆動
//
#include<stdio.h>
#include<windows.h>
#include"cdio.h"

void sub( int  );

void main()
{
          short     id;
          int       i;
          int       r;

          r = DioInit("DIO000",&id);
          printf("ret %d ¥n", r);

          for(i= 1;i <= 5 ;i++){
                    sub( i );
          }
          for(i= 1;i <= 5 ;i++){
                    sub( 5 );
          }
          for(i= 5;i >= 1 ;i--){
                    sub( i );
          }
}

void sub( int x )
{
          static int n = 0;
          int i,m,t,tt;
          short     id;

          t = 500 / x;
          tt = x * 2;
          printf("%d %d %d ¥n",x ,t,tt);
          for( i = 0; i < tt ;i++){
                    m = 0x11;
                    m = m << n;
                    DioOutByte(id, 0, m);
                    n= (++n) % 4;
                    Sleep( t );
          }
}
```

3.3 応用制御プログラミング

3.3.1 マルチスレッド点滅プログラム

複数の LED の点滅を異なるタイミングで平行して実行します。このような場合、シングルスレッドのプログラムでも記述可能ですが、マルチスレッドにすると簡単に実現できます。

```
//
//  iotest5mt.cpp           console mult thread
//    bcc32 -WM mthc3.cpp
//           +----マルチスレッドのスイッチ

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>                    // getch()の宣言

#include <windows.h>
#include <process.h>                  // スレッド用
#include"cdio.h"

DWORD WINAPI th_Proc( LPVOID );
DWORD WINAPI th_Proc2( LPVOID );

short    id;
int      x;

void main( )
{
         DWORD tid,tid2;
         int r;

         r = DioInit("DIO000",&id);
         printf("ret %d ¥n", r);
         x = 0;
         CreateThread(NULL, // LPSECURITY_ATTRIBUTES lpThreadAttributes,
                        // pointer to thread security attributes
                        0,        // DWORD dwStackSize,
                                  // initial thread stack size, in bytes
                                  // 0はデフォルト
                        th_Proc, // LPTHREAD_START_ROUTINE lpStartAddress,
                        // pointer to thread function
                                  // DWORD WINAPI ThreadFunc(LPVOID);
                        NULL,     // LPVOID lpParameter,
                                  // argument for new thread
                        0,        // DWORD dwCreationFlags,
                                  // creation flags
                        &tid      //  LPDWORD lpThreadId
                                  // pointer to returned thread identifier
                                  );

         CreateThread(NULL, 0, th_Proc2,
                        NULL, 0, &tid2   );

         printf("push any key ¥n");
         getch();

}
```

```
// スレッド
DWORD WINAPI th_Proc( LPVOID )
{
	int i;
	for(i=0;i<10;i++){
		x = x ^ 0x01;
		DioOutByte(id, 0, ~x);
		printf("***¥n");
		Sleep(1000);
	}
	return 0;
}

DWORD WINAPI th_Proc2( LPVOID )
{
	int i;
	for(i=0;i<20;i++){
		x = x ^ 0x04;
		DioOutByte(id, 0, ~x);
		Beep(200,100);
		Sleep(400);
	}
	return 0;
}
```

3.3.2 マルチスレッド版台形駆動

台形駆動も駆動パルス発生動作を独立したスレッドにするとプログラムが簡単になります。

```
//
//  iotest6mt.cpp           console mult thread
//    bcc32 -WM mthc3.cpp
//           +----マルチスレッドのスイッチ

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>                    // getch()の宣言

#include <windows.h>
#include <process.h>                  // スレッド用
#include"cdio.h"

DWORD WINAPI th_Proc( LPVOID );

short    id;
int      n,t;

void main( )
{
         DWORD tid,tid2;
         int i,r;

         r = DioInit("DIO000",&id);
         printf("ret %d ¥n", r);

         CreateThread(NULL, 0, th_Proc,
                           NULL, 0, &tid     );
         n = 0;
         for(i= 1;i <= 5 ;i++){
                  t = 500 / i;
                  printf("%d %d¥n", i , t );
                  Sleep( 1000 );
         }
         for(i= 1;i <= 5 ;i++){
                  t = 100 ;
                  printf("%d %d¥n", i , t );
                  Sleep( 1000 );
         }
         for(i= 5;i >= 1 ;i--){
                  t = 500 / i;
                  printf("%d %d¥n", i , t );
                  Sleep( 1000 );
         }

}
```

```
// スレッド
DWORD WINAPI th_Proc( LPVOID )
{
	int i,m;

	while(1){
		m = 0x11;
		m = m << n;
		DioOutByte(id, 0, m);
		n= (++n) % 4;
		Sleep( t );
	}
	return 0;
}
```

3.3.3 タイマ版点滅プログラム

Windows のタイマ API を使えば、一定間隔毎に任意の関数を呼び出すことができます。この機能も平行動作が必要な場合に役に立ちます。3.3.1 と同様な動作をタイマを使って実現します。

```
//
// iotest7wt.cpp
//  タイマプロシージャのテスト
//
#include<conio.h> // kbhit(),getch()
#include<windows.h>
#include<stdio.h>
#include"cdio.h"

#define ID_ti  100
#define ID_ti2 101

short    id;
int      x;

VOID CALLBACK TimProc(HWND ,UINT ,UINT ,DWORD ti)       //  タイマプロシージャ
{
         x = x ^ 0x01;
         DioOutByte(id, 0, ~x);
         printf("0:何かキーを押すと終了します 経過時間 %d msec ¥n",ti);

         return;
}

VOID CALLBACK TimProc2(HWND ,UINT ,UINT ,DWORD ti)      //  タイマプロシージャ
{
         x = x ^ 0x04;
         DioOutByte(id, 0, ~x);
         printf("2:何かキーを押すと終了します 経過時間 %d msec ¥n",ti);

         return;
}

int main( )
{
         MSG msg;
         int r;

         r = DioInit("DIO000",&id);
         printf("ret %d ¥n", r);

         SetTimer(NULL, ID_ti,  1000,(TIMERPROC)TimProc);
         // 1秒毎にTimProcを呼び出す
         SetTimer(NULL, ID_ti2,  500,(TIMERPROC)TimProc2);
         // 0.5秒毎にTimProc2を呼び出す
         while( ! kbhit() ){
                  GetMessage( &msg, NULL, 0, 0 );
                  DispatchMessage( &msg );
         }
         printf("メッセージポンプを終了しました¥n");
}
```

3.3.4 ダイアル操作

少しプログラムが複雑になりますが、Windows の GUI からモータを制御する例を示します。

```
//
// dial2.cpp
//
#include <windows.h>
#include<stdio.h>
#include<math.h>
#include"cdio.h"

LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM );
                                                        // ウィンドウプロシージャの宣言
short     id;

int WINAPI WinMain( HINSTANCE hi, HINSTANCE, LPSTR, int )       // 第１引数以外は使わない
{
          WNDCLASSEX wc;                              // 新しくつくるウインドクラス用

          int       r,n;
          unsigned char x;

          DioInit("DIO000",&id);

          memset( &wc, 0, sizeof( wc ));
          wc.cbSize        = sizeof(WNDCLASSEX);              // WNDCLASSEXの大きさ
          wc.lpfnWndProc   = WndProc;                         // このクラスの持つウインドプロシージャ
          wc.hInstance     = hi;
          wc.hIcon         = LoadIcon( NULL, IDI_APPLICATION );
          wc.hIconSm       = LoadIcon( NULL, IDI_WINLOGO );
          wc.hCursor       = LoadCursor( NULL, IDC_ARROW );
          wc.hbrBackground = (HBRUSH)GetStockObject( WHITE_BRUSH );
          wc.lpszClassName = "wh04";                          // このウィンドウクラスの名前

          if(! RegisterClassEx( & wc )) return 0;             // ウィンドクラスの登録
                                                                    // できないと終了
          CreateWindow( "wh04",                                     // クラスの名前
                        "dial",                                     // キャプションの内容
                        WS_OVERLAPPEDWINDOW | WS_VISIBLE,           // ウィンドウの属性, 初期状態は表示
                        CW_USEDEFAULT, CW_USEDEFAULT,               // 位置、大きさは指定しない
                        200, 150,                                   // WS_OVERLAPPEDWINDOWのときのみ可
                        HWND_DESKTOP,                                     // 親はディスクトップ
                        NULL, hi, NULL                                  );  // ウィンドウプロシージャへ
                                                                          // 渡すパラメータなし
          MSG msg;
          while( GetMessage( &msg, NULL, 0, 0 ) ){
                    TranslateMessage( &msg );
                    DispatchMessage( &msg );
          }

          return msg.wParam ;
}

LRESULT CALLBACK WndProc(  HWND hwnd,    UINT msg,
                                    WPARAM wp, LPARAM  lp  )
{
          static int sw = 0;              // ダイアル内に入ったかどうかのスイッチ
          static int k0 ;                           //  直前の角度
          static int kk = 0;              //  矢印の角度
          char cbuf[100];
          int k, kk2,kkk,kkkk, n,m;
          int x,y;

          switch(msg){
          case WM_MOUSEMOVE:{
                    x =   LOWORD(lp) - 60;
                    y = -(HIWORD(lp) - 60);
                    if( x*x +y*y < 2500 ){          // ダイアル内にマウスがある
                              if(x !=0 )
                                        k = (int)(atan((float)y/(float)x) / 3.1415 /2.0 *360.0);
                              else{
                                        if(y > 0)
                                                  k = 90;
                                        else
                                                  k =-90;
                              }
```

```
                        if(x<0){
                                if(y>0)
                                        k = 180 + k;
                                else
                                        k = k - 180;
                        }
                        // k = -180 ～ 179
                        if(sw == 0){            //  マウスが最初にダイアル内に入った
                                sw = 1;
                                k0 = k;                 //  その時の角度を保存
                        }
                        else{
                                int d = k - k0;         //  直前の角度からの変化分の計算
                                if(d < -300)
                                        d = d + 360;
                                if(300 < d )
                                        d = d - 360;

                                //  矢印の描画
                                int x = (int)(sin((float)kk / 360.0 * 3.1415 * 2.0) *45);
                                int y = (int)(cos((float)kk / 360.0 * 3.1415 * 2.0) *45);
                                HDC hdc = GetDC(hwnd);
                                SelectObject( hdc, GetStockObject( WHITE_PEN ) );
                                //  もとの矢印を消す
                                MoveToEx( hdc, 60, 60, NULL );
                                LineTo( hdc,60 + x,60 + y );
                                //  新しい矢印を描く
                                kk = (kk + d) % 360;
                                x = (int)(sin((float)kk / 360.0 * 3.1415 * 2.0) *45);
                                y = (int)(cos((float)kk / 360.0 * 3.1415 * 2.0) *45);
                                SelectObject( hdc, GetStockObject( BLACK_PEN ) );
                                MoveToEx( hdc, 60, 60, NULL );
                                LineTo( hdc,60 + x,60 + y );
                                ReleaseDC( hwnd, hdc );
                                k0 = k;                 //  直前の角度の更新

                                if(kk < 0)
                                        kk2 = kk + 360;
                                else
                                        kk2 = kk;
                                kkk  = kk2 / 18;
                                kkkk = kkk % 4;
                                wsprintf(cbuf,"%d %d %d",kk,kk2,kkkk);
                                SetWindowText( hwnd , cbuf );

                                n = 0x80 >> kkkk;
                                m = 1 << kkkk;

                                DioOutByte(id, 0, ~(n | m));

                        }
                }
                else{                                   //  マウスが外に出た
                        sw = 0;
                }
                break;
        }
        case WM_PAINT:{
                PAINTSTRUCT ps;
                BeginPaint( hwnd, &ps );
                Ellipse(ps.hdc,10,10,110,110);
                x = (int)(sin((float)kk / 360.0 * 3.1415 * 2.0) *45);
                y = (int)(cos((float)kk / 360.0 * 3.1415 * 2.0) *45);
                MoveToEx( ps.hdc, 60, 60, NULL );
                LineTo( ps.hdc,60 + x,60 + y );
                EndPaint( hwnd, &ps );
                break;
        }
        case WM_DESTROY:                        // 必ず必要, DefWindowProcでは処理されない
                PostQuitMessage( 0 );                   // メッセージループを終了させる
                break;
        default:
                return DefWindowProc( hwnd, msg, wp, lp );
        }
        return 0;
}
```

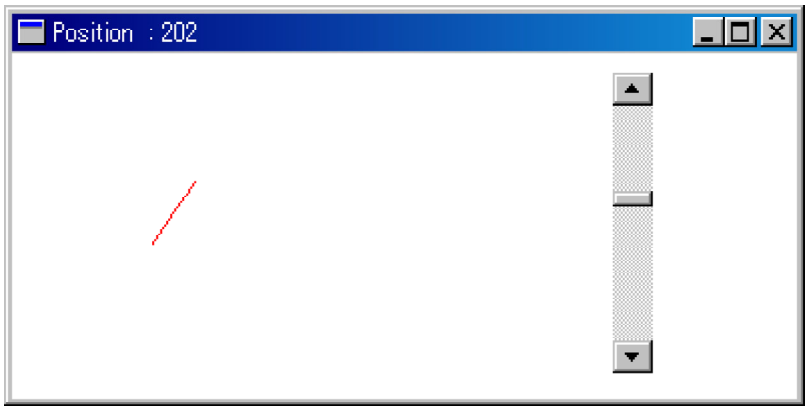
3.3.5 GUIからの制御

Windows のスクロールバー・コントロールを使ったモータの速度制御の例を示します。

```
// vkuru00.cpp
// スレッドを使った並列動作とその制御
//

#include  <stdio.h>
#include  <windows.h>
#include <math.h>
#include <process.h>
#include"cdio.h"

int       vpos;                                    // スクロールの位置
int       vmin = 10,     vmax = 500;

HWND      hMwnd, hScrl,  hText[2];

LRESULT CALLBACK WndProc( HWND, UINT, WPARAM, LPARAM );

DWORD WINAPI sub( LPVOID );

typedef struct{                                    // 描画位置の情報
          HWND hw0;
          int       xc,yc;
          int ti;
          HANDLE th;
}HXY;

short     id;

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE, LPSTR, int nCmdShow )
{
          MSG                 msg;
          WNDCLASSEX wndclass;
          SCROLLINFO si;
          char                str[100];
          int                 i;

          DioInit("DI0000",&id);

          vpos = 200;

          // ウィンドウクラスの定義
          wndclass.cbSize        = sizeof( WNDCLASSEX );              // WNDCLASSEX構造体のサイズ
          wndclass.style         = CS_HREDRAW | CS_VREDRAW;   // ウィンドウの型
          wndclass.lpfnWndProc   = WndProc;                           // プロシージャへのアドレス
          wndclass.cbClsExtra    = 0;                         // エキストラクラス情報
          wndclass.cbWndExtra    = 0;                         // エキストラウィンドウ情報
          wndclass.hInstance     = hInstance;                 // インスタンスのハンドル
          wndclass.hIcon         = LoadIcon( NULL, IDI_APPLICATION );
                                                              // ラージアイコンのハンドル
          wndclass.hIconSm       = LoadIcon( NULL, IDI_APPLICATION );
                                                              // スモールアイコンのハンドル
          wndclass.hCursor       = LoadCursor( NULL, IDC_ARROW );     // マウスカーソルのハンドル
          wndclass.hbrBackground = (HBRUSH)GetStockObject( WHITE_BRUSH );        // 背景の色
          wndclass.lpszMenuName  = NULL;                      // メインメニュー名
          wndclass.lpszClassName = "scroll";                  // ウィンドウクラス名

          //ウィンドウクラスの登録（出来なければ終了）
          if( !RegisterClassEx( &wndclass ) )  return 0;

          //ウィンドウの生成
          hMwnd = CreateWindow( "scroll",                     // ウィンドウクラスの名前
                                "Scrollbar Example",          // ウィンドウのタイトル
                                WS_OVERLAPPEDWINDOW ,               // ウィンドウスタイル
                                CW_USEDEFAULT, CW_USEDEFAULT,  // ウインドウ作成位置(x,y)
                                400, 200,                     // ウィンドウのサイズ（幅、高さ）
                                HWND_DESKTOP,                       // 親ウィンドウのハンドル
                                NULL,                               // メインメニューのハンドル
                                hInstance,                    // アプリケーションインスタンスのハンドル
                                NULL );                             // 追加情報へのポインタ

          hScrl = CreateWindow( "SCROLLBAR","",
                                WS_CHILD | WS_VISIBLE | SBS_VERT,
                                300, 10,   20, 150,
```

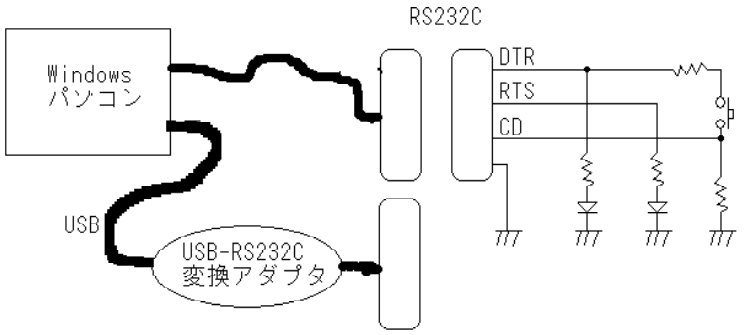
```
                        hMwnd,
                        NULL, hInstance,    NULL);

        si.cbSize = sizeof( SCROLLINFO );                   // スクロール情報構造体の設定
        si.fMask  = SIF_POS | SIF_RANGE | SIF_PAGE;         // 位置, 範囲, つまみの大きさ指定する
        si.nPos   = 200;                                    // 位置
        si.nMin   = vmin;                                   // 範囲
        si.nMax   = vmax;
        si.nPage  = 1;                                      // ページサイズ, つまみの大きさ

        SetScrollInfo( hScrl,  SB_CTL,  &si, FALSE );

        ShowWindow( hMwnd, nCmdShow );                      // SetScrollInfo()の後, 表示する

        //メッセージループ
        while( GetMessage( &msg, NULL, 0, 0 ) ){
                DispatchMessage( &msg );
        }

        return msg.wParam;

}

LRESULT CALLBACK WndProc( HWND hWnd, UINT iMessage,WPARAM wParam, LPARAM lParam )
{
        DWORD tid;
        static HXY hxy;                                     // 外部から参照されるのでstaticにする

        SCROLLINFO si;
        int             i;
        char    str[100];

        switch( iMessage ){
        case WM_CREATE:
                hxy.hw0 = hWnd;
                hxy.xc =  80;                   // くるくるのx座標
                hxy.yc = 80;                    // くるくるのy座標
                hxy.ti = 200;
                hxy.th = CreateThread( NULL, 0, sub, &hxy, 0, &tid);
                break;

        case WM_VSCROLL:
                //  スクロールの区別
                switch( LOWORD( wParam ) ){
                case SB_LINEUP:
                        vpos--;
                        if( vpos < vmin )  vpos = vmin;
                        break;
                case SB_LINEDOWN:
                        vpos++;
                        if( vpos > vmax )  vpos = vmax;
                        break;
                case SB_PAGEUP:
                        vpos -= 5;
                        if( vpos < vmin )  vpos = vmin;
                        break;
                case SB_PAGEDOWN:
                        vpos += 5;
                        if( vpos > vmax )  vpos = vmax;
                        break;
                case SB_THUMBTRACK:
                        vpos = HIWORD( wParam );
                        break;
                case SB_THUMBPOSITION:
                        vpos = HIWORD( wParam );
                        break;
                }
                // パラメータを計算してからSetScrollInfoでつまみを動かす
                si.cbSize = sizeof( SCROLLINFO );
                si.fMask  = SIF_POS;
                si.nPos   = vpos;
                SetScrollInfo( (HWND)lParam, SB_CTL, &si, TRUE );
                hxy.ti = vpos;
                wsprintf( str, "Position  : %3d ", vpos );
                SetWindowText( hWnd, str );

                break;

        case WM_DESTROY:
```

```
                    TerminateThread( hxy.th , 0 );

                    PostQuitMessage( 0 );
                    break;

          default:
                    return DefWindowProc( hWnd, iMessage, wParam, lParam );
          }

          return 0;

}

DWORD WINAPI sub(VOID *phxy)
{
          int n,m,m2;

          int  xx  = ((HXY *)phxy ) ->  xc;
          int  yy  = ((HXY *)phxy ) ->  yc;
          HWND hw0 = ((HXY *)phxy ) -> hw0;

          char cbuf[100];

          HDC  hdc = GetDC( hw0 );
          HPEN hp0 = CreatePen( PS_SOLID, 1, RGB(255,255,255) );          //白ペン
          HPEN hp1 = CreatePen( PS_SOLID, 1, RGB(255,  0,  0) );          //赤ペン
          int x = 20, y = 0, r = 0;
          while( 1 ){
                    SelectObject(hdc,hp0);
                    MoveToEx(hdc,xx + x,yy +y , NULL);
                    LineTo(hdc,          xx - x ,yy -y);
                    r = ( r + 1 ) % 20;
                    SelectObject( hdc, hp1 );
                    x = (int)( 20 * cos( 2 * 3.14 * r/20.0 ) );
                    y = (int)( 20 * sin( 2 * 3.14 * r/20.0 ) );
                    MoveToEx( hdc, xx + x, yy + y , NULL );
                    LineTo( hdc, xx - x, yy - y );

                    n = r % 4;
                    m = 0x8 >> n;
                    m2 = 0x10 << n;
                    DioOutByte(id, 0, ~(m | m2));

                    Sleep(((HXY *)phxy ) -> ti);                                        // スレッドの一時休止
          }

          DeleteObject( hp0 );                                  // 実際は実行されないが一応後処理
          DeleteObject( hp1 );
          ReleaseDC( hw0, hdc );
          return 0;
}
```



3.4　簡易型Windowsからの外部制御

Windows では COM ポートの制御線を個別に制御することが可能です。この制御は USB 経由のＣＯＭポートでも可能です。前述のスイッチ入力を Windows からのＣＯＭポートの制御に置き換えると以下のようなプログラムになります。またパラレルポートも出力ポートとして利用できます。



注：回路を簡略にするため DTR 出力を High レベルにすることによって得られる電圧を使って CD 入力の High レベルを供給します。

```
// list0910_com1.cpp シリアルポートのコントロールラインの制御
//
#include<stdio.h>
#include  <Windows.h>

main()
{
          HANDLE h;

          h = CreateFile( "COM5", 0, 0,                         // 非共有
                              0,                                // セキュリティ属性, 使用しない
                              OPEN_EXISTING,                    // 既存ファイルのオープン
                              0, 0 );                           // 属性, テンプレート

          if( h == INVALID_HANDLE_VALUE ){
                    h = CreateFile( "COM3", 0,                  // COM5がだめならCOM3を開ける
                                        0, 0, OPEN_EXISTING,            // 既存ファイルのオープン
                                        0, 0 );                         // 属性, テンプレート
                    if( h == INVALID_HANDLE_VALUE ){
                              printf( "Open err" );
                              return 0;
                    }
          }

          EscapeCommFunction( h, SETDTR );

          while(1){
                    DWORD x00 ;
                    GetCommModemStatus( h, &x00 );
                    if(       x00 & MS_RLSD_ON )    {
                              printf("CD_ON ¥n"  );
                              EscapeCommFunction( h, SETRTS );
                    }
                    else      {
                              printf("CD_OFF ¥n" );
                              EscapeCommFunction( h, CLRRTS );
                    }
          }

}
```

```
// list0910_com2.cpp シリアルポートのコントロールラインの制御
//
#include<stdio.h>
#include  <Windows.h>

main()
{
          HANDLE h;

          h = CreateFile( "COM5", 0, 0,                         // 非共有
                              0,                                // セキュリティ属性, 使用しない
                              OPEN_EXISTING,                    // 既存ファイルのオープン
                              0, 0 );                           // 属性, テンプレート

          if( h == INVALID_HANDLE_VALUE ){
                    h = CreateFile( "COM3", 0,                  // COM5がだめならCOM3を開ける
                                        0, 0, OPEN_EXISTING,            // 既存ファイルのオープン
                                        0, 0 );                         // 属性, テンプレート
                    if( h == INVALID_HANDLE_VALUE ){
                              printf( "Open err" );
                              return 0;
                    }
          }

          EscapeCommFunction( h, SETDTR );

          int a,b,sw = 0;
          DWORD x00 ;

          GetCommModemStatus( h, &x00 );
          b = x00 & MS_RLSD_ON ;

          while(1){

                    GetCommModemStatus( h, &x00 );
                    a = x00 & MS_RLSD_ON ;
                    if(a != b){                         //変化があった
                              b = a;
                              if(a){                    //プラスのエッジ
                                        if(sw){
                                                  sw = 0;
                                                  printf("CD_OFF ¥n" );
                                                  EscapeCommFunction( h, CLRRTS );
                                        }
                                        else  {
                                                  sw = 1;
                                                  printf("CD_ON ¥n"  );
                                                  EscapeCommFunction( h, SETRTS );
                                        }
                              }
                    }
          }
}
```