Windows カスタムライブラリ

ライブラリ説明書

C++ クラスライブラリ編

（ 1.6.5.6 版 ）

－ 目次 －

1. 概　要

本ライブラリは、ライブラリ本体（AjrCst32.dll/AjrCst64.dll）を、ラッパークラス化したものです。
ライブラリ本体を全てラップしたわけではありませんので、ラッパークラス化されていないものに関しては、ライブラリ本体のＡＰＩを直接呼び出す必要があります。
（主に、イベント通知やコールバックを含む機能についてラッパークラス化して、イベント／コールバックを仮想関数化しています）

本クラスライブラリは、STL や MFC とは独立して、純粋に C++のクラスとして存在します。

各クラスのメンバ関数や定義シンボルは（接頭語部分(Ajc‥）を除いて）ライブラリ本体とほとんど同じです。
従って、関数の機能や引数については、「AjrCst32.pdf(C/C++ 用レギュラーDLL(AjrCst32.dll / AjrCst64.dll)編)」を参照してください。
（例えば、CAjxTch クラスの Stop()メンバ関数の場合、AjrCst32.pdf 中の「AjcTchStop()」を参照します。）
本書では、C++ クラスのメンバ変数／メンバ関数の一覧だけを示します。

本ライブラリはソースプログラム（VisualStudio プロジェクト）を同梱して配布していますので、ユーザ自身でカスタマイズ／修正を行い、ライブラリを再ビルドすることができます。(詳細は「AjrCstBuildPack.pdf」を参照してください)

1.1. ファイル構成

本ライブラリは、以下のファイルで構成されます。

INCLUDE

#	ファイル名	内容
1	AjxCpp.h	代表インクルードファイル
2	Ajx*.h	その他のインクルードファイル（AjxCpp.h 内でインクルードされます）

LIB

#	ファイル名	内容
1	AjxCpp32.lib	C++ ラッパークラス ライブラリ(32bit 用)
2	AjxCpp64.lib	C++ ラッパークラス ライブラリ(64bit 用)
3	AjrCst32.lib	ライブラリ処理本体（32 bit 用）
4	AjrCst64.lib	ライブラリ処理本体（64 bit 用）

DLL

#	ファイル名	内容
1	AjxCpp32.dll	C++ ラッパークラス ライブラリ(32bit 用)
2	AjxCpp64.dll	C++ ラッパークラス ライブラリ(64bit 用)
3	AjrCst32.dll	ライブラリ処理本体（32 bit 用）
4	AjrCst64.dll	ライブラリ処理本体（64 bit 用）

1.2. 実行可能な Windows／VisualStudio バージョン

本ライブラリは、Windows 10, Windows11 における 32 ビット・プラットフォーム(x86)と、64 ビット・プラットフォーム(x64)で動作可能です。
VisualStudio は、VisualStudio2010 以降でビルド可能です。

1.3. マルチバイト文字とワイド文字(UNICODE)

基本的にマルチバイト文字用の関数と、ワイド文字(UNICODE)用の関数は、関数のオーバーロードで対応します。

ex.

```
BOOL    DlgItemLoadTextBox (UI id, C_BCP pSect, BOOL fExcRdOnly = TRUE);
BOOL    DlgItemLoadTextBox (UI id, C_WCP pSect, BOOL fExcRdOnly = TRUE);
```

※「pSect」はバイト文字(“XXXXX”)とワイド文字(L”XXXXX”)のいずれでも指定可能

1.4. イベント通知／コールバック

ウインドメッセージによるイベント通知や、コールバックは仮想関数／純粋仮想関数として実装します。

1.5. バージョン更新時の注意事項

本ライブラリのバージョンを更新した場合は、ライブラリを使用しているプログラムを全て再ビルドしてください。
プログラムを配布する場合は、プログラムをビルドしたバージョンのＤＬＬを同梱してください。

1.6. サンプルプログラム

本書のサンプルプログラムは、STL や MFC は使用せずに、純粋な（ネイティブな）C++プログラムとしています。
尚、MFC を使用したサンプルプログラムは、「S_MFC_02」(AjrCst32.pdf)を参照してください。

2. クラス定義

2.1. タイムチャート・グラフ表示コントロール（ CAjxTch ）

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxTch();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	BOOL Stop ();	停止
2	BOOL Start();	開始
3	BOOL Purge ();	クリアー
4	BOOL PutData (double dat[]); BOOL PutData (double d0); BOOL PutData (double d0, double d1); BOOL PutData (double d0, double d1, double d2);	実数データ投与
5	BOOL PutData (int dat[]); BOOL PutData (int d0); BOOL PutData (int d0, int d1); BOOL PutData (int d0, int d1, int d2);	整数データ投与
6	BOOL ShowBorder (BOOL fShow, COLORREF rgb);	外枠表示
7	BOOL ShowFilter (BOOL fShow);	フィルタ表示
8	BOOL ShowScale (BOOL fLine, BOOL fValue);	スケール表示
9	BOOL GetProp (PAJCTCPROP pBuf);	プロパティ取得
10	BOOL SetProp (PCAJCTCPROP pProp);	プロパティ設定
11	BOOL GetRange (double *low, double *high);	実数レンジ取得
12	BOOL GetRange (int *low, int *high);	整数レンジ取得
13	BOOL SetRange (double low, double high);	実数レンジ設定
14	BOOL SetRange (int low, int high);	整数レンジ設定
15	BOOL AdjustRange ();	レンジ自動調整
16	BOOL SetBufSize (int n);	バッファサイズ設定
17	BOOL SetItemNumber (int n);	データ項目数設定
18	BOOL SetAveNumber (int n);	平均化個数設定
19	BOOL SetTimeScale (int n);	タイムスケール幅設定
20	HBITMAP GetBitmap ();	ビットマップ取得
21	BOOL LoadPermInfo (C_BCP pProfileSect = L"TchPerm_*", C_BCP pKeyPrefix = L"Tch", UI PermItem = AJCTCH_PERM_FILTER);	プロファイルから永続情報読み出し (ASCII)
22	BOOL LoadPermInfo (C_WCP pProfileSect = L"TchPerm_*", C_WCP pKeyPrefix = L"Tch", UI PermItem = AJCTCH_PERM_FILTER);	プロファイルから永続情報読み出し (UNICODE)
23	BOOL SavePermInfo ();	プロファイルへ永続情報書き込み
24	BOOL LoadProp (C_BCP pProfileSect = "TchProp", PCAJCTCPROP pDefProp = NULL);	プロファイルからプロパティ読み出し(ASCII)
25	BOOL LoadProp (C_WCP pProfileSect = L"TchProp", PCAJCTCPROP pDefProp = NULL);	プロファイルからプロパティ読み出し(UNICODE)
26	BOOL SaveProp (C_BCP pProfileSect = "TchProp");	プロファイルへプロパティ書き込み(ASCII)
27	BOOL SaveProp (C_WCP pProfileSect = "TchProp");	プロファイルへプロパティ書き込み(UNICODE)
28	BOOL LoadPropEx (C_BCP pProfileSect = "TchProp", PCAJCTCPROP pDefProp = NULL);	プロファイルからプロパティ，フィルタ設定，ウインドスタイル読み出し (ASCII)
29	BOOL LoadPropEx (C_WCP pProfileSect = L"TchProp", PCAJCTCPROP pDefProp = NULL);	プロファイルからプロパティ，フィルタ設定，ウインドスタイル読み出し (UNICODE)

#	変数／関数形式	内容
30	BOOL EnablePopupMenu (BOOL fEnable);	右クリックによるポップアップメニューの許可／禁止
31	BOOL SetNtcRClk (BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);	右クリック通知設定
32	BOOL SetTipText (C_BCP pTxt);	ツールチップ・テキスト設定(ASCII)
33	BOOL SetTipText (C_WCP pTxt);	ツールチップ・テキスト設定(UNICODE)
34	BOOL GetTipText (BCP pBuf, UI lBuf);	ツールチップ・テキスト取得(ASCII)
35	BOOL GetTipText (WCP pBuf, UI lBuf);	ツールチップ・テキスト取得(UNICODE)
36	BOOL SetTipShowAlways(BOOL fShowAlways);	ツールチップ・テキスト表示条件設定
37	BOOL SetFontInfo(const LOGFONTA *pLogFont); BOOL SetFontInfo(const LOGFONTA *pLogFont, UI LSpace);	フォント情報の設定
38	UI GetFontInfo(LPLOGFONTA *pBuf);	フォント情報の取得 (戻り値＝行間スペース)
39	BOOL GetTipShowAlways();	ツールチップ・テキスト表示条件取得
40	BOOL SetChkBoxTipShowAlways (UI n, BOOL fShowAlways);	フィィルタ (チェックボックス) のツールチップ・テキスト表示条件設定
41	BOOL GetChkBoxTipShowAlways (UI n);	フィィルタ (チェックボックス) のツールチップ・テキスト表示条件取得
42	BOOL SetTipShowAlwaysAll (BOOL fShowAlways);	全てのツールチップ・テキスト表示条件設定
43	BOOL SetChkBoxTipText (UI n, C_BCP pTxt);	フィィルタ (チェックボックス) のツールチップ・テキスト設定(ASCII)
44	BOOL SetChkBoxTipText (UI n, C_WCP pTxt);	フィィルタ (チェックボックス) のツールチップ・テキスト設定(UNICODE)
45	BOOL GetChkBoxTipText (UI n, BCP pBuf, UI lBuf);	フィィルタ (チェックボックス) のツールチップ・テキスト取得(ASCII)
46	BOOL GetChkBoxTipText (UI n, WCP pBuf, UI lBuf);	フィィルタ (チェックボックス) のツールチップ・テキスト取得(UNICODE)
47	BOOL SetMaxLineDist (double MaxLineDist);	最大結線長設定
48	double GetMaxLineDist ();	最大結線長取得
49	int GetScrollPos ();	スクロール位置の取得
50	BOOL SetScrollPos (int pos);	スクロール位置の設定
51	BOOL SetFilter (UI n, BOOL state);	フィルタの設定
52	BOOL GetFilter (UI n);	フィルタの取得
53	BOOL SetHLineAtt (UI id, COLORREF color, int width = 1, int style = AJCTCH_DASH);	横線の属性設定
54	BOOL SetHLinePos (UI id, double pos);	横線の描画位置設定
55	BOOL EnableHLine (UI id, BOOL fEnable);	横線描画の許可／禁止
56	BOOL SetVLine (COLORREF color, int width = 1, int style = AJCTCH_DASH);	縦線描画 (最後に投与したデータの位置に縦線を描画する)
57	BOOL EnableVLine (BOOL fEnable);	縦線描画の許可／禁止
58	BOOL SetPoint (int r);	プロット点描画 (最後に格納したデータの位置にプロット点 (円) を描画する)
59	BOOL EnablePoint (BOOL fEnable);	プロット点描画の許可／禁止
60	BOOL GetDroppedFile (BC buf[MAX_PATH]);	ドロップされたファイル名取得(ASCII)
61	BOOL GetDroppedFile (WC buf[MAX_PATH]);	ドロップされたファイル名取得(UNICODE)
62	BOOL GetDroppedDir (BC buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得(末尾の「¥」付加指定付き)(ASCII)
63	BOOL GetDroppedDir (WC buf[MAX_PATH], BOOL fTailIsDelimiter);	ドロップされたディレクトリ名取得(末尾の「¥」付加指定付き)(UNICODE)
64	BOOL SetTitleText (C_BCP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定(ASCII)
65	BOOL SetTitleText (C_WCP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定(UNICODE)
66	BOOL SetIpInfo (PCAJCTCIPINFO pInfo);	波形補間表示情報の設定
67	BOOL GetIpInfo (PAJCTCIPINFO pBuf);	波形補間表示情報の取得
68	BOOL Pause (BOOL fPause);	表示の停止／再開
69	BOOL EnableMesDraw (BOOL fEnable);	描画時間計測情報の許可／禁止
70	BOOL MesPeriShow (BOOL fShow);	プロット周期計測情報表示／非表示の設定
71	BOOL MesPeriReset ();	プロット周期計測情報リセット
72	HFONT SetTextFont (HFONT hFont);	テキスト描画フォント設定
73	UI TextOut (int x, int y, C_BCP pTxt);	テキスト描画 (ピクセル位置指定) (ASCII)
74	UI TextOut (int x, int y, C_WCP pTxt);	テキスト描画 (ピクセル位置指定) (UNICODE)
75	UI PrintF (int x, int y, C_BCP pFmt, ...);	書式テキスト描画 (ピクセル位置指定) (ASCII)
76	UI PrintF (int x, int y, C_WCP pFmt, ...);	書式テキスト描画 (ピクセル位置指定) (UNICODE)
77	UI GetText (UI key, BCP pBuf, UI lBuf);	描画テキスト取得(ASCII)
78	UI GetText (UI key, WCP pBuf, UI lBuf);	描画テキスト取得(UNICODE)
79	BOOL ClearText (UI key);	描画テキストクリアー
80	BOOL ClearText ();	全ての描画テキストクリアー
81	BOOL Clear ();	全てのデータと描画テキストクリアー

仮想関数

#	変数／関数形式	内容	
1	virtual VO Attach (HWND hWnd);	ハンドルを関連付け (override 時、継承元 Attach()実行要) ※1	
2	virtual VO OnNtcRange (PCAJCTC_NTC_RANGE pRange);	レンジ変化通知	AJCTCN_RANGE
3	virtual VO OnNtcScrPos (UI ScrPos);	スクロール位置変化通知	AJCTCN_SCRPOS
4	virtual VO OnNtcClear ();	グラフクリアー通知	AJCTCN_CLEAR
5	virtual VO OnNtcDblClk ();	ダブルクリック通知	AJCTCN_DBLCLK
6	virtual VO OnNtcDropFile (UI nFiles);	ファイルドロップ通知	AJCTCN_DROPFILE
7	virtual VO OnNtcDropDir (UI nDirs);	ディレクトリドロップ通知	AJCTCN_DROPDIR
8	virtual VO OnNtcRClick (PCAJCTCRCLK pRClk);	右クリック通知	AJCTCN_RCLICK

※ AJCTCN_*XXXXX*は対応する通知コードを示します。

※1：タイムチャートウインドを生成したら、Attach()を実行して、タイムチャートウインドのハンドルを関連付けてください。
例えば、ダイアログボックスにタイムチャートウインド（ *IDC_TCH* ）を配置した場合、以下のようにします。

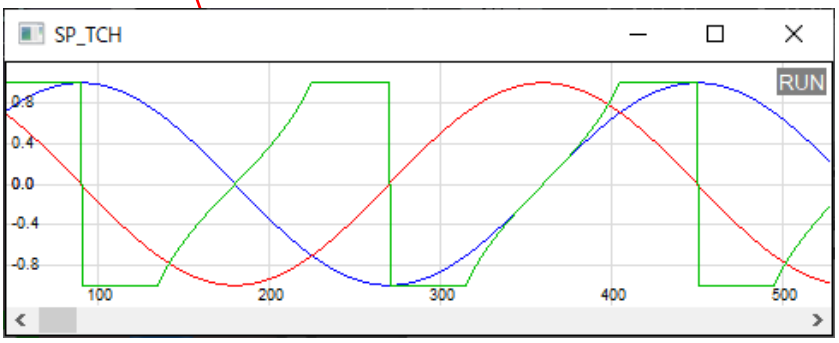
```
CAjxTch    m_tch;
   ・・・
m_tch.Attach(GetDlgItem(IDC_TCH)->m_hWnd);    //  MFC の CDialog クラスの場合
m_tch.Attach(::GetDlgItem(hDlg, IDC_TCH);     //  Windows API をコールする場合
```

また、CAjxTch クラスを継承し、Attach()を override する場合は、継承元の Attach()を呼び出す必要があります。

```
class CAjxTchEx :
    public CAjxTch
{
public:
    ・・・
    VO  Attach (HWND hwnd)  override
    {
        ・・・・
        CAjxTch::Attach(hwnd);  // 基底クラスの Attach()呼び出し
    }
    ・・・
};
```

使用例

Sin, Cos, Tan の波形を表示します (Tan は± 1 でカット)




```
//
//  SP_TCH.cpp
//
#include    <AjxCpp.h>
#include    <tchar.h>
using namespace AjxControl;

#define IDC_TCH 5001
static  HINSTANCE   hInst    = NULL;
static  HWND        hWndMain = NULL;
static  HWND        hWndTch  = NULL;
static  BOOL        fTimer   = FALSE;

AJC_WNDPROC_DEF(Main);

static const UT TipMsg[] =  TEXT("三角関数の値をグラフ表示します。( Tan は±1 でカット)¥n")
                            TEXT("ダブルクリックで、グラフ表示を開始／停止します。¥n")
                            TEXT("このツールチップ上にカーソルを置くと表示を継続します。¥n")
                            TEXT("このツールチップ上をクリックするとツールチップは消えます。");

//----- CAjxTch の派生クラス ----------------------------------------------------------------------------------//
class CAjxTchEx :
    public CAjxTch,
    public CAjxTip
{
public:
    VO  SetHWnd(HWND hwnd)
    {
        CAjxTch::Attach(hwnd);
        CAjxTip::Attach(hwnd);
    }
    VO  OnNtcDblClk () override {                           //  ダブルクリック通知
        if (fTimer) {SetTitleText(TEXT("STOP")); KillTimer(hWndMain, 1);}
        else        {SetTitleText(TEXT("RUN" )); SetTimer (hWndMain, 1, 10, NULL);}
        fTimer = ~fTimer;
    }
    VO  OnNeedText (HWND hCtrl, UTP pBuf, UI lBuf) override {   //  チップテキスト取得要求
        if (fTimer) _tcscpy_s(pBuf, lBuf, TEXT("グラフ表示中です。ダブルクリックで表示を停止します。"));
        else        _tcscpy_s(pBuf, lBuf, TEXT("グラフ停止中です。ダブルクリックで表示を開始します。"));
    }
};
static  CAjxTchEx   tch;

//----- WinMain -----------------------------------------------------------------------------------------------//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG         msg = {0};
    WNDCLASS    wc  = {0};
    ATOM        atm = 0;

    hInst = hInstance;
    //  ウインド生成
    wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
    wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_TCH");
    atm = RegisterClass(&wc);
    hWndMain = CreateWindow(TEXT("SP_TCH"), TEXT("SP_TCH"),             // window class name, caption
                            WS_OVERLAPPEDWINDOW,                        // window style
                            CW_USEDEFAULT, CW_USEDEFAULT, 500, 200,     // position, size
                            NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
    ShowWindow(hWndMain, iCmdShow);
    //  メッセージループ
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage (&msg);
    }
    UnregisterClass((UTP)atm, hInstance);
    return (int)msg.wParam ;
}
//----- WM_CREATE ---------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_CREATE     )
```

```
 72 : {
 73 :     //  タイムチャートウインド生成
 74 :     RECT    r;
 75 :     GetClientRect(hwnd, &r);
 76 :     GetClientRect(hwnd, &r);
 77 :     hWndTch = CreateWindow(TEXT("AjcCtrlTmChart"),                  // window class
 78 :                            TEXT("P: L=-1.2, H=1.2, B=10000"),       // window caption
 79 :                            WS_CHILD,                                // window style
 80 :                            0, 0, r.right, r.bottom,                 // position, size
 81 :                            hwnd, (HMENU)IDC_TCH, hInst, NULL);      // parent, menu, instance, param
 82 :
 83 :     SAjxTip::ShowCenter(hWndTch, TipMsg);   //  チップテキスト初期表示
 84 :     tch.SetHWnd(hWndTch);                   //  タイムチャートウインドハンドル割り当て
 85 :     tch.SetTitleText(TEXT("STOP"));         //  タイムチャートウインドタイトル設定
 86 :     ShowWindow(hWndTch, SW_SHOW);           //  タイムチャートウインド表示
 87 :     return 0;
 88 : }
 89 : //----- WM_DESTROY ----------------------------------------------------------------------------------------//
 90 : AJC_WNDPROC(Main, WM_DESTROY    )
 91 : {
 92 :     DestroyWindow(hWndTch);
 93 :     PostQuitMessage(0);
 94 :     return 0;
 95 : }
 96 : //----- WM_SIZE -------------------------------------------------------------------------------------------//
 97 : AJC_WNDPROC(Main, WM_SIZE       )
 98 : {
 99 :     MoveWindow(hWndTch, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
100 :     return 0;
101 : }
102 : //----- WM_TIMER ------------------------------------------------------------------------------------------//
103 : AJC_WNDPROC(Main, WM_TIMER      )
104 : {
105 :     static double x = 0.0;
106 :     double  s = sin(x);     double  c = cos(x);     double  t = tan(x);
107 :     t = __min(t, 1.0); t = __max(t, -1.0);
108 :     tch.PutData(s, c, t);
109 :     x += (3.14159 / 180.0); x = fmod(x, 3.14159 * 2);
110 :     return 0;
111 : }
112 : //---------------------------------------------------------------------------------------------------------//
113 : AJC_WNDMAP_DEF(Main)
114 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
115 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
116 :     AJC_WNDMAP_MSG(Main, WM_SIZE    )
117 :     AJC_WNDMAP_MSG(Main, WM_TIMER   )
118 : AJC_WNDMAP_END
```

2.2. ２Ｄグラフィック・コントロール（ CAjxG2d ）

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxG2d();	コンストラクタ

メンバ関数（２Ｄファンクション）

#	変数／関数形式	内容
1	BOOL G2Init (double x1, double y1, double x2, double y2, UI style = AJC3DGS_2DMODE);	２Ｄグラフモード初期化
2	BOOL G2Init (PCAJC2DVEC pLo, PCAJC2DVEC pHi, UI style = AJC3DGS_2DMODE);	２Ｄグラフモード初期化（ベクトル指定）
3	BOOL G2SetPlane (AJCG2DAXIS_DIR HoriAxis, AJCG2DAXIS_DIR VertAxis);	平面のアングル設定
4	BOOL G2SetProp (PCAJC2DGPROP pProp);	プロパティ設定
5	BOOL G2GetProp (PAJC2DGPROP pBuf);	プロパティ取得
6	BOOL G2AdjustRange ();	レンジ自動調整
7	BOOL G2SetRange (double x1, double y1, double x2, double y2);	各軸のレンジ設定
8	BOOL G2SetCenter (double xc, double yc);	各軸の中心位置設定
9	BOOL G2SetWidth (double xr, double yr);	各軸の幅（半径）設定
10	BOOL SetSameRangeWidth ();	軸のレンジ幅を同じにする
11	BOOL SetFixedAspect (BOOL fFixed);	アスペクトを１にする
12	BOOL SetColor (UI id, COLORREF rgb);	表示色設定
13	BOOL SetPlotNumber (UI id, UI PlotNumber);	プロットデータ数設定
14	BOOL SetPlotSize (UI id, UI PixelSize, UI PixelSizeE);	プロットデータのピクセルサイズ設定
15	BOOL PutPlotData (UI id, double x, double y);	プロットデータ投与
16	BOOL PutPlotData (UI id, PCAJC2DVEC pPoint);	プロットデータ投与（ベクトル指定）
17	HBITMAP GetBitmap ();	ビットマップデータ取得
18	BOOL LoadProp (C_BCP pProfileSect = “G2dProp”, PCAJC2DGPROP pDefProp = NULL);	プロファイルからプロパティ値読出し(ASCII)
19	BOOL LoadProp (C_WCP pProfileSect = L“G2dProp”, PCAJC2DGPROP pDefProp = NULL);	プロファイルからプロパティ値読出し(UNICODE)
20	BOOL SaveProp (C_BCP pProfileSect = “G2dProp”);	プロファイルへプロパティ値書き込み(ASCII)
21	BOOL SaveProp (C_WCP pProfileSect = L“G2dProp”);	プロファイルへプロパティ値書き込み(UNICODE)
22	BOOL LoadPropEx (C_BCP pProfileSect = “G2dProp”, PCAJC2DGPROP pDefProp = NULL);	プロファイルから詳細プロパティ値読出し(ASCII)
23	BOOL LoadPropEx (C_WCP pProfileSect = L“G2dProp”, PCAJC2DGPROP pDefProp = NULL);	プロファイルから詳細プロパティ値読出し(UNICODE)
24	BOOL SavePropEx (C_BCP pProfileSect = “G2dProp”);	プロファイルへ詳細プロパティ値書き込み(ASCII)
25	BOOL SavePropEx (C_WCP pProfileSect = L“G2dProp”);	プロファイルへ詳細プロパティ値書き込み(UNICODE)
26	BOOL FillB (UI id, UI idBorder, double x, double y);	ボーダー色で囲まれた部分の塗りつぶし
27	BOOL FillB (UI id, UI idBorder, PCAJC2DVEC pPoint);	ボーダー色で囲まれた部分の塗りつぶし（ベクトル指定）
28	BOOL FillS (UI id, double x, double y);	連続する白色部分の塗りつぶし
29	BOOL FillS (UI id, PCAJC2DVEC pPoint);	連続する白色部分の塗りつぶし（ベクトル指定）
30	COLORREF GetPixel (double x, double y);	ピクセルの表示色取得
31	COLORREF GetPixel (PCAJC2DVEC pPoint);	ピクセルの表示色取得（ベクトル指定）
32	HFONT SetTextFont (HFONT hFont);	テキスト描画フォント設定
33	UI TextOut (int x, int y, C_BCP pTxt);	テキスト描画（ピクセル位置指定）(ASCII)
34	UI TextOut (int x, int y, C_WCP pTxt);	テキスト描画（ピクセル位置指定）(UNICODE)
35	UI PrintF (int x, int y, C_BCP pFmt, ...);	書式テキスト描画（ピクセル位置指定）(ASCII)
36	UI PrintF (int x, int y, C_WCP pFmt, ...);	書式テキスト描画（ピクセル位置指定）(UNICODE)
37	UI TextOutV (AJCG2DTXOMD md, PCAJC2DVEC pV, C_BCP pTxt);	テキスト描画（２Ｄ座標指定）(ASCII)
38	UI TextOutV (AJCG2DTXOMD md, PCAJC2DVEC pV, C_WCP pTxt);	テキスト描画（２Ｄ座標指定）(UNICODE)

メンバ関数（２Ｄファンクション）

#	変数／関数形式	内容
39	UI PrintFV (AJCG2DTXOMD md, PCAJC2DVEC pV, C_BCP pFmt, ...);	書式テキスト描画（２Ｄ座標指定）(ASCII)
40	UI PrintFV (AJCG2DTXOMD md, PCAJC2DVEC pV, C_WCP pFmt, ...);	書式テキスト描画（２Ｄ座標指定）(UNICODE)
41	UI GetText (UI key, BCP pBuf, UI lBuf);	描画テキスト取得(ASCII)
42	UI GetText (UI key, WCP pBuf, UI lBuf);	描画テキスト取得(UNICODE)
43	BOOL ClearShape (UI id);	図形描画データクリアー
44	BOOL ClearShape ();	全ての図形描画データクリアー
45	BOOL ClearPlot (UI id);	プロットデータクリアー
46	BOOL ClearPlot ();	全てのプロットデータクリアー
47	BOOL ClearText (UI id);	描画テキストクリアー
48	BOOL ClearText ();	全ての描画テキストクリアー

メンバ関数（平面描画ファンクション）

#	変数／関数形式	内容
1	BOOL Pixel (UI id, double x, double y, UI PixelSize);	平面にピクセル描画
2	BOOL Pixel (UI id, PCAJC2DVEC pPoint, UI PixelSize);	平面にピクセル描画（ベクトル指定）
3	BOOL Line (UI id, double x1, double y1, double x2, double y2);	平面にライン描画
4	BOOL Line (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);	平面にライン描画（ベクトル指定）
5	BOOL MoveTo (UI id, double x, double y);	平面に描画するライン／矢印の始点設定
6	BOOL MoveTo (UI id, PCAJC2DVEC ps);	平面に描画するライン／矢印の始点設定（ベクトル指定）
7	BOOL LineTo (UI id, double x, double y);	終点を指定し平面にライン描画
8	BOOL LineTo (UI id, PCAJC2DVEC pe);	終点を指定し平面にライン描画（ベクトル指定）
9	BOOL Arrow (UI id, double x1, double y1, double x2, double y2);	平面に矢印描画
10	BOOL Arrow (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);	平面に矢印描画（ベクトル指定）
11	BOOL ArrowTo (UI id, double x, double y);	終点を指定し平面に矢印描画
12	BOOL ArrowTo (UI id, PCAJC2DVEC pe);	終点を指定し平面に矢印描画（ベクトル指定）
13	BOOL Triangle (UI id, double x1, double y1, double x2, double y2, double x3, double y3);	平面に三角形描画
14	BOOL Triangle (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2, PCAJC2DVEC p3);	平面に三角形描画（ベクトル指定）
15	BOOL Square (UI id, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4);	平面に四角形描画
16	BOOL Square (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2, PCAJC2DVEC p3, PCAJC2DVEC p4);	平面に四角形描画（ベクトル指定）
17	BOOL Rectangle (UI id, double x1, double y1, double x2, double y2);	平面に長方形描画
18	BOOL Rectangle (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);	平面に長方形描画（ベクトル指定）
19	BOOL Ellipse (UI id, double xc, double yc, double rx, double ry);	平面に円／楕円描画
20	BOOL Ellipse (UI id, PCAJC2DVEC pCent, double rx, double ry);	平面に円／楕円描画（ベクトル指定）
21	BOOL Star (UI id, double xc, double yc, double r1, double r2 = 0.0, UI nVertex = 5, double rot = 0.0, BOOL fInscribedLine = FALSE);	平面に星形描画
22	BOOL Star (UI id, PCAJC2DVEC pCent , double r1, double r2 = 0.0, UI nVertex = 5, double rot = 0.0, BOOL fInscribedLine = FALSE);	平面に星形描画（ベクトル指定）

メンバ関数（その他のファンクション）

#	変数／関数形式	内容
23	BOOL EnablePopupMenu (BOOL fEnable);	右クリックによるポップアップメニューの許可／禁止
24	BOOL SetNtcRClk (BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);	右クリック通知設定
25	BOOL SetTipText (C_BCP pTxt);	ツールチップの設定(ASCII)
26	BOOL SetTipText (C_WCP pTxt);	ツールチップの設定(UNICODE)
27	UI GetTipText (BCP pBuf, UI lBuf);	ツールチップの取得(ASCII)
28	UI GetTipText (WCP pBuf, UI lBuf);	ツールチップの取得(UNICODE)
29	BOOL SetChkBoxTipText (UI n, C_BCP pTxt);	フィルタ・チェックボックス・ツールチップの設定(ASCII)
30	BOOL SetChkBoxTipText (UI n, C_WCP pTxt);	フィルタ・チェックボックス・ツールチップの設定(UNICODE)
31	UI GetChkBoxTipText (UI n, BCP pBuf, UI lBuf);	フィルタ・チェックボックス・ツールチップの取得(ASCII)
32	UI GetChkBoxTipText (UI n, WCP pBuf, UI lBuf);	フィルタ・チェックボックス・ツールチップの取得(UNICODE)
33	BOOL SetTipShowAlways (BOOL fShowAlways);	ツールチップ表示条件の設定
34	BOOL GetTipShowAlways ();	ツールチップ表示条件の取得
35	BOOL SetChkBoxTipShowAlways(UI n, BOOL fShowAlways);	フィルタ・チェックボックス・ツールチップの設定
36	BOOL GetChkBoxTipShowAlways(UI n);	フィルタ・チェックボックス・ツールチップの取得
37	BOOL SetTipShowAlwaysAll (BOOL fShowAlways);	全ツールチップ表示条件の設定
38	BOOL SetFilter (UI n, BOOL state);	フィルタの設定
39	BOOL GetFilter (UI n);	フィルタの取得
40	BOOL GetDroppedFile (BC buf[MAX_PATH]);	ドロップされたファイル名取得(ASCII)
41	BOOL GetDroppedFile (WC buf[MAX_PATH]);	ドロップされたファイル名取得(UNICODE)
42	BOOL GetDroppedDir (BC buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得(ASCII)
43	BOOL GetDroppedDir (WC buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得(UNICODE)
44	BOOL SetTitleText (C_BCP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定(ASCII)
45	BOOL SetTitleText (C_WCP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定(UNICODE)
46	BOOL Pause (BOOL fPause);	画面表示の停止／再開
47	BOOL EnableMesDraw (BOOL fEnable);	描画時間計測情報の許可／禁止

仮想関数

#	変数／関数形式	内容	
1	virtual VO Attach (HWND hWnd);	ハンドルを関連付け（override 時、継承元 Attach()実行要） ※１	
2	virtual VO OnNtcPlotList(PCAJC3DGPLOTLIST pList);	プロットリスト通知	AJC3DGN_PLOTLIST
3	virtual VO OnNtcClear (int Factor);	データクリアー通知	AJC3DGN_CLEAR
4	virtual VO OnNtcDblClk ();	ダブルクリック通知	AJC3DGN_DBLCLK
5	virtual VO OnNtcDropFile(UI n);	ファイルドロップ通知	AJC3DGN_DROPFILE
6	virtual VO OnNtcDropDir (UI n);	フォルダドロップ通知	AJC3DGN_DROPDIR
7	virtual VO OnNtcRClick (PCAJC3DGRCLK pRClk);	右クリック通知	AJC3DGN_RCLICK

※ AJC3DGN_*XXXXX*は、対応する通知コードを意味します。

※１：２Ｄグラフィックウインドを生成したら、Attach()を実行して、２Ｄグラフィックウインドのハンドルを関連付けてください。
例えば、ダイアログボックスに２Ｄグラフィックウインド（ *IDC_G2D* ）を配置した場合、以下のようにします。

```
CAjxG2d    m_G2d;
   ・・・
m_G2d.Attach(GetDlgItem(IDC_G2D)->m_hWnd);    //  MFC の CDialog クラスの場合
m_G2d.Attach(::GetDlgItem(hDlg, IDC_G2D);     //  Windows API をコールする場合
```

また、CAjxG2d クラスを継承し、Attach()を override する場合は、継承元の Attach()を呼び出す必要があります。

```
class CAjxG2dEx :
    public CAjxG2d
{
public:
    ・・・
    VO  Attach (HWND hwnd)  override
    {
        ・・・・
        CAjxG2d::Attach(hwnd);  // 基底クラスの Attach()呼び出し
    }
    ・・・
};
```

2.3. ３Ｄグラフィック・コントロール（ CAjxG3d ）

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxG3d();	コンストラクタ

メンバ関数（３Ｄファンクション）

#	変数／関数形式	内容
1	BOOL Init (double x1, double y1, double z1, double x2, double y2, double z2, UI style = AJC3DGS_3DMODE);	３Ｄグラフモード初期化
2	BOOL Init (PCAJC3DVEC pLo, PAJC3DVEC pHi, PAJC3DVEC pRot, UI style = AJC3DGS_3DMODE);	３Ｄグラフモード初期化（ベクトル指定）
3	BOOL SetPlane(AJCG3DAXIS_DIR HoriAxis, AJCG3DAXIS_DIR VertAxis);	平面表示
4	BOOL SetProp (PCAJC3DGPROP pProp);	プロパティ設定
5	BOOL GetProp (PAJC3DGPROP pBuf);	プロパティ取得
6	BOOL AdjustRange ();	レンジ自動調整
7	BOOL SetRange (double x1, double y1, double z1, double x2, double y2, double z2);	各軸のレンジ設定
8	BOOL SetCenter (double xc, double yc, double zc);	各軸の中心位置設定
9	BOOL SetWidth (double xr, double yr, double zr);	各軸の幅（半径）設定
10	BOOL SetSameRangeWidth ();	各軸のレンジ幅を同じにする
11	BOOL SetFixedAspect (BOOL fFixed);	アスペクトを１固定に設定する
12	BOOL SetColor (UI id, COLORREF rgbP, COLORREF rgbN);	表示色設定
13	BOOL SetPlotNumber (UI id, UI PlotNumber);	プロットデータ数設定
14	BOOL SetPlotSize (UI id, UI PixelSize, UI PixelSizeE);	プロットデータのピクセルサイズ設定
15	BOOL PutPlotData (UI id, double x, double y, double z);	プロットデータ投与
16	BOOL PutPlotData (UI id, PCAJC3DVEC pPoint);	プロットデータ投与（ベクトル指定）
17	BOOL Pixel (UI id, double x , double y , double z , UI PixelSize = 2);	ピクセル描画
18	BOOL Pixel (UI id, PCAJC3DVEC pPoint, UI PixelSize);	ピクセル描画（ベクトル指定）
19	BOOL Line (UI id, double x1, double y1, double z1, double x2, double y2, double z2);	ライン描画
20	BOOL Line (UI id, PCAJC3DVEC p1, PCAJC3DVEC p2);	ライン描画（ベクトル指定）
21	BOOL MoveTo (UI id, double x, double y, double z);	ライン／矢印の始点設定
22	BOOL MoveTo (UI id, PCAJC3DVEC ps);	ライン／矢印の始点設定（ベクトル指定）
23	BOOL LineTo (UI id, double x, double y, double z);	ラインの終点を設定しライン描画
24	BOOL LineTo (UI id, PCAJC3DVEC pe);	ラインの終点を設定しライン描画（ベクトル指定）
25	BOOL Arrow (UI id, double x1, double y1, double z1, double x2, double y2, double z2);	矢印描画
26	BOOL Arrow (UI id, PCAJC3DVEC p1, PCAJC3DVEC p2);	矢印描画（ベクトル指定）
27	BOOL ArrowTo (UI id, double x, double y, double z);	ラインの終点を設定し矢印描画
28	BOOL ArrowTo (UI id, PCAJC3DVEC pe);	ラインの終点を設定し矢印描画（ベクトル指定）
29	BOOL Triangle (UI id, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3);	三角形描画
31	BOOL Triangle (UI id, PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3);	三角形描画（ベクトル指定）
32	BOOL Square (UI id, double x1, double y1, double z1, double x2, double y2, double z2, double x3, double y3, double z3, double x4, double y4, double z4);	四角形描画

メンバ関数（３Ｄファンクション）

#	変数／関数形式	内容
33	BOOL Square (UI id, PCAJC3DVEC p1, PCAJC3DVEC p2, PCAJC3DVEC p3, PCAJC3DVEC p4);	四角形描画（ベクトル指定）
34	BOOL Cube (UI id, double xc, double yc, double zc, double xr, double yr, double zr, UI division = 8);	立方体／長方体描画
35	BOOL Cube (UI id, PCAJC3DVEC pCent, double xr, double yr, double zr, UI division = 8);	立方体／長方体描画（ベクトル指定）
36	BOOL Sphere (UI id, double xc, double yc, double zc, double xr, double yr, double zr, UI slice = 12, UI stack = 12);	球／楕球描画
37	BOOL Sphere (UI id, PCAJC3DVEC pCent, double rx, double ry, double rz, UI slice = 12, UI stack = 12);	球／楕球描画（ベクトル指定）
38	BOOL DefPlane (UI id, double xc, double yc, double zc, double xv, double yv, double zv, double xo, double yo, double zo);	３Ｄ空間上に任意の平面を定義
39	BOOL DefPlane (UI id, PCAJC3DLVEC pLVec, PCAJC3DVEC pVOrg);	３Ｄ空間上に任意の平面を定義（ベクトル指定）
40	BOOL SetAngle (double rtx, double rty, double rtz);	視点設定（任意設定）
41	BOOL SetAngleXY ();	視点をＸ－Ｙ平面に設定
42	BOOL SetAngleXZ ();	視点をＸ－Ｚ平面に設定
43	BOOL SetAngleYZ ();	視点をＹ－Ｚ平面に設定
44	BOOL SetAngle3D ();	視点を３Ｄイメージに設定
45	HBITMAP GetBitmap ();	ビットマップ取得
46	BOOL LoadProp (C_BCP pProfileSect = "G3dProp", PCAJC3DGPROP pDefProp = NULL);	プロファイルからプロパティ値読出し(ASCII)
47	BOOL LoadProp (C_WCP pProfileSect = L"G3dProp", PCAJC3DGPROP pDefProp = NULL);	プロファイルからプロパティ値読出し(UNICODE)
48	BOOL SaveProp (C_BCP pProfileSect = "G3dProp");	プロファイルへプロパティ値書き込み(ASCII)
49	BOOL SaveProp (C_WCP pProfileSect = L"G3dProp");	プロファイルへプロパティ値書き込み(UNICODE)
50	BOOL LoadPropEx (C_BCP pProfileSect = "G3dProp", PCAJC3DGPROP pDefProp = NULL);	プロファイルから詳細プロパティ値読出し(ASCII)
51	BOOL LoadPropEx (C_WCP pProfileSect = L"G3dProp", PCAJC3DGPROP pDefProp = NULL);	プロファイルから詳細プロパティ値読出し(UNICODE)
52	BOOL SavePropEx (C_BCP pProfileSect = "G3dProp");	プロファイルへ詳細プロパティ値書き込み(ASCII)
53	BOOL SavePropEx (C_WCP pProfileSect = L"G3dProp");	プロファイルへ詳細プロパティ値書き込み(UNICODE)
54	HFONT SetTextFont (HFONT hFont);	テキスト描画フォント設定
55	UI TextOut (int x, int y, C_BCP pTxt);	テキスト描画（ピクセル位置指定）(ASCII)
56	UI TextOut (int x, int y, C_WCP pTxt);	テキスト描画（ピクセル位置指定）(UNICODE)
57	UI PrintF (int x, int y, C_BCP pFmt, ...);	書式テキスト描画（ピクセル位置指定）(ASCII)
58	UI PrintF (int x, int y, C_WCP pFmt, ...);	書式テキスト描画（ピクセル位置指定）(UNICODE)
59	UI TextOutV (AJCG3DTXOMD md, PCAJC3DVEC pV, C_BCP pTxt);	テキスト描画（３Ｄ座標指定）(ASCII)
60	UI TextOutV (AJCG3DTXOMD md, PCAJC3DVEC pV, C_WCP pTxt);	テキスト描画（３Ｄ座標指定）(UNICODE)
61	UI PrintFV (AJCG3DTXOMD md, PCAJC3DVEC pV, C_BCP pFmt, ...);	書式テキスト描画（３Ｄ座標指定）(ASCII)
62	UI PrintFV (AJCG3DTXOMD md, PCAJC3DVEC pV, C_WCP pFmt, ...);	書式テキスト描画（３Ｄ座標指定）(UNICODE)
63	UI GetText (UI key, BCP pBuf, UI lBuf);	描画テキスト取得(ASCII)
64	UI GetText (UI key, WCP pBuf, UI lBuf);	描画テキスト取得(UNICODE)
65	BOOL ClearShape (UI id);	図形描画データクリアー
66	BOOL ClearShape ();	全ての図形描画データクリアー
67	BOOL ClearPlot (UI id);	プロットデータクリアー
68	BOOL ClearPlot ();	全てのプロットデータクリアー
69	BOOL ClearText (UI key);	描画テキストクリアー
70	BOOL ClearText ();	全ての描画テキストクリアー
71	BOOL Clear ();	全てのデータ(図形，プロット，描画テキスト)クリアー

メンバ関数（平面描画ファンクション）

#	変数／関数形式	内容
1	BOOL Pixel (UI id, double x, double y, UI PixelSize);	平面にピクセル描画
2	BOOL Pixel (UI id, PCAJC2DVEC pPoint, UI PixelSize);	平面にピクセル描画（ベクトル指定）
3	BOOL Line (UI id, double x1, double y1, double x2, double y2);	平面にライン描画
4	BOOL Line (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);	平面にライン描画（ベクトル指定）
5	BOOL MoveTo (UI id, double x, double y);	平面に描画するライン／矢印の始点設定
6	BOOL MoveTo (UI id, PCAJC2DVEC ps);	平面に描画するライン／矢印の始点設定（ベクトル指定）
7	BOOL LineTo (UI id, double x, double y);	終点を指定し平面にライン描画
8	BOOL LineTo (UI id, PCAJC2DVEC pe);	終点を指定し平面にライン描画（ベクトル指定）
9	BOOL Arrow (UI id, double x1, double y1, double x2, double y2);	平面に矢印描画
10	BOOL Arrow (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);	平面に矢印描画（ベクトル指定）
11	BOOL ArrowTo (UI id, double x, double y);	終点を指定し平面に矢印描画
12	BOOL ArrowTo (UI id, PCAJC2DVEC pe);	終点を指定し平面に矢印描画（ベクトル指定）
13	BOOL Triangle (UI id, double x1, double y1, double x2, double y2, double x3, double y3);	平面に三角形描画
14	BOOL Triangle (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2, PCAJC2DVEC p3);	平面に三角形描画（ベクトル指定）
15	BOOL Square (UI id, double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4);	平面に四角形描画
16	BOOL Square (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2, PCAJC2DVEC p3, PCAJC2DVEC p4);	平面に四角形描画（ベクトル指定）
17	BOOL Rectangle (UI id, double x1, double y1, double x2, double y2);	平面に長方形描画
18	BOOL Rectangle (UI id, PCAJC2DVEC p1, PCAJC2DVEC p2);	平面に長方形描画（ベクトル指定）
19	BOOL Ellipse (UI id, double xc, double yc, double rx, double ry);	平面に円／楕円描画
20	BOOL Ellipse (UI id, PCAJC2DVEC pCent, double rx, double ry);	平面に円／楕円描画（ベクトル指定）
21	BOOL Star (UI id, double xc, double yc, double r1, double r2 = 0.0, UI nVertex = 5, double rot = 0.0, BOOL fInscribedLine = FALSE);	平面に星形描画
22	BOOL Star (UI id, PCAJC2DVEC pCent , double r1, double r2 = 0.0, UI nVertex = 5, double rot = 0.0, BOOL fInscribedLine = FALSE);	平面に星形描画（ベクトル指定）

メンバ関数（その他のファンクション）

#	変数／関数形式	内容
23	BOOL EnablePopupMenu (BOOL fEnable);	右クリックによるポップアップメニューの許可／禁止
24	BOOL SetNtcRClk (BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);	右クリック通知設定
25	BOOL SetTipText (C_BCP pTxt);	ツールチップの設定(ASCII)
26	BOOL SetTipText (C_WCP pTxt);	ツールチップの設定(UNICODE)
27	UI GetTipText (BCP pBuf, UI lBuf);	ツールチップの取得(ASCII)
28	UI GetTipText (WCP pBuf, UI lBuf);	ツールチップの取得(UNICODE)
29	BOOL SetChkBoxTipText (UI n, C_BCP pTxt);	フィルタ・チェックボックス・ツールチップの設定(ASCII)
30	BOOL SetChkBoxTipText (UI n, C_WCP pTxt);	フィルタ・チェックボックス・ツールチップの設定(UNICODE)
31	UI GetChkBoxTipText (UI n, BCP pBuf, UI lBuf);	フィルタ・チェックボックス・ツールチップの取得(ASCII)
32	UI GetChkBoxTipText (UI n, WCP pBuf, UI lBuf);	フィルタ・チェックボックス・ツールチップの取得(UNICODE)
33	BOOL SetTipShowAlways (BOOL fShowAlways);	ツールチップ表示条件の設定
34	BOOL GetTipShowAlways ();	ツールチップ表示条件の取得
35	BOOL SetChkBoxTipShowAlways(UI n, BOOL fShowAlways);	フィルタ・チェックボックス・ツールチップの設定
36	BOOL GetChkBoxTipShowAlways(UI n);	フィルタ・チェックボックス・ツールチップの取得
37	BOOL SetTipShowAlwaysAll (BOOL fShowAlways);	全ツールチップ表示条件の設定
38	BOOL SetFilter (UI n, BOOL state);	フィルタの設定
39	BOOL GetFilter (UI n);	フィルタの取得
40	BOOL GetDroppedFile (BC buf[MAX_PATH]);	ドロップされたファイル名取得(ASCII)
41	BOOL GetDroppedFile (WC buf[MAX_PATH]);	ドロップされたファイル名取得(UNICODE)
42	BOOL GetDroppedDir (BC buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得(ASCII)
43	BOOL GetDroppedDir (WC buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得(UNICODE)
44	BOOL SetTitleText (C_BCP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定(ASCII)
45	BOOL SetTitleText (C_WCP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定(UNICODE)
46	BOOL Pause (BOOL fPause);	画面表示の停止／再開
47	BOOL EnableMesDraw (BOOL fEnable);	描画時間計測情報の許可／禁止

仮想関数

#	変数／関数形式	内容	
1	virtual VO Attach (HWND hWnd);	ハンドルを関連付け (override 時、継承元 Attach()実行要) ※1	
2	virtual VO OnNtcRotTheta(PCAJC3DVEC pVec);	視点角度通知	AJC3DGN_ROTTHETA
3	virtual VO OnNtcPlotList(PCAJC3DGPLOTLIST pList);	プロットリスト通知	AJC3DGN_PLOTLIST
4	virtual VO OnNtcClear (int Factor);	データクリアー通知	AJC3DGN_CLEAR
5	virtual VO OnNtcDblClk ();	ダブルクリック通知	AJC3DGN_DBLCLK
6	virtual VO OnNtcDropFile(UI n);	ファイルドロップ通知	AJC3DGN_DROPFILE
7	virtual VO OnNtcDropDir (UI n);	フォルダドロップ通知	AJC3DGN_DROPDIR
8	virtual VO OnNtcRClick (PCAJC3DGRCLK pRClk);	右クリック通知	AJC3DGN_RCLICK

※ AJC3DGN_*XXXXX*は、対応する通知コードを意味します。

※１：３Ｄグラフィックウインドを生成したら、Attach()を実行して、３Ｄグラフィックウインドのハンドルを関連付けてください。
例えば、ダイアログボックスに３Ｄグラフィックウインド（ *IDC_G3D* ）を配置した場合、以下のようにします。

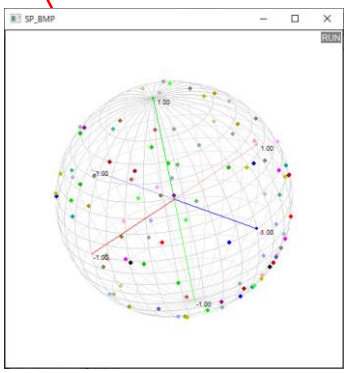
```
CAjxG3d    m_G3d;
   ・・・
m_G3d.Attach(GetDlgItem(IDC_G3D)->m_hWnd);    //  MFC の CDialog クラスの場合
m_G3d.Attach(::GetDlgItem(hDlg, IDC_G3D);     //  Windows API をコールする場合
```

また、CAjxG3d クラスを継承し、Attach()を override する場合は、継承元の Attach()を呼び出す必要があります。

```
class CAjxG3dEx :
    public CAjxG3d
{
public:
    ・・・
    VO  Attach (HWND hwnd)  override
    {
        ・・・・
        CAjxG3d::Attach(hwnd);  // 基底クラスの Attach()呼び出し
    }
    ・・・
};
```

使用例

ウインドをダブルクリックすると、球面上に点の描画を開始／停止します。




```
  1 : //
  2 : //  SP_G3D.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <tchar.h>
  6 : using namespace AjxControl;
  7 :
  8 : #define IDC_G3D     5001
  9 :
 10 : static  HINSTANCE       hInst    = NULL;
 11 : static  HWND            hWndMain = NULL;
 12 : static  HWND            hWndG3d  = NULL;
 13 : static  BOOL            fTimer   = FALSE;
 14 :
 15 : AJC_WNDPROC_DEF(Main);
 16 :
 17 : static const UT TipMsg[] =  TEXT("ダブルクリックで、球面上に点を描画します。¥n")
 18 :                            TEXT("ウインドをドラッグすることにより視点を変更できます。¥n")
 19 :                            TEXT("右クリックでポップアップメニューを表示します¥n")
 20 :                            TEXT("このツールチップ上にカーソルを置くと表示を継続します。¥n")
 21 :                            TEXT("このツールチップ上をクリックするとツールチップは消えます。");
 22 :
 23 : //----- CAjxG3d の派生クラス ----------------------------------------------------------------------------------//
 24 : class CAjxG3dEx :
 25 :     public CAjxG3d
 26 : {
 27 : public:
 28 :     VO      OnNtcDblClk() override  {   //  ダブルクリック通知
 29 :         if (fTimer) {SetTitleText(TEXT("STOP")); KillTimer(hWndMain, 1);}
 30 :         else        {SetTitleText(TEXT("RUN" )); SetTimer(hWndMain, 1, 100, NULL);}
 31 :         fTimer = ~fTimer;
 32 :     }
 33 : };
 34 : static  CAjxG3dEx   g3d;
 35 :
 36 : //----- WinMain ----------------------------------------------------------------------------------------------//
 37 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 38 : {
 39 :     MSG         msg = {0};
 40 :     WNDCLASS    wc  = {0};
 41 :     ATOM        atm = 0;
 42 :
 43 :     hInst = hInstance;
 44 :     //  ウインド生成
 45 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
 46 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 47 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_BMP");
 48 :     atm = RegisterClass(&wc);
 49 :
 50 :     hWndMain = CreateWindow(TEXT("SP_BMP"), TEXT("SP_BMP"), // window class name, caption
 51 :                WS_OVERLAPPEDWINDOW,                         // window style
 52 :                CW_USEDEFAULT, CW_USEDEFAULT, 500, 523,      // position, size
 53 :                NULL, NULL, hInstance, NULL);                // parent, menu, instance, param
 54 :     ShowWindow(hWndMain, iCmdShow);
 55 :     //  メッセージループ
 56 :     while (GetMessage(&msg, NULL, 0, 0)) {
 57 :         TranslateMessage(&msg);
 58 :         DispatchMessage (&msg);
 59 :     }
 60 :     UnregisterClass((UTP)atm, hInstance);
 61 :     return (int)msg.wParam ;
 62 : }
 63 : //----- WM_CREATE --------------------------------------------------------------------------------------------//
 64 : AJC_WNDPROC(Main, WM_CREATE     )
 65 : {
 66 :     RECT    r;
 67 :     //  ３Ｄグラフィックウインド生成
 68 :     GetClientRect(hwnd, &r);
 69 :     hWndG3d = CreateWindow(TEXT("AjcCtrl3dGraph"),
 70 :                            TEXT("P: XR=1, YR=1, ZR=1"),         // window class name, caption
 71 :                            WS_CHILD | AJC3DGS_3DMODE,           // window style
 72 :                            0, 0, r.right, r.bottom,             // position, size
 73 :                            hwnd, (HMENU)IDC_G3D, hInst, NULL);  // parent, menu, instance, param
 74 :     //  ツールチップ設定
 75 :     SAjxTip::ShowCenter(hWndG3d, TipMsg);   //  ツールチップ設定
 76 :     SAjxTip::Add       (hWndG3d, TipMsg);   //  ・
 77 :     g3d.Attach(hWndG3d);                    //  棒グラウインドハンドル割り当て
 78 :     g3d.SetAngle3D();                       //  棒グラウインド視点設定
 79 :     g3d.SetTitleText(TEXT("STOP"));         //  棒グラウインドタイトル設定
 80 :     ShowWindow(hWndG3d, SW_SHOW);           //  棒グラウインド表示
 81 :     return 0;
 82 : }
 83 : //----- WM_DESTROY -------------------------------------------------------------------------------------------//
 84 : AJC_WNDPROC(Main, WM_DESTROY    )
 85 : {
 86 :     DestroyWindow(hWndG3d);
 87 :     PostQuitMessage(0);                     //  プログラム終了
 88 :     return 0;
```

```
 89 : }
 90 : //----- WM_SIZE ----------------------------------------------------------------------------------------------//
 91 : AJC_WNDPROC(Main, WM_SIZE       )
 92 : {
 93 :     MoveWindow(hWndG3d, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
 94 :     return 0;
 95 : }
 96 : //----- WM_TIMER ---------------------------------------------------------------------------------------------//
 97 : AJC_WNDPROC(Main, WM_TIMER      )
 98 : {
 99 :     AJC3DVEC    v;
100 :     v.x = rand() - (RAND_MAX / 2);
101 :     v.y = rand() - (RAND_MAX / 2);
102 :     v.z = rand() - (RAND_MAX / 2);
103 :     AjcV3dNormal(&v, &v);
104 :     g3d.Pixel(rand() & 15, &v, 3);
105 :     return 0;
106 : }
107 : //----------------------------------------------------------------------------------------------------------//
108 : AJC_WNDMAP_DEF(Main)
109 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
110 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
111 :     AJC_WNDMAP_MSG(Main, WM_SIZE    )
112 :     AJC_WNDMAP_MSG(Main, WM_TIMER   )
113 : AJC_WNDMAP_END
```

2.4. ＶＴ－１００エミュレーション・ウインド コントロール（ CAjxVth ）

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxVth(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxVth(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式			内容
1	BOOL	SetProp	(PCAJCVTHPROP pProp);	プロパティ設定
2	BOOL	GetProp	(PAJCVTHPROP pBuf);	プロパティ取得
3	BOOL	GetVramFitSize	(UIP pWidth, UIP pHeight);	ＶＲＡＭにフィットしたウインドサイズ取得
4	BOOL	PutChar	(BC c);	１文字描画(ASCII)
5	BOOL	PutChar	(WC c);	１文字描画(UNICODE)
6	BOOL	PutTextAuto	(C_BCP pTxt, UI lTxt = -1);	マルチバイト文字コードを自動判別してテキスト描画(ASCII)
7	BOOL	PutTextEUC	(C_BCP pTxt, UI lTxt = -1);	日本語ＥＵＣコードテキスト描画 (ASCII)
8	BOOL	PutTextUTF8	(C_BCP pTxt, UI lTxt = -1);	ＵＴＦ－８コードテキスト描画 (ASCII)
9	BOOL	PutText	(C_UTP pTxt, UI lTxt = -1);	テキスト描画
10	BOOL	PrintF	(C_UTP pFmt, ...);	書式テキスト描画
11	BOOL	TimeStamp	();	タイムスタンプ描画
12	BOOL	HexDump	(C_VOP pDat, UI lDat);	バイナリデータの１６進ダンプ描画
13	BOOL	Locate	(UI line, UI col);	カーソル位置設定
14	BOOL	SetColor	(UI PaletteNo);	テキスト描画用パレット設定
15	BOOL	SetBkColor	(UI PaletteNo);	文字背景描画用パレット設定
16	BOOL	SetPalette	(UI PaletteNo, COLORREF rgb);	パレットの色コード設定
17	UI	GetCursorPos	(UIP pLine, UIP pCol);	カーソル位置取得
18	UI	GetColor	();	テキスト描画用パレット番号取得
19	UI	GetBkColor	();	文字背景描画用パレット番号取得
20	COLORREF	GetPalette	(UI PaletteNo);	パレットの色コード取得
21	BOOL	Select	(UI slp, UI scp, UI elp, UI ecp);	部分テキストを選択
22	BOOL	SelectAll	();	全てのテキストを選択
23	BOOL	CopyText	();	選択テキストをクリップボードへコピー
24	BOOL	SetFont	();	ダイアログによるフォントの設定
25	BOOL	ShowCaret	(BOOL fShow);	キャレット表示／非表示
26	BOOL	Clear	();	全テキストクリアー
27	BOOL	ClearAllText	();	同上
28	BOOL	GetDroppedFile	(UT buf[MAX_PATH]);	ドロップされたファイル名取得
29	BOOL	GetDroppedDir	(UT buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得
30	BOOL	LoadProp	(C_UTP pProfileSect = "VthProp", PCAJCVTHPROP pDefProp = NULL);	プロファイルからプロパティ値読出し
31	BOOL	SaveProp	(C_UTP pProfileSect = "VthProp");	プロファイルへプロパティ書き込み
32	BOOL	LoadPermInfo	(C_UTP pProfileSect = _T("VthPerm_*"), C_UTP pKeyPrefix = _T("Vth"), UI PermItem = AJCVTH_PERM_FONT);	設定情報の読み出し
33	BOOL	SavePermInfo	();	設定情報の書き込み
34	BOOL	EnablePopupMenu	(BOOL fEnable);	右クリックによるポップアップメニューの許可／禁止
35	BOOL	SetNtcRClk	(BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);	右クリック通知設定
36	UI	GetText	(UTP pBuf, UI lBuf);	テキストの取得

メンバ関数

#	変数／関数形式			内容
37	UI	GetSelectedText	(UTP pBuf, UI lBuf);	選択されているテキストの取得
38	UI	GetDblClickedLine	(UTP pBuf, UI lBuf);	ダブルクリックした行位置のテキスト取得
39	UI	GetDblClickedLine	(UTP pBuf, UI lBuf, UIP pLine, UIP pCol);	ダブルクリックした行位置のテキスト取得
40	BOOL	SetTipText	(C_UTP pTxt);	ツールチップの設定
41	BOOL	GetTipText	(UTP pBuf, UI lBuf);	ツールチップの取得
42	BOOL	SetTipShowAlways	(BOOL fShowAlways);	ツールチップ表示条件の設定
43	BOOL	GetTipShowAlways	();	ツールチップ表示条件の取得
44	BOOL	GetCharInfo	(UIP pCx, UIP pCy, UIP pLy);	文字サイズ／行の高さ取得
45	UI	GetLinesPerWindow	();	ウインドに表示可能な行数の取得
46	UI	GetValidLines	();	バッファに格納されている有効な行数の取得
47	UI	GetIxOfWndTopLine	();	ウインド先頭行の位置取得（０～）
48	BOOL	SetTitleText	(C_UTP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定
49	BOOL	SaveTextToFile	(C_UTP pPath, BOOL fAllText = TRUE, EAJCTEC tec = AJCTEC_MBC, BOOL fBom = FALSE);	テキストをファイルへ書き込み
50	BOOL	SaveHtmlToFile	(C_UTP pPath, BOOL fAllText = FALSE, UI FontSize = 3);	テキストをＨＴＭＬファイルへ書き込み
51	BOOL	SaveAllTextWithEsc	(C_UTP pPath, EAJCTEC tec = AJCTEC_MBC, BOOL fBom = FALSE);	ＥＳＣシーケンスを付加し全テキストをファイルへ書込み
52	UI	GetLineCount	();	バッファに格納されている行数の取得
53	BOOL	GetCursorPosInfo	(UIP pLine, UIP pCol);	カーソル位置の行番号と桁位置取得
54	UI	GetLineText	(UI pos, UTP pBuf, UI lBuf);	指定行位置の行テキスト取得
55	UI	GetVScrollPos	();	縦スクロール位置の取得
56	UI	GetHScrollPos	();	横スクロール位置の取得
57	BOOL	SetVScrollPos	(UI pos);	縦スクロール位置の設定
58	BOOL	SetHScrollPos	(UI pos);	横スクロール位置の設定
59	BOOL	GetWindowSize	(LPSIZE pSize);	表示ウインドサイズ（行数, 文字数）取得
60	UI	SearchBelow	(C_UTP pStr, BC delimiter = 0);	文字列の後方検索
61	UI	SearchAbove	(C_UTP pStr, BC delimiter = 0);	文字列の前方検索
62	BOOL	Pause	(BOOL fPause);	画面表示の停止／再開
63	BOOL	SetLFActInPopupMenu	(BOOL flag);	改行動作の設定をポップアップメニューに含めるか否かの設定
64	BOOL	GetLFActInPopupMenu	();	改行動作の設定をポップアップメニューに含めるか否かの取得

仮想関数

#	変数／関数形式			内容	
1	virtual VO	Attach	(HWND hWnd);	ハンドルを関連付け（override 時、継承元 Attach()実行要）　※１	
2	virtual VO	OnNtcDblClk	(UI flag);	ダブルクリック通知	AJCVTHN_DBLCLK
3	virtual VO	OnNtcKeyIn	(UI key, UI rep);	キー入力通知	AJCVTHN_KEYIN
4	virtual VO	OnNtcVKeyIn	(UI key, UI rep);	拡張キー押下通知	AJCVTHN_VKEYIN
5	virtual VO	OnNtcVKeyOut	(UI key);	拡張キー離し通知	AJCVTHN_VKEYOUT
6	virtual VO	OnNtcDropFile	(UI nFiles);	ファイルドロップ通知	AJCVTHN_DROPFILE
7	virtual VO	OnNtcDropDir	(UI nDirs);	ディレクトリドロップ通知	AJCVTHN_DROPDIR
8	virtual VO	OnNtcCharInfo	(UI height);	文字サイズ情報の変化通知	AJCVTHN_CHARINFO
9	virtual VO	OnNtcHScroll	(UI left);	横スクロール通知	AJCVTHN_HSCROLL
10	virtual VO	OnNtcVScroll	(UI top);	縦スクロール通知	AJCVTHN_VSCROLL
11	virtual VO	OnNtcRClick	(PCAJCVTHRCLK pRClk);	右クリック通知	AJCVTHN_RCLICK
12	virtual VO	OnNtcClear	();	画面クリアー通知	AJCVTHN_CLEAR

※ AJCVTHN_*XXXXX*は、対応する通知コードを示します。

※１：ＶＴ－１００エミュレーション・ウインドを生成したら、Attach()を実行して、ＶＴ－１００エミュレーション・ウインドのハンドルを関連付けてください。
例えば、ダイアログボックスにＶＴ－１００エミュレーション・ウインド (*IDC_VTH*) を配置した場合、以下のようにします。

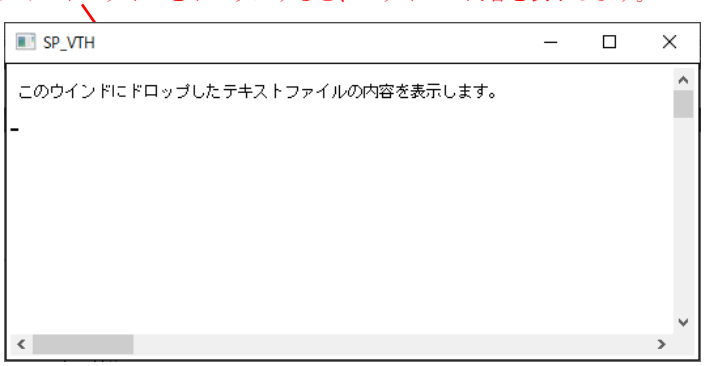
```
CAjxVth    m_Vth;
   ・・・
m_Vth.Attach(GetDlgItem(IDC_VTH)->m_hWnd);    //  MFC の CDialog クラスの場合
m_Vth.Attach(::GetDlgItem(hDlg, IDC_VTH);     //  Windows API をコールする場合
```

また、CAjxVth クラスを継承し、Attach()を override する場合は、継承元の Attach()を呼び出す必要があります。

```
class CAjxVthEx :
    public CAjxVth
{
public:
    ・・・
    VO  Attach (HWND hwnd)  override
    {
        ・・・・
        CAjxVth::Attach(hwnd);  // 基底クラスの Attach()呼び出し
    }
    ・・・
};
```

使用例

ここにテキストファイルをドロップすると、ファイルの内容を表示します。




```
//
//  SP_VTH.cpp
//
#include    <AjxCpp.h>
#include    <tchar.h>
using namespace AjxControl;

#define IDC_VTH 5001

static  HINSTANCE   hInst    = NULL;
static  HWND        hWndMain = NULL;
static  HWND        hWndVth  = NULL;
static  BOOL        fTimer   = FALSE;

AJC_WNDPROC_DEF(Main);
static const UT TipMsg[] =  TEXT("このウインドにドロップしたテキストファイルの内容を表示します。¥n")
                            TEXT("このツールチップ上にカーソルを置くと表示を継続します。¥n")
                            TEXT("このツールチップ上をクリックするとツールチップは消えます。");

//----- CAjxVth の派生クラス --------------------------------------------------------------------------------//
class CAjxVthEx :
    public CAjxVth
{
public:
    VO  OnNtcDropFile (UI nFiles) override  {       //  ファイルドロップ通知
        CAjxFile f;
        UT  path[MAX_PATH];
        UT  buf[1024];
        while (GetDroppedFile(path)) {
            PrintF(TEXT("¥x1B[34m【 %s 】¥x1B[0m¥n"), path);
            if (f.FOpen(path)) {
                while (f.FGetS(buf, AJCTSIZE(buf))) {
                    PutText(buf);
                }
                f.FClose();
            }
        }
    }
};
static  CAjxVthEx   vth;

//----- WinMain ---------------------------------------------------------------------------------------------//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG         msg = {0};
    WNDCLASS    wc  = {0};
    ATOM        atm = 0;

    hInst = hInstance;

    //  ウインド生成
    wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
    wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_VTH");
    atm = RegisterClass(&wc);
    hWndMain = CreateWindowEx(0,
                TEXT("SP_VTH"), TEXT("SP_VTH"),             // window class name, caption
                WS_OVERLAPPEDWINDOW,                        // window style
                CW_USEDEFAULT, CW_USEDEFAULT, 800, 400,     // position, size
                NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
    ShowWindow(hWndMain, iCmdShow);
    //  メッセージループ
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage (&msg);
    }
    UnregisterClass((UTP)atm, hInstance);
    return (int)msg.wParam ;
}
//----- WM_CREATE -------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_CREATE     )
{
    //  ＶＴ１００ウインド生成
    RECT    r;
    GetClientRect(hwnd, &r);
    GetClientRect(hwnd, &r);
    hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,
                            TEXT("AjcCtrlVT100"),                   // window class
                            TEXT("P: VW=512, VH=128, ML=10000, TS=4, FN=ＭＳ ゴシック, LF=12"),     // window caption
                            WS_CHILD,                               // window style
                            0, 0, r.right, r.bottom,                // position, size
                            hwnd, (HMENU)IDC_VTH, hInst, NULL);         // parent, menu, instance, param
```

```
 83 :
 84 :     SAjxTip::Add       (hWndVth, TipMsg); //  ツールチップ設定
 85 :     SAjxTip::ShowCenter(hWndVth, TipMsg); //  ツールチップ初期表示
 86 :     vth.Attach(hWndVth);                  //  VT100 ウインドハンドル割り当て
 87 :     vth.PutText(TEXT("¥n このウインドにドロップしたテキストファイルの内容を表示します。¥n¥n"));
 88 :     ShowWindow(hWndVth, SW_SHOW);         //  VT100 ウインド表示
 89 :     return 0;
 90 : }
 91 : //----- WM_DESTROY ------------------------------------------------------------------------------------------//
 92 : AJC_WNDPROC(Main, WM_DESTROY    )
 93 : {
 94 :     DestroyWindow(hWndVth);
 95 :     PostQuitMessage(0);
 96 :     return 0;
 97 : }
 98 : //----- WM_SIZE ---------------------------------------------------------------------------------------------//
 99 : AJC_WNDPROC(Main, WM_SIZE       )
100 : {
101 :     MoveWindow(hWndVth, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
102 :     return 0;
103 : }
104 : AJC_WNDMAP_DEF(Main)
105 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
106 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
107 :     AJC_WNDMAP_MSG(Main, WM_SIZE    )
108 : AJC_WNDMAP_END
```

2.5. 数値入力コントロール（ CAjxInp ）

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxInp();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	BOOL SetTextFormat (int width, UI precision);	テキスト表示書式の設定
2	BOOL GetProp (PAJCIVPROP pBuf);	プロパティ取得
3	BOOL SetProp (PCAJCIVPROP pProp);	プロパティ設定
4	COLORREF GetBorderColor ();	コントロール外枠の表示色取得
5	BOOL SetBorderColor (COLORREF color);	コントロール外枠の表示色設定
6	COLORREF GetBlinkColor ();	ブリンク表示色取得
7	BOOL SetBlinkColor (COLORREF color);	ブリンク表示色設定
8	HWND GetSilderHandle ();	スライダのウインドハンドル取得
9	HWND GetBtnHandle ();	ボタンのウインドハンドル取得
10	HWND GetTxtHandle ();	テキストボックスのウインドハンドル取得
11	HWND GetSpnHandle ();	スピンボタンのウインドハンドル取得
12	UI GetTxtLen ();	テキストボックスのサイズ（桁数）取得
13	BOOL SetTxtLen (UI len);	テキストボックスのサイズ（桁数）設定
14	BOOL SetNtcRClk (BOOL fNtcRClk, UI msgRBtnDown, UI msgRBtnUp);	右ボタン操作通知設定
15	double GetValue ();	値の取得（実数）
16	int GetValueInt ();	値の取得（整数）
17	BOOL SetValue (double value, BOOL fNtc = FALSE);	値の設定
18	BOOL SetRange (double minValue, double maxValue);	数値範囲の設定
19	BOOL SetSldUnit (double unit);	数値の最小単位値の設定
20	BOOL SetSldPage (double page);	スライダのページサイズ設定
21	BOOL SetSpnStep (double step);	スピンボタンのステップサイズ設定
22	BOOL SetPrecision (int prec);	数値の精度設定
23	BOOL SetTipText (C_UTP pTipTxt);	ツールチップの設定
24	UI GetTipText (UTP pBuf, UI lBuf);	ツールチップの取得
25	BOOL SetTipShowAlways(BOOL fShowAlways);	ツールチップ表示条件の設定
26	BOOL GetTipShowAlways();	ツールチップ表示条件の取得
27	BOOL EnaDefTipText (BOOL fEnable);	デフォルト ツールチップテキストの許可／禁止
28	BOOL GetEditState ();	数値編集状態の取得
29	BOOL LoadPermInfo (C_UTP pProfileSect = _T("InpPerm_*"), C_BCP pKey = _T("Inp"), BOOL fNtc = FALSE);	永続化情報の読み出し
30	BOOL SavePermInfo ();	永続化情報の書き込み

仮想関数

#	変数／関数形式	内容	
1	virtual VO Attach (HWND hWnd);	ハンドルを関連付け（override 時、継承元 Attach()実行要） ※1	
2	virtual VO OnNtcIntValue (int val);	整数モードでの数値変化通知	AJCIVN_INTVALUE
3	virtual VO OnNtcRealValue(double val);	実数モードでの数値変化通知	AJCIVN_REALVALUE
4	virtual VO OnNtcRClick (PCAJCIVRCLK pRClk);	右クリック通知	AJCIVN_RCLICK

※ AJCIVN_*XXXXX*は、対応する通知コードを示します。

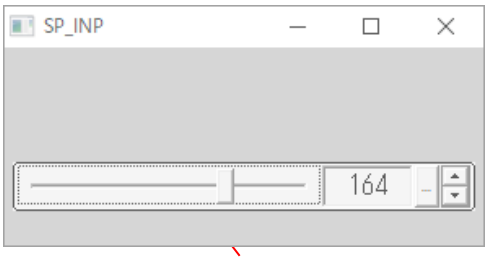
※１：数値入力ウインドを生成したら、Attach()を実行して、数値入力ウインドのハンドルを関連付けてください。
例えば、ダイアログボックスに数値入力ウインド (*IDC_INP*) を配置した場合、以下のようにします。

```
CAjxInp    m_Vth ;
   ・・・
m_Inp.Attach(GetDlgItem(IDC_INP)->m_hWnd);    //  MFC の CDialog クラスの場合
m_Inp.Attach(::GetDlgItem(hDlg, IDC_INP);     //  Windows API をコールする場合
```

また、CAjxINP クラスを継承し、Attach()を override する場合は、継承元の Attach()を呼び出す必要があります。

```
class CAjxInpEx :
    public CAjxInp
{
public:
    ・・・
    VO  Attach (HWND hwnd)  override
    {
        ・・・・
        CAjxInp::Attach(hwnd);  // 基底クラスの Attach()呼び出し
    }
    ・・・
};
```

使用例



自ウインドの透明度（255(不透明)～128(半透明)）を設定します

```
  1 : //
  2 : //  SP_INP.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <tchar.h>
  6 : using namespace AjxControl;
  7 :
  8 : #define IDC_INP 5001
  9 :
 10 : static  HINSTANCE   hInst = NULL;
 11 : static  HWND        hWndMain = NULL;
 12 : static  HWND        hWndInp  = NULL;
 13 : AJC_WNDPROC_DEF(Main);
 14 :
 15 : //----- CAjxInp の派生クラス ----------------------------------------------------------------------------------------//
 16 : class CAjxInpEx :
 17 :     public CAjxInp
 18 : {
 19 : public:
 20 :     VO  OnNtcIntValue (int val) override    {   //  数値入力通知
 21 :         SetLayeredWindowAttributes(hWndMain, 0, val, LWA_ALPHA);
 22 :     }
 23 : };
 24 : static  CAjxInpEx   inp;
 25 :
 26 : //----- WinMain ----------------------------------------------------------------------------------------------------//
 27 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 28 : {
 29 :     MSG         msg = {0};
 30 :     WNDCLASS    wc  = {0};
 31 :     ATOM        atm = 0;
 32 :
 33 :     hInst = hInstance;
 34 :
 35 :     //  ウインド生成
 36 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
 37 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 38 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_INP");
 39 :     atm = RegisterClass(&wc);
 40 :     hWndMain = CreateWindowEx(WS_EX_LAYERED,
 41 :                 TEXT("SP_INP"), TEXT("SP_INP"), // window class name, caption
 42 :                 WS_OVERLAPPEDWINDOW & ~WS_THICKFRAME,       // window style
 43 :                 CW_USEDEFAULT, CW_USEDEFAULT, 300, 150,     // position, size
 44 :                 NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
 45 :     ShowWindow(hWndMain, iCmdShow);
 46 :     //  メッセージループ
 47 :     while (GetMessage(&msg, NULL, 0, 0)) {
 48 :         TranslateMessage(&msg);
 49 :         DispatchMessage (&msg);
 50 :     }
 51 :     UnregisterClass((UTP)atm, hInstance);
 52 :     return (int)msg.wParam ;
 53 : }
 54 : //----- WM_CREATE --------------------------------------------------------------------------------------------------//
 55 : AJC_WNDPROC(Main, WM_CREATE     )
 56 : {
 57 :     SetLayeredWindowAttributes(hwnd, 0, 255, LWA_ALPHA);
 58 :
 59 :     //  数値入力ウインド生成
 60 :     RECT    r;
 61 :     GetClientRect(hwnd, &r);
 62 :     GetClientRect(hwnd, &r);
 63 :     hWndInp = CreateWindow(TEXT("AjcCtrlInpVal"),                   // window class
 64 :                            TEXT("I: L=255, R=128"),                 // window caption
 65 :                            WS_CHILD,                                // window style
 66 :                            5, 70, r.right - 10, 30,                 // position, size
 67 :                            hwnd, (HMENU)IDC_INP, hInst, NULL);      // parent, menu, instance, param
 68 :
 69 :     SAjxTip::ShowCenter(hwnd, TEXT("自ウインドの透明度を設定します。¥n")
 70 :                               TEXT("設定値は、255(不透明)～128(半透明)"));
 71 :     inp.Attach(hWndInp);            //  数値入力ウインドハンドル割り当て
 72 :     inp.SetValue(255);              //  数値初期化(255 : 不透明)
 73 :     ShowWindow(hWndInp, SW_SHOW);   //  数値入力ウインド表示
 74 :     return 0;
 75 : }
 76 : //----- WM_DESTROY -------------------------------------------------------------------------------------------------//
 77 : AJC_WNDPROC(Main, WM_DESTROY    )
 78 : {
 79 :     DestroyWindow(hWndInp);
 80 :     PostQuitMessage(0);
 81 :     return 0;
 82 : }
 83 : //------------------------------------------------------------------------------------------------------------------//
 84 : AJC_WNDMAP_DEF(Main)
 85 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
 86 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
 87 : AJC_WNDMAP_END    82 : AJC_WNDMAP_END
```

2.6. 棒グラフ／折れ線グラフ表示コントロール（ CAjxBar ）

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxBar();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	BOOL PutData (double dat[] , C_UTP pBarTtl); BOOL PutData (double d0 , C_UTP pBarTtl); BOOL PutData (double d0, double d1 , C_UTP pBarTtl); BOOL PutData (double d0, double d1, double d2, C_UTP pBarTtl);	データ投与(実数)
2	BOOL PutData (int dat[] , C_UTP pBarTtl); BOOL PutData (int d0 , C_UTP pBarTtl); BOOL PutData (int d0, int d1 , C_UTP pBarTtl); BOOL PutData (int d0, int d1, int d2 , C_UTP pBarTtl);	データ投与(整数)
3	BOOL SetBarTtl (C_UTP pTtl);	棒タイトル名設定
4		
5	BOOL SetVUnit (C_UTP pUnit);	縦軸単位名設定
6	BOOL Purge ();	データクリアー
7	BOOL ShowBorder (BOOL fShow, COLORREF rgb);	外枠表示／非表示
8	BOOL ShowFilter (BOOL fShow);	フィルタ表示／非表示
9	BOOL GetProp (PAJCBARPROP pBuf);	プロパティ取得
10	BOOL SetProp (PCAJCBARPROP pProp);	プロパティ設定
11	BOOL GetRange (double *pLow, double *pHigh);	レンジ取得（実数）
12	BOOL GetRange (int *pLow, int *pHigh);	レンジ取得（整数）
13	BOOL SetRange (double low, double high);	レンジ設定（実数）
14	BOOL SetRange (int low, int high);	レンジ設定（整数）
15	BOOL GetBase (double *base);	ベース値取得（実数）
16	BOOL GetBase (int *base);	ベース値取得（整数）
17	BOOL SetBase (double base);	ベース値設定（実数）
18	BOOL SetBase (int base);	ベース値設定（整数）
19	BOOL SetBufSize (int n);	バッファサイズ設定
20	BOOL SetItemNumber (int n);	データ項目数設定
21	BOOL SetScaleWidth (int width);	スケール値表示域の幅設定
22	BOOL SetBarWidth (int width);	棒の幅設定
23	BOOL SetItemWidth (int width);	棒表示域の幅設定
24	BOOL SetTtlLines (int lines);	棒タイトルの最大行数設定
25	HBITMAP GetBitmap ();	ビットマップデータ取得
26	BOOL LoadProp (C_UTP pProfileSect = _T("BarProp"), PCAJCBARPROP pDefProp = NULL);	プロファイルからプロパティ読出し
27	BOOL SaveProp (C_UTP pProfileSect = _T("BarProp"));	プロファイルへプロパティ書き込み
28	BOOL LoadPropEx (C_UTP pProfileSect = _T("BarProp"), PCAJCBARPROP pDefProp = NULL);	プロファイルから詳細プロパティ読出し
29	BOOL SavePropEx (C_UTP pProfileSect = _T("BarProp"));	プロファイルから詳細プロパティ書き込み
30	BOOL EnablePopupMenu (BOOL fEnable);	右クリックによるポップアップメニュー許可／禁止
31	BOOL SetNtcRClk (BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);	右クリック通知設定
32	BOOL SetTipText (C_UTP pTxt);	ツールチップの設定
33	UI GetTipText (UTP pBuf, UI lBuf);	ツールチップの取得
34	BOOL SetTipShowAlways (BOOL fShowAlways);	ツールチップ表示条件の設定
35	BOOL GetTipShowAlways ();	ツールチップ表示条件の取得
36	BOOL SetChkBoxTipText (UI n, C_UTP pTxt);	フィィルタチェックボックス・ツールチップの設定
37	UI GetChkBoxTipText (UI n, UTP pBuf, UI lBuf);	フィィルタチェックボックス・ツールチップの取得
38	BOOL SetChkBoxTipShowAlway(UI n, BOOL fShowAlways = FALSE);	フィルタ・チェックボックス・ツールチップ表示条件の設定
39	BOOL GetChkBoxTipShowAlway(UI n);	フィルタ・チェックボックス・ツールチップ表示条件の取得
40	BOOL SetTipShowAlwaysAll (BOOL fShowAlways = FALSE);	全ツールチップ表示条件の設定

メンバ関数

#	変数／関数形式	内容
41	int GetScrollPos ();	スクロール位置の取得
42	BOOL SetScrollPos (int pos);	スクロール位置の設定
43	BOOL SetFilter (UI n, BOOL state);	フィルタの設定
44	BOOL GetFilter (UI n);	フィルタの取得
45	BOOL SetHLineAtt (UI id, COLORREF color, int width = 1, int style = AJCBAR_DASH);	横線の属性設定
46	BOOL SetHLinePos (UI id, double pos);	横線の描画位置設定
47	BOOL EnableHLine (UI id, BOOL fEnable);	横線描画の許可／禁止
48	BOOL GetCharSize (LPSIZE pSize);	文字サイズの取得
49	BOOL GetDroppedFile (UT buf[MAX_PATH]);	ドロップされたファイル名取得
51	BOOL GetDroppedDir (UT buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリ名取得
52	BOOL SetTitleText (C_UTP pTitleText, COLORREF TextColor = (COLORREF)-1, COLORREF BackColor = (COLORREF)-1, HFONT hFont = NULL);	タイトル文字列の設定
53	HFONT SetTextFont (HFONT hFont);	テキスト描画フォント設定
54	UI TextOut (int x, int y, C_BCP pTxt);	テキスト描画（ピクセル位置指定）(ASCII)
55	UI TextOut (int x, int y, C_WCP pTxt);	テキスト描画（ピクセル位置指定）(UNICODE)
56	UI PrintF (int x, int y, C_BCP pFmt, ...);	書式テキスト描画（ピクセル位置指定）(ASCII)
57	UI PrintF (int x, int y, C_WCP pFmt, ...);	書式テキスト描画（ピクセル位置指定）(UNICODE)
58	UI GetText (UI key, BCP pBuf, UI lBuf);	描画テキスト取得(ASCII)
59	UI GetText (UI key, WCP pBuf, UI lBuf);	描画テキスト取得(UNICODE)
60	BOOL ClearText (UI key);	描画テキストクリアー
61	BOOL ClearText ();	全ての描画テキストクリアー
62	BOOL Clear ();	全てのデータと描画テキストクリアー

仮想関数

#	変数／関数形式	内容	
1	virtual VO Attach (HWND hWnd);	ハンドルを関連付け	
2	virtual VO OnNtcRange (PCAJCBAR_NTC_RANGE pRange);	レンジ変化通知	AJCBARN_RANGE
3	virtual VO OnNtcScrPos (UI ScrPos);	スクロール位置変化通知	AJCBARN_SCRPOS
4	virtual VO OnNtcDblClk ();	ダブルクリック通知	AJCBARN_DBLCLK
5	virtual VO OnNtcDropFile (UI nFiles);	ファイルドロップ通知	AJCBARN_DROPFILE
6	virtual VO OnNtcDropDir (UI nDirs);	ディレクトリドロップ通知	AJCBARN_DROPDIR
7	virtual VO OnNtcRClick (PCAJCBARRCLK pRClk);	右クリック通知	AJCBARN_RCLICK

※AJCBARN_*XXXXX*は、対応する通知コードを意味します。

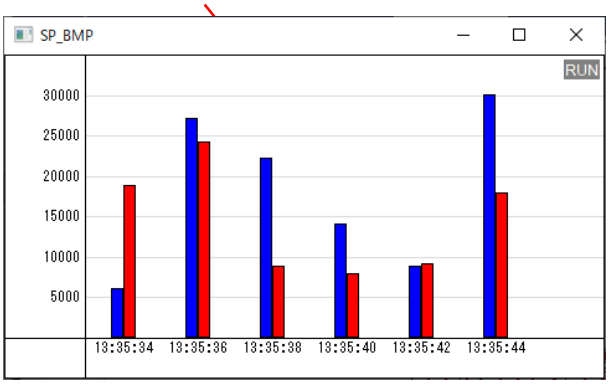
※1：棒グラフ／折れ線グラフ・ウインドを生成したら、Attach()を実行して、棒グラフ／折れ線グラフ・ウインドのハンドルを関連付けてください。
例えば、ダイアログボックスに棒グラフ／折れ線グラフ・ウインド (*IDC_BAR*) を配置した場合、以下のようにします。

```
CAjxBar    m_Bar ;
   ・・・
m_Bar.Attach(GetDlgItem(IDC_BAR)->m_hWnd);    //  MFC の CDialog クラスの場合
m_Bar.Attach(::GetDlgItem(hDlg, IDC_BAR);     //  Windows API をコールする場合
```

また、CAjxBAR クラスを継承し、Attach()を override する場合は、継承元の Attach()を呼び出す必要があります。

```
class CAjxBarEx :
    public CAjxBar
{
public:
    ・・・
    VO  Attach (HWND hwnd)  override
    {
        ・・・・
        CAjxBar::Attach(hwnd);  // 基底クラスの Attach()呼び出し
    }
    ・・・
};
```

２秒毎にランダムな値をグラフ表示します。
ダブルクリックで、グラフ表示を開始／停止します。
CTRL+右クリックで、棒グラフ／折れ線グラフを切り替えます。



使用例

```
  1 : //
  2 : //  SP_BAR.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <tchar.h>
  6 : using namespace AjxControl;
  7 :
  8 : #define IDC_BAR     5001
  9 :
 10 : static  HINSTANCE       hInst    = NULL;
 11 : static  HWND            hWndMain = NULL;
 12 : static  HWND            hWndBar  = NULL;
 13 : static  BOOL            fTimer   = FALSE;
 14 :
 15 : AJC_WNDPROC_DEF(Main);
 16 :
 17 : static const UT TipMsg[] =  TEXT("２秒毎にランダムな値をグラフ表示します。¥n")
 18 :                            TEXT("ダブルクリックで、グラフ表示を開始／停止します。¥n")
 19 :                            TEXT("CTRL+右クリックで、棒グラフ／折れ線グラフを切り替えます。¥n")
 20 :                            TEXT("このツールチップ上にカーソルを置くと表示を継続します。¥n")
 21 :                            TEXT("このツールチップ上をクリックするとツールチップは消えます。");
 22 :
 23 : //----- CAjxBar の派生クラス ----------------------------------------------------------------------------------//
 24 : class CAjxBarEx :
 25 :     public CAjxBar
 26 : {
 27 : public:
 28 :     VO      OnNtcDblClk() override  {                       //  ダブルクリック通知
 29 :         if (fTimer) {SetTitleText(TEXT("STOP")); KillTimer(hWndMain, 1);}
 30 :         else        {SetTitleText(TEXT("RUN" )); SetTimer(hWndMain, 1, 2000, NULL);}
 31 :         fTimer = ~fTimer;
 32 :     }
 33 :     VO      OnNtcRClick(PCAJCBARRCLK pRClk) override    {   //  右クリック通知
 34 :         if (pRClk->fCtrl) {
 35 :             UI  sty = (UI)MAjcGetWindowLong(this->m_hCtrl, GWL_STYLE);
 36 :             MAjcSetWindowLong(this->m_hCtrl, GWL_STYLE, sty ^ AJCBARS_LINEGRAPH);
 37 :         }
 38 :     }
 39 : };
 40 : static  CAjxBarEx   bar;
 41 :
 42 : //----- WinMain ----------------------------------------------------------------------------------------------//
 43 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 44 : {
 45 :     MSG         msg = {0};
 46 :     WNDCLASS    wc  = {0};
 47 :     ATOM        atm = 0;
 48 :
 49 :     hInst = hInstance;
 50 :     //  ウインド生成
 51 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
 52 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 53 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_BMP");
 54 :     atm = RegisterClass(&wc);
 55 :     hWndMain = CreateWindow(    TEXT("SP_BMP"), TEXT("SP_BMP"), // window class name, caption
 56 :                 WS_OVERLAPPEDWINDOW,                        // window style
 57 :                 CW_USEDEFAULT, CW_USEDEFAULT, 500, 300,     // position, size
 58 :                 NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
 59 :     ShowWindow(hWndMain, iCmdShow);
 60 :     //  メッセージループ
 61 :     while (GetMessage(&msg, NULL, 0, 0)) {
 62 :         TranslateMessage(&msg);
 63 :         DispatchMessage (&msg);
 64 :     }
 65 :     UnregisterClass((UTP)atm, hInstance);
 66 :     return (int)msg.wParam ;
 67 : }
 68 : //----- WM_CREATE --------------------------------------------------------------------------------------------//
 69 : AJC_WNDPROC(Main, WM_CREATE     )
 70 : {
 71 :     RECT    r;
 72 :     srand(GetTickCount());
 73 :     //  棒グラウインド生成
 74 :     GetClientRect(hwnd, &r);
 75 :     hWndBar = CreateWindow(TEXT("AjcCtrlBarGraph"),                 // window class
 76 :                            TEXT("P: L=0, H=35000, I=2, MW=60, TL=1"),   // window caption
 77 :                            WS_CHILD,                                // window style
 78 :                            0, 0, r.right, r.bottom,                 // position, size
```

```
 79 :                             hwnd, (HMENU)IDC_BAR, hInst, NULL);          // parent, menu, instance, param
 80 :
 81 :     SAjxTip::Add       (hWndBar, TipMsg);   //  ツールチップ設定
 82 :     SAjxTip::ShowCenter(hWndBar, TipMsg);   //  ・
 83 :     bar.Attach(hWndBar);                    //  棒グラフウインドハンドル割り当て
 84 :     bar.SetTitleText(TEXT("STOP"));         //  棒グラフウインドタイトル設定
 85 :     ShowWindow(hWndBar, SW_SHOW);           //  棒グラフウインド表示
 86 :     return 0;
 87 : }
 88 : //----- WM_DESTROY ----------------------------------------------------------------------------------------------//
 89 : AJC_WNDPROC(Main, WM_DESTROY    )
 90 : {
 91 :     DestroyWindow(hWndBar);
 92 :     PostQuitMessage(0);
 93 :     return 0;
 94 : }
 95 : //----- WM_SIZE -------------------------------------------------------------------------------------------------//
 96 : AJC_WNDPROC(Main, WM_SIZE       )
 97 : {
 98 :     MoveWindow(hWndBar, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
 99 :     return 0;
100 : }
101 : //----- WM_TIMER ------------------------------------------------------------------------------------------------//
102 : AJC_WNDPROC(Main, WM_TIMER      )
103 : {
104 :     SYSTEMTIME  st;
105 :     UT          szTime[32];
106 :     GetLocalTime(&st);
107 :     AjcSnPrintF(szTime, AJCTSIZE(szTime), TEXT("%02d:%02d:%02d"), st.wHour, st.wMinute, st.wSecond);
108 :     int x = rand();
109 :     int y = rand();
110 :     bar.PutData(x, y, szTime);
111 :     return 0;
112 : }
113 : //---------------------------------------------------------------------------------------------------------------//
114 : AJC_WNDMAP_DEF(Main)
115 :     AJC_WNDMAP_MSG(Main, WM_CREATE  )
116 :     AJC_WNDMAP_MSG(Main, WM_DESTROY )
117 :     AJC_WNDMAP_MSG(Main, WM_SIZE    )
118 :     AJC_WNDMAP_MSG(Main, WM_TIMER   )
119 : AJC_WNDMAP_END
```

2.7. リストボックス・コントロール (CAjxLbx)

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxLbx(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxLbx(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	UI GetAllItems (PAJCLBXITEM *pArr);	全リストボックス項目データの配列取得
2	VO RelAllItems (VOP pArr);	取得した全リストボックス項目の配列解放
3	UI GetSelectedItems(PAJCLBXITEMA *pArr);	全選択項目の配列取得
4	VO RelSelectedItems(VOP pArr);	取得した選択項目の配列解放
5	BOOL LoadItems (C_UTP pSect, C_UTP pKeyPrefix = TEXT("Lbx"));	プロファイルから全リストボックス項目の読出し
6	BOOL SaveItems (C_UTP pSect, C_UTP pKeyPrefix = TEXT("Lbx"));	プロファイルへ全リストボックス項目の書込み
7	BOOL SetTipText (C_UTP pTxt);	ツールチップの設定
8	BOOL GetTipText (UTP pBuf, UI lBuf);	ツールチップの取得
9	BOOL SetTipShowAlways(BOOL fShowAlways);	ツールチップ表示条件の設定
10	BOOL GetTipShowAlways();	ツールチップ表示条件の取得
11	int AddString (C_UTP pStr);	リストボックス項目の追加
12	int InsertString (UI ix, C_UTP pStr);	リストボックス項目の挿入
13	int DeleteString (UI ix);	リストボックス項目の削除
14	int FindString (UI ix, C_UTP pStr);	リストボックス項目の検索
15	int GetCount ();	リストボックス項目の項目数取得
16	int GetCurSel ();	選択されているリストボックス項目のインデクス取得
17	BOOL GetSel (UI ix);	リストボックス項目の選択状態取得
18	int GetSelCount ();	選択されているリストボックス項目の個数取得
19	BOOL GetText (UI ix, UTP pBuf, UI lBuf);	リストボックス項目の取得
20	int GetTextLen (UI ix);	リストボックス項目の文字列長取得
21	BOOL ResetContent ();	全リストボックス項目の消去
22	int SelectString (UI ix, C_UTP pStr);	リストボックス項目の選択
23	BOOL SetCount (UI count);	リストボックス項目数の設定
24	BOOL SetCurSel (UI ix);	リストボックス項目の選択
25	BOOL SetSel (UI ix, BOOL fSelect);	リストボックス項目の選択／非選択状態設定
26	BOOL SetItemData (UI ix, UX data);	リストボックス項目に関連付けられた数値の設定
27	UX GetItemData (UI ix);	リストボックス項目に関連付けられた数値の取得
28	BOOL SetBasePath (C_UTP pBasePath);	相対アドレス変換時のベースディレクトリパス設定
29	BOOL SetNtcRClk (BOOL fNtcRClk, UI MsgRBDown, UI MsgRBUp);	右クリック通知設定
30	BOOL SetFileFilter (C_UTP pFilter, C_UTP pDefExt)	ファイル選択時のフィルタ，デフォルト拡張子の設定
31	BOOL GetDroppedFile (UT buf[MAX_PATH]);	ドロップされたファイルのパス名を取得
32	BOOL GetDroppedDir (UT buf[MAX_PATH], BOOL fTailIsDelimiter = TRUE);	ドロップされたディレクトリのパス名を取得
33	BOOL GetRemovedItem (UTP pBuf, UI lBuf);	削除された項目の文字列を取得

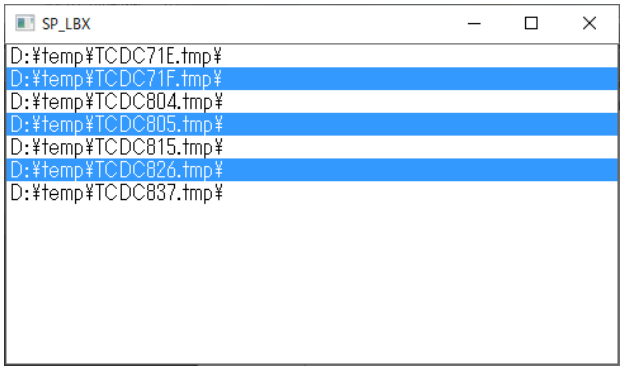
仮想関数

#	変数／関数形式	内容	
1	virtual VO Attach (HWND hWnd);	ハンドルを関連付け (override 時、継承元 Attach()実行要)	
2	virtual VO OnNtcDblClk ();	ダブルクリック通知	AJCLBXN_DBLCLK
3	virtual VO OnNtcSelCancel ();	選択キャンセル通知	AJCLBXN_SELCANCEL
4	virtual VO OnNtcSelChange ();	選択変更通知	AJCLBXN_SELCHANGE
5	virtual VO OnNtcSetFocus ();	フォーカス取得通知	AJCLBXN_SETFOCUS
6	virtual VO OnNtcKillFocus ();	フォーカス喪失通知	AJCLBXN_KILLFOCUS
7	virtual VO OnNtcRClick (PAJCLBXRCLK pRClk);	右クリック通知	AJCLBXN_RCLICK
8	virtual VO OnNtcRemoved (UI n);	項目削除通知	AJCLBXN_REMOVED
9	virtual VO OnNtcDropDir (UI n);	ディレクトリドロップ通知	AJCLBXN_DROPDIR
10	virtual VO OnNtcDropFile (UI n);	ファイルロップ通知	AJCLBXN_DROPFILE
11	virtual VO OnNtcErrSpace ();	メモリ不足通知	AJCLBXN_ERRSPACE

※ AJCLBXN_*XXXXX*は対応する通知コードを示します。

使用例

ドロップしたディレクトリ群をリストボックスに追加します。
最後にドロップしたディレクトリ群は選択状態にします。




```
 1 : //
 2 : //  SP_LBX.cpp
 3 : #include    <AjxCpp.h>
 4 : #include    <tchar.h>
 5 : using namespace AjxControl;
 6 :
 7 : #define IDC_LBX     5001
 8 :
 9 : static  HINSTANCE       hInst    = NULL;
10 : static  HWND            hWndMain = NULL;
11 : static  HWND            hWndLbx  = NULL;
12 :
13 : AJC_WNDPROC_DEF(Main);
14 :
15 : static const UT TipMsg[] =  TEXT("ドロップしたディレクトリをリストボックスに追加し、¥n")
16 :                            TEXT("当該ディレクトリを選択状態にします。");
17 :
18 : //----- CAjxLbx の派生クラス ----------------------------------------------------------------------------------//
19 : class CAjxLbxEx :
20 :     public CAjxLbx
21 : {
22 : public:
23 :     VO      OnNtcDropDir (UI n) override    {               //  ディレクトリドロップ通知
24 :         //  選択状態解除
25 :         SetSel(-1, FALSE);
26 :         //  ドロップしたディレクトリを選択状態にする
27 :         UT  path[MAX_PATH];
28 :         while (GetDroppedDir(path)) {
29 :             SelectString (-1, path);
30 :         }
31 :     }
32 : };
33 : static  CAjxLbxEx   bar;
34 :
35 : //----- WinMain ----------------------------------------------------------------------------------------------//
36 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
37 : {
38 :     MSG         msg = {0};
39 :     WNDCLASS    wc  = {0};
40 :     ATOM        atm = 0;
41 :
42 :     hInst = hInstance;
43 :     //  ウインド生成
44 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
45 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
46 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_LBX");
47 :     atm = RegisterClass(&wc);
48 :     hWndMain = CreateWindow(    TEXT("SP_LBX"), TEXT("SP_LBX"), // window class name, caption
49 :                 WS_OVERLAPPEDWINDOW,                            // window style
```

```
 50 :                CW_USEDEFAULT, CW_USEDEFAULT, 500, 293,          // position, size
 51 :                NULL, NULL, hInstance, NULL);                    // parent, menu, instance, param
 52 :    ShowWindow(hWndMain, iCmdShow);
 53 :    //  メッセージループ
 54 :    while (GetMessage(&msg, NULL, 0, 0)) {
 55 :        TranslateMessage(&msg);
 56 :        DispatchMessage (&msg);
 57 :    }
 58 :    UnregisterClass((UTP)atm, hInstance);
 59 :    return (int)msg.wParam ;
 60 : }
 61 : //----- WM_CREATE ------------------------------------------------------------------------------------------//
 62 : AJC_WNDPROC(Main, WM_CREATE     )
 63 : {
 64 :    RECT    r;
 65 :    //  リストボックス生成
 66 :    GetClientRect(hwnd, &r);
 67 :    hWndLbx = CreateWindowEx(WS_EX_ACCEPTFILES,
 68 :                             TEXT("AjcCtrlListBox"),                  // window class
 69 :                             TEXT(""),                                // window caption
 70 :                             WS_CHILD |                               // window style
 71 :                             AJCLBXS_FILE          |                  //  リスト項目としてファイル／フォルダのパス名を扱う
 72 :                             AJCLBXS_ADDITEMINDROP |                  //  ドロップ操作で、当該ディレクトリ／ファイルをリストボックス項目として追加する
 73 :                             AJCLBXS_DIRTAIL       |                  //  ディレクトリ名の末尾に「¥」を付加する
 74 :                             AJCLBXS_ACCEPTDIRS    |                  //  ディレクトリのドラッグ＆ドロップを可能とする
 75 :                             AJCLBXS_SORT,                            //  アイテム追加時にソートする
 76 :                             0, 0, r.right, r.bottom,                 // position, size
 77 :                             hwnd, (HMENU)IDC_LBX, hInst, NULL);      // parent, menu, instance, param
 78 :
 79 :    SAjxTip::Add       (hWndLbx, TipMsg);   //  ツールチップ設定
 80 :    SAjxTip::ShowCenter(hWndLbx, TipMsg);   //  ・
 81 :    bar.Attach(hWndLbx);                    //  リストボックスハンドル割り当て
 82 :    ShowWindow(hWndLbx, SW_SHOW);           //  リストボックス表示
 83 :    return 0;
 84 : }
 85 : //----- WM_DESTROY -----------------------------------------------------------------------------------------//
 86 : AJC_WNDPROC(Main, WM_DESTROY    )
 87 : {
 88 :    DestroyWindow(hWndLbx);
 89 :    PostQuitMessage(0);
 90 :    return 0;
 91 : }
 92 : //----- WM_SIZE --------------------------------------------------------------------------------------------//
 93 : AJC_WNDPROC(Main, WM_SIZE       )
 94 : {
 95 :    MoveWindow(hWndLbx, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
 96 :    return 0;
 97 : }
 98 : //--------------------------------------------------------------------------------------------------------//
 99 : AJC_WNDMAP_DEF(Main)
100 :    AJC_WNDMAP_MSG(Main, WM_CREATE  )
101 :    AJC_WNDMAP_MSG(Main, WM_DESTROY )
102 :    AJC_WNDMAP_MSG(Main, WM_SIZE    )
103 : AJC_WNDMAP_END
```

2.8. シリアル通信（CAjxScp）

メンバ変数

#	変数／関数形式	内容
1	HAJCSCP m_hScp;	ＳＣＰインスタンスドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxScp();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	#ifdef _CONSOLE　　//　コンソールアプリ 　BOOL Init (C_UTP pSect, 　　AJCSCP_CHUNKMODE ChunkMode = AJCSCP_CM_BIN, 　　BOOL fUseWaitEvent = **TRUE**, 　　BOOL fCreateMySlot = TRUE, 　　C_UTP pMySlot　　= _T("MySlot"), 　　C_UTP pRmtHost　　= _T(""), 　　C_UTP pRmtSlot　　= _T"RmtSlot")); #else　　//　Windows アプリ 　BOOL Init (C_UTP pSect, 　　AJCSCP_CHUNKMODE ChunkMode = AJCSCP_CM_BIN, 　　BOOL fUseWaitEvent = **FALSE**, 　　BOOL fCreateMySlot = TRUE, 　　C_UTP pMySlot　　= _T("MySlot"), 　　C_UTP pRmtHost　　= _T(""), 　　C_UTP pRmtSlot　　= _T"RmtSlot")); #endif	初期設定 pSect　　：通信パラメタ永続化用プロファイルセクション名 ChunkMode　　：受信チャンクデータ通知モード fUseWaitEvent：WaitEvent()関数使用の可否 fCreateMySlot：自メールスロット生成フラグ pMySlot　　：自メールスロット名 pRmtHost　　：通信相手ホスト名 pRmtSlot　　：通信相手スロット名 **※ 他の関数に先駆けて、最初に必ず実行しなければなりません。** ※ 内部で、以下の関数を実行します。 ・AjcScpCreateEx() ・AjcScpSetMode()
2	BOOL　　Open　　();　　ポートオープン（選択済通信リソース）
3	BOOL　　Open　　(UI Port, UI Rate = 115200, UI DataBits = 8, UI Parity = 'N', UI StopBit = 1);　　ポートオープン（ＣＯＭポート）
4	BOOL　　Open　　(UI Port, LPDCB pDcb, LPCOMMTIMEOUTS pTmo = NULL);　　ポートオープン（ＣＯＭポート，DCB，TIMEOUT 指定）
5	BOOL　　Open　　(C_UTP pRmtHost, C_UTP pRmtSlot);　　ポートオープン（メールスロット）
6	BOOL　　Open　　(C_UTP pServ, UI PortNo);　　ポートオープン（ソケット）
7	BOOL　　Open　　(AJCSCP_PORTSEL sel);　　ポートオープン（初回通信リソース選択）
8	BOOL　　Close　　();　　ポートクローズ
9	UI　　GetSelectedPort();　　設定されている通信ポート種別取得
10	UI　　IsOpened　　();　　ポートオープン状態取得
11	BOOL　　SendChar　　(UT code);　　１文字送信
12	BOOL　　SendWord14LF　　(UI data);　　バイトペアによるワード（14Bit）データ送信（Low byte first）
13	BOOL　　SendWord14HF　　(UI data);　　バイトペアによるワード（14Bit）データ送信（High byte first）
14	BOOL　　SetByteSeqRxWord14(BOOL fLowByteFirst);　　１４ビットワードデータ（バイトペア）受信時のバイト順設定
15	BOOL　　GetByteSeqRxWord14();　　１４ビットワードデータ（バイトペア）受信時のバイト順取得
16	BOOL　　SendText　　(C_UTP pTxt, UI lTxt = -1);　　テキストデータ送信
17	BOOL　　SendTextF　　(C_UTP pFmt, ...);　　書式テキストデータ送信(ASCII)
18	BOOL　　SendBinData　　(C_VOP pDat, UI lDat);　　バイナリデータ送信
19	UI　　SendPacket　　(C_VOP pPkt, UI lPkt);　　パケットデータ送信
20	BOOL　　SetEvtMask　　(UI Mask);　　イベントマスク設定（使用イベントの選択）
21	UI　　GetEvtMask　　();　　イベントマスク取得
22	BOOL　　SendBreak　　(BOOL fBreak);　　ブレーク信号送出／停止
23	BOOL　　SetDTR　　(BOOL fActive);　　ＤＴＲ信号設定
24	BOOL　　SetRTS　　(BOOL fActive);　　ＲＴＳ信号設定

メンバ関数

#	変数／関数形式	内容
25	UI GetSigState ();	信号状態取得
26	ULL GetTxBytes ();	送信待ちデータバイト数取得
27	BOOL PurgeRecvData ();	全受信済データ破棄
28	BOOL PurgeSendData ();	全送信待ちデータ破棄
29	BOOL PurgeAllData ();	全送受信データ破棄
30	BOOL SetParam (const DCB *pDcb = NULL, const COMMTIMEOUTS *pTmo = NULL);	ＤＣＢ情報とタイムアウト情報設定
31	UI GetParam (LPDCB pDcb = NULL, LPCOMMTIMEOUTS pTmo = NULL);	ＤＣＢ情報とタイムアウト情報取得
32	BOOL WaitEvent (UI msTime);	イベント発生待ち ※イベントが発生した場合、対応する仮想関数が実行されます。
33	BOOL SetPktCtrlCode (UI stx = 0 , UI etx = 0 , UI dle = 0);	パケットフレームを認識する為の制御コード設定
34	BOOL GetPktCtrlCode (UIP pStx = NULL, UIP pEtx = NULL, UIP pDle = NULL);	パケットフレームを認識する為の制御コード取得
35	BOOL SetPktTimeout (UI msTime);	パケットフレーム受信タイムアウト値設定
36	BOOL GetPktTimeout (UIP pMsTime);	パケットフレーム受信タイムアウト値取得
37	BOOL EnableComPortSelection (BOOL fEnableComPort);	通信パラメタ設定ダイアログによるＣＯＭポートの選択許可／禁止
38	BOOL EnableMailslotSelection (BOOL fEnableMailSlot);	通信パラメタ設定ダイアログによるメールスロットの選択許可／禁止
39	BOOL EnableSocketSelection (BOOL fEnableSocket);	通信パラメタ設定ダイアログによるソケット通信 の選択許可／禁止
40	BOOL EnablePortSelection (BOOL fEnableComPort, BOOL fEnableMailSlot, BOOL fEnableSocket);	ＣＯＭポート、メールスロット，およびソケット通信の選択許可／禁止
41	UI DlgParamEasy (HWND hWndOwner);	ダイアログによる通信パラメタ設定
42	UI DlgParamEasy (HWND hWndOwner, int x, int y);	ダイアログによる通信パラメタ設定（ウインド位置指定）
43	UI DlgParamDetail (HWND hWndOwner);	ダイアログによるＣＯＭポート通信パラメタ詳細設定
44	UI DlgParamDetailEx (HWND hWndOwner, int x, int y);	ダイアログによるＣＯＭポート通信パラメタ詳細設定（ウインド位置指定）
45	VO GetPortPathName (UTP *pPathName);	ポートパス名称取得
46	VO GetPortName (UTP *pName);	ポート名称取得
47	BOOL SetChunkMode (AJCSCP_CHUNKMODE ChunkMode);	チャンクデータの通知モード設定
48	AJCSCP_CHUNKMODE GetChunkMode ();	チャンクデータの通知モード取得
49	BOOL SetRxTextCode (AJCSCP_TEXTCODE code);	受信テキストの文字コード種別設定
50	AJCSCP_TEXTCODE GetRxTextCode ();	受信テキストの文字コード種別取得
51	BOOL SetTxTextCode (AJCSCP_TEXTCODE code);	送信テキストの文字コード種別設定
52	AJCSCP_TEXTCODE GetTxTextCode ();	送信テキストの文字コード種別取得
53	AJCSCP_TEXTCODE GetActualRxTextCode();	実際の受信テキストの文字コード種別取得
54	AJCSCP_TEXTCODE GetActualTxTextCode();	実際の送信テキストの文字コード種別取得
55	BOOL CreateMySlot ();	自メールスロット生成
56	BOOL DeleteMySlot ();	自メールスロット消去
57	BOOL MySlotIsCreated ();	自メールスロット生成状態取得
58	VO GetMySlotPathName (UTP *pPathName);	自メールスロットのパス名取得
59	BOOL GetMyComputerName (UT pBuf[AJCMAX_HOSTNAME_LENGTH]);	自コンピュータ名取得
60	BOOL SetMailSlotNames (C_UTP pMySlot, C_UTP pRmyHost, C_UTP pRmtSlot);	メールスロット名情報設定
61	BOOL GetMailSlotNames (UT pMySlot[64], UT pRmyHost[AJCMAX_HOSTNAME_LENGTH], UT pRmtSlot[64]);	メールスロット名情報取得
62	BOOL SetTxSpeedLimit (BOOL flag, UI bps);	メールスロット送信制限速度の設定
63	BOOL GetTxSpeedLimit (UIP pBps);	メールスロット送信制限速度の取得

仮想関数

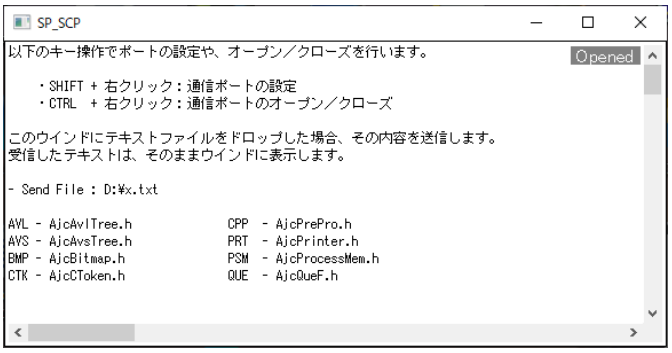
#	変数／関数形式	内容	
1	virtual VO OnNtcPortState (C_UTP pPortName, UI Param);	ポート状態通知	AJCSCP_EV_PORTSTATE
2	virtual VO OnNtcRxTextChunk(C_UTP pText);	テキストチャンク受信通知	AJCSCP_EV_RXCHUNK
3	virtual VO OnNtcRxBinChunk (C_VOP pData, UI Bytes);	バイナリチャンク受信通知	
4	virtual VO OnNtcRxText (C_UTP pText);	テキスト受信通知	AJCSCP_EV_RXTEXT
5	virtual VO OnNtcRxEsc (C_UTP pData);	ＥＳＣシーケンス受信通知	AJCSCP_EV_RXESC
6	virtual VO OnNtcRxCtrl (UI Ctrl);	制御コード受信通知	AJCSCP_EV_RXCTRL
7	virtual VO OnNtcRxPkt (C_VOP pData, UI Bytes);	パケット受信通知	AJCSCP_EV_RXPKT
8	virtual VO OnNtcTxEmpty ();	送信完了通知	AJCSCP_EV_TXEMPTY
9	virtual VO OnNtcRxNoPkt (C_UTP pText);	パケット外データ受信通知	AJCSCP_EV_RXNOPKT
10	virtual VO OnNtcInvChunk (C_VOP pData, UI lData);	不正テキストチャンク受信通知	AJCSCP_EV_INVCHUNK
11	virtual VO OnNtcErr (UI param);	通信エラー発生通知	AJCSCP_EV_ERR
12	virtual VO OnNtcBreak ();	BREAK 検出通知	AJCSCP_EV_BREAK
13	virtual VO OnNtcRing (UI param);	RING 変化通知	AJCSCP_EV_RING
14	virtual VO OnNtcRlsd (UI param);	RLSD 変化通知	AJCSCP_EV_RLSD
15	virtual VO OnNtcDsr (UI param);	DSR 変化通知	AJCSCP_EV_DSR
16	virtual VO OnNtcCts (UI param);	CTS 変化通知	AJCSCP_EV_CTS
17	virtual VO OnNtcRxWord14 (UW data);	バイトペアによるワード(14Bit)受信通知	AJCSCP_EV_RXWORD14

※ AJCSCP_EV_*XXXXX*は、対応するイベントコードを示します。

スタティック関数

#	変数／関数形式	内容
1	static UI EnumSerialPorts (UBP pMap, UI lMap, UI CurrentPort);	ＣＯＭポートの列挙
2	static int GetPortDevName (C_UTP pPortName, BCP pBuf, UI lBuf);	ＣＯＭポートのデバイス名取得 （デバイスマネージャーで表示されるデバイス名称の取得）

使用例１ (Windows アプリ)

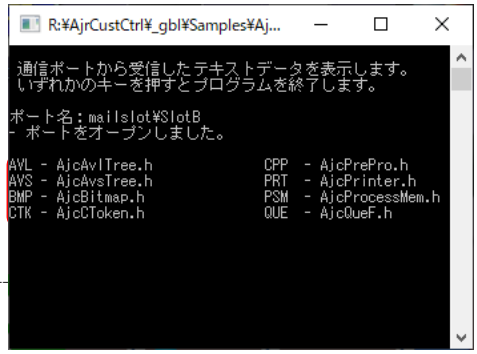



```
  1 : //
  2 : //  SP_SCP1.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <tchar.h>
  6 : using namespace AjxControl;
  7 :
  8 : #define IDC_VTH     5001
  9 : #define WM_MYPOPUP  (WM_APP + 10)
 10 : #define MID_SETPORT 1
 11 : #define MID_OPEN    2
 12 : #define RB_DOWN     999
 13 :
 14 : static  HINSTANCE   hInst    = NULL;
 15 : static  HWND        hWndMain = NULL;
 16 : static  HWND        hWndVth  = NULL;
 17 : static  BOOL        fTimer   = FALSE;
 18 : static const UT TipMsg[] =  TEXT("右クリック(ポップアップメニュー)でポートの設定や、オープン／クローズを行います。¥n")
 19 :                            TEXT("このウインドにテキストファイルをドロップすると、その内容を送信します。¥n")
 20 :                            TEXT("受信したテキストは、そのままウインドに表示します。");
 21 : AJC_WNDPROC_DEF(Main);
 22 : //----- CAjxScp + CAjxVth の派生クラス ------------------------------------------------------------------------//
 23 : class CAjxVthEx :
 24 :     public CAjxScp,
 25 :     public CAjxVth
 26 : {
 27 : public:
 28 :     VO  OnNtcPortState(C_UTP pPortName, UI Param) override {    //  ポート状態通知
 29 :         switch (Param) {
 30 :             case AJCSCP_CLOSED:     SetTitleText(TEXT(" Closed "));             break;
 31 :             case AJCSCP_OPENED:     SetTitleText(TEXT(" Opened "));             break;
 32 :             case AJCSCP_OPENFAIL:   SetTitleText(TEXT(" Closed "));
 33 :               PrintF(TEXT("¥x1b[31m ポートのオープンを失敗しました¥x1b[0m¥n"));  break;
 34 :         }
 35 :     }
 36 :     VO  OnNtcRxTextChunk(C_UTP pText) override {               //  テキストチャンク受信通知
 37 :         PutText(pText);
 38 :     }
 39 :     VO  OnNtcDropFile (UI nFiles) override  {                  //  ファイルドロップ通知
 40 :         CAjxFile f;
 41 :         UT  path[MAX_PATH];
 42 :         UT  buf[1024];
 43 :         while (GetDroppedFile(path)) {
 44 :             PrintF(TEXT("- Send File : %s¥n¥n"), path);
 45 :             SendTextF(TEXT("¥n【【【【【 %s 】】】】】¥n"), path);
 46 :             if (f.FOpen(path)) {
 47 :                 while (f.FGetS(buf, AJCTSIZE(buf))) {
 48 :                     SendText(buf);
 49 :                 }
 50 :                 f.FClose();
 51 :             }
 52 :         }
 53 :     }
 54 : };
 55 : static  CAjxVthEx   scp_vth;
 56 : //----- WinMain ----------------------------------------------------------------------------------------------//
 57 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 58 : {
 59 :     MSG         msg = {0};
 60 :     WNDCLASS    wc  = {0};
 61 :     ATOM        atm = 0;
 62 :
 63 :     hInst = hInstance;
 64 :
 65 :     //  ウインド生成
 66 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
 67 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 68 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_SCP");
 69 :     atm = RegisterClass(&wc);
 70 :     hWndMain = CreateWindowEx(0,
 71 :                 TEXT("SP_SCP"), TEXT("SP_SCP"),             // window class name, caption
 72 :                 WS_OVERLAPPEDWINDOW,                        // window style
 73 :                 CW_USEDEFAULT, CW_USEDEFAULT, 800, 400,     // position, size
 74 :                 NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
 75 :     ShowWindow(hWndMain, iCmdShow);
```

```
76 :     //  メッセージループ
77 :     while (GetMessage(&msg, NULL, 0, 0)) {
78 :         TranslateMessage(&msg);
79 :         DispatchMessage (&msg);
80 :     }
81 :     UnregisterClass((UTP)atm, hInstance);
82 :     return (int)msg.wParam ;
83 : }
84 : //----- WM_CREATE ------------------------------------------------------------------------------------------//
85 : AJC_WNDPROC(Main, WM_CREATE     )
86 : {
87 :     //  ＳＣＰ初期化
88 :     scp_vth.Init    (TEXT("MyScpSect"), AJCSCP_CM_TEXT);
89 :     scp_vth.SetEvtMask(AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXCHUNK);
90 :     //  ＶＴ１００ウインド生成
91 :     RECT    r;
92 :     GetClientRect(hwnd, &r);
93 :     GetClientRect(hwnd, &r);
94 :     hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,
95 :                              TEXT("AjcCtrlVT100"),                  // window class
96 :                              TEXT("P: VW=512, VH=128, ML=10000, TS=4, LS=3, FN=ＭＳ ゴシック, LF=12"),  // window caption
97 :                              WS_CHILD,                              // window style
98 :                              0, 0, r.right, r.bottom,               // position, size
99 :                              hwnd, (HMENU)IDC_VTH, hInst, NULL);    // parent, menu, instance, param
100 :
101 :     SAjxTip::Add       (hWndVth, TipMsg);   //  ツールチップ設定
102 :     scp_vth.Attach(hWndVth);                //  VT100 ウインドハンドル割り当て
103 :     scp_vth.PutText(TipMsg); scp_vth.PutText(TEXT("¥n¥n"));
104 :     scp_vth.SetTitleText(TEXT(" Closed "));
105 :     scp_vth.EnablePopupMenu(FALSE);         //  VT100 のポップアップメニュー禁止(親へ WM_RBUTOONDOWN/UP を送る)
106 :     ShowWindow(hWndVth, SW_SHOW);           //  VT100 ウインド表示
107 :     return 0;
108 : }
109 : //----- WM_DESTROY -----------------------------------------------------------------------------------------//
110 : AJC_WNDPROC(Main, WM_DESTROY    )
111 : {
112 :     DestroyWindow(hWndVth);
113 :     PostQuitMessage(0);
114 :     return 0;
115 : }
116 : //----- WM_SIZE --------------------------------------------------------------------------------------------//
117 : AJC_WNDPROC(Main, WM_SIZE       )
118 : {
119 :     MoveWindow(hWndVth, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
120 :     return 0;
121 : }
122 : //----- WM_RBUTTONDOWN -------------------------------------------------------------------------------------//
123 : AJC_WNDPROC(Main, WM_RBUTTONDOWN)
124 : {
125 :     AJCPPMTBL   tbl[] = {{0, MID_SETPORT, 0, TEXT("ポート設定")},
126 :                          {0, MID_OPEN   , 0, TEXT("オープン／クローズ")},
127 :                          {AJCPPMF_EOT}};
128 :     POINT       pt;
129 :     GetCursorPos(&pt);
130 :     AjcPopupMenu(hwnd, WM_MYPOPUP, pt.x, pt.y, tbl, 0);
131 :     return 0;
132 : }
133 : //----- WM_MYPOPUP -----------------------------------------------------------------------------------------//
134 : AJC_WNDPROC(Main, WM_MYPOPUP    )
135 : {
136 :     if (wParam == MID_SETPORT) scp_vth.DlgParamEasy(hWndMain);
137 :     else {
138 :         if (scp_vth.IsOpened()) scp_vth.Close();
139 :         else                    scp_vth.Open();
140 :     }
141 :     return 0;
142 : }
143 : //--------------------------------------------------------------------------------------------------------//
144 : AJC_WNDMAP_DEF(Main)
145 :     AJC_WNDMAP_MSG(Main, WM_CREATE      )
146 :     AJC_WNDMAP_MSG(Main, WM_DESTROY     )
147 :     AJC_WNDMAP_MSG(Main, WM_SIZE        )
148 :     AJC_WNDMAP_MSG(Main, WM_RBUTTONDOWN )
149 :     AJC_WNDMAP_MSG(Main, WM_MYPOPUP     )
150 : AJC_WNDMAP_END
```

使用例２（コンソールアプリ）

最初に通信ポートの設定を行い、以降、受信したデータをテキスト表示します。
いずれかのキーを押すとプログラムを表示します。




```
 1 : //
 2 : //  SP_SCP2.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : #include    <conio.h>
 7 : using namespace AjxControl;
 8 :
 9 : //----- CAjxScp の派生クラス ----------------------------------------
10 : class CAjxScpEx : public CAjxScp
11 : {
12 : public:
13 :     CAjxScpEx() {                                          //  コンストラクタ
14 :         Init(TEXT("MyScpSect"), AJCSCP_CM_TEXT);
15 :     }
16 :     VO  OnNtcPortState(C_UTP pPortName, UI Param) override {    //  ポート状態通知
17 :         SAjxCon::PrintF(TEXT("ポート名：%s¥n"), pPortName);
18 :         switch (Param) {
19 :             case AJCSCP_CLOSED:     SAjxCon::PrintF(TEXT("- ポートをクローズしました。¥n¥n"));      break;
20 :             case AJCSCP_OPENED:     SAjxCon::PrintF(TEXT("- ポートをオープンしました。¥n¥n"));      break;
21 :             case AJCSCP_OPENFAIL:   SAjxCon::PrintF(TEXT("- ポートのオープンを失敗しました¥n¥n"));  break;
22 :             case AJCSCP_PORTCHG:    SAjxCon::PrintF(TEXT("- 通信ポートが変更されました¥n¥n"));      break;
23 :         }
24 :     }
25 :     VO  OnNtcRxTextChunk(C_UTP pText) override {               //  テキストチャンク受信通知
26 :         SAjxCon::PrintF(pText);
27 :     }
28 : };
29 : static CAjxScpEx    scp;
30 :
31 :
32 : int  AjcMain(int argc, UTP argv[])
33 : {
34 :     SAjxCon::SetStdMode();
35 :
36 :     //----- コマンドガイド表示 ----------------------------------------------------------------------------//
37 :     SAjxCon::PrintF(TEXT("¥n"));
38 :     SAjxCon::PrintF(TEXT(" 通信ポートから受信したテキストデータを表示します。¥n"));
39 :     SAjxCon::PrintF(TEXT(" いずれかのキーを押すとプログラムを終了します。¥n¥n"));
40 :
41 :     //  イベントマスク設定
42 :     scp.SetEvtMask(AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXCHUNK);
43 :     //  ポートパラメタ設定
44 :     UT WndTxt[256];
45 :     GetConsoleTitle(WndTxt, 256);
46 :     scp.DlgParamEasy(FindWindow(NULL, WndTxt));
47 :     //  ポートオープン
48 :     scp.Open();
49 :
50 :     while (!_kbhit()) {
51 :         //  ＳＣＰイベント待ち
52 :         scp.WaitEvent(100);
53 :     }
54 :
55 :     //  ポートクローズ
56 :     scp.Close();
57 :
58 :     return 0;
59 : }
60 :
```

2.9. ソケット(TCP/IP)サーバ機能

2.9.1. サーバー処理(CAjxSsvSvr)

メンバ変数

#	変数／関数形式	内容
1	HAJCSSV m_hSsv;	ＳＳＶインスタンスドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxSsvSvr();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	#ifdef _CONSOLE // コンソールアプリ BOOL Start (C_BCP pPort, UI MaxClients = 10, BOOL fUseWaitEvent = **TRUE** , int AddressFamily = AF_INET); #else // Windows アプリ BOOL Start (C_BCP pPort, UI MaxClients = 10, BOOL fUseWaitEvent = **FALSE**, int AddressFamily = AF_INET); #endif	サーバ開始 pPort ：ポート番号／サービス名文字列 MaxClients ：クライアント最大接続数 fUseWaitEvent：WaitEvent()関数使用の可否 AddressFamily：アドレスファミリ(AF_INET/ AF_INET6) **※ 他の関数に先駆けて、最初に実行します。** ※ 内部で、以下の関数を実行します。 ・AjcSsvCreate() ・AjcSsvStart ()
2	BOOL Stop ();	サーバ停止
3	BOOL SetOpt (AJCSSV_SERVOPT opt);	オプションの設定
4	AJCSSV_SERVOPT GetOpt ();	オプションの取得
5	BOOL SetChunkMode (AJCSSV_CHUNKMODE ChunkMode);	チャンクデータの通知モード設定
6	AJCSSV_CHUNKMODE GetChunkMode ();	チャンクデータの通知モード取得
7	BOOL SetEvtMask (UI EvtMsk);	イベントマスク設定
8	UI GetEvtMask ();	イベントマスク取得
9	BOOL SetRxTextCode (AJCSSV_TEXTCODE code);	受信テキストの文字コード設定
10	AJCSSV_TEXTCODE GetRxTextCode ();	受信テキストの文字コード取得
11	BOOL SetTxTextCode (AJCSSV_TEXTCODE code);	送信テキストの文字コード設定
12	AJCSSV_TEXTCODE GetTxTextCode ();	送信テキストの文字コード取得
13	BOOL WaitEvent (UI msTime);	イベント発生待ち
14	BOOL SetPktCtrlCode (UI stx, UI etx, UI dle);	パケットフレームを認識する為の制御コード設定
15	BOOL GetPktCtrlCode (UIP pStx, UIP pEtx, UIP pDle);	パケットフレームを認識する為の制御コード取得
16	BOOL SetPktTimeout (UI msTime);	パケットフレーム受信タイムアウト値設定
17	BOOL GetPktTimeout (UIP pMsTime);	パケットフレーム受信タイムアウト値取得
18	UI GetClientCount ();	接続済のクライアント数取得
19	UI GetConnectCount ();	通算接続回数取得
20	UI EnumClients ();	接続済の全クライアント列挙
21	BOOL SetClientData (HAJCSSVCLI hCli, UX data);	クライアントにデータを関連付ける
22	UX GetClientData (HAJCSSVCLI hCli);	クライアントに関連付けたデータ取得

仮想関数（サーバイベント）

#	変数／関数形式	内容	
1	virtual VO OnNtcStart ();	サーバ開始通知	AJCSSV_EV_START
2	virtual VO OnNtcStop ();	サーバ停止通知	AJCSSV_EV_STOP
3	virtual VO OnNtcError (UI err);	エラー通知	AJCSSV_EV_ERR
4	virtual BOOL OnNtcClients (HAJCSSVCLI hCli, CAjxSsvCli* pCli);	クライアント列挙 (pCli は、OnNtcConnect()で返された クライアント処理クラスへのポインタ)	cbNtcClients()

※ AJCSSV_EV_*XXXXX*は、対応するイベントコードを、*cbXXXXX*()は対応するコールバックを示します。

仮想関数（サーバ側でのクライアントイベント）

#	変数／関数形式	内容		
1	virtual CAjxSsvCli* OnNtcConnect(HAJCSSVCLI hCli);	クライアント接続通知	AJCSSV_EV_CONNECT	※１
2	virtual VO OnNtcDisconnect (HAJCSSVCLI hCli, CAjxSsvCli* pCli);	クライアント切断通知	AJCSSV_EV_DISCONNECT	※２
3	virtual VO OnNtcRxTextChunk(HAJCSSVCLI hCli, C_UTP pText);	テキストチャンク受信通知	AJCSSV_EV_RXCHUNK	※３
4	virtual VO OnNtcRxBinChunk (HAJCSSVCLI hCli, C_VOP pData, UI Bytes);	バイナリチャンク受信通知		
5	virtual VO OnNtcRxText (HAJCSSVCLI hCli, C_UTP pText);	テキスト受信通知	AJCSSV_EV_RXTEXT	
6	virtual VO OnNtcRxEsc (HAJCSSVCLI hCli, C_UTP pData);	ＥＳＣシーケンス受信通知	AJCSSV_EV_RXESC	
7	virtual VO OnNtcRxCtrl (HAJCSSVCLI hCli, UI Ctrl);	制御コード受信通知	AJCSSV_EV_RXCTRL	
8	virtual VO OnNtcRxPkt (HAJCSSVCLI hCli, C_VOP pData, UI Bytes);	パケット受信通知	AJCSSV_EV_RXPKT	
9	virtual VO OnNtcTxEmpty (HAJCSSVCLI hCli);	送信完了通知	AJCSSV_EV_TXEMPTY	
10	virtual VO OnNtcRxNoPkt (HAJCSSVCLI hCli, C_UTP pText);	パケット外データ受信通知	AJCSSV_EV_RXNOPKT	
11	virtual VO OnNtcInvChunk (HAJCSSVCLI hCli, C_VOP pData, UI lData);	不正テキストチャンク受信通知	AJCSSV_EV_INVCHUNK	
12	virtual VO OnNtcRxError (HAJCSSVCLI hCli, UI error);	受信エラー通知	AJCSSV_EV_RXERR	
13	virtual VO OnNtcTxError (HAJCSSVCLI hCli, UI error);	送信エラー通知	AJCSSV_EV_TXERR	

※ AJCSSV_EV_*XXXXX*は、対応するイベントコードを示します。

※１：クライアント処理クラス（AjxSsvCli クラス）の派生クラスを生成し、クラスへのポインタを返すと、当該クライアント処理クラスを連結します。
この場合、上記＃３以降のイベントが発生したら、連結したクライアント処理クラスの 対応する仮想関数が呼び出されます。（使用例の ① 参照）
OnNtcConnect()で NULL を返した場合は、当該クライアント処理クラスは連結されません。
OnNtcConnect()をオーバーライドする場合、派生元の仮想関数を呼び出す必要はありません。（CAjxSsvSvr:: OnNtcConnect()は、単に NULL を返します）

※２：クライアント処理クラス（AjxSsvCli クラス）の派生クラスを生成した場合、このイベントで当該クライアント処理クラスを削除(delete)してください。
pCli 引数は、OnNtcConnect()関数で返された値を示します。（使用例の ② 参照）
OnNtcDisconnect()をオーバーライドする場合、派生元の仮想関数を呼び出す必要はありません。（CAjxSsvSvr:: OnNtcDisconnect()は、何もしません）

※３：これらの仮想関数は、通常はオーバーライドする必要はありません。
これらの仮想関数は、OnNtcConnect()で、クライアント処理クラスへのポインタが返された場合、当該クライアントクラスの対応する仮想関数を呼び出す処理が実装されています。

2.9.2. クライアント処理(CAjxSsvSvr)

サーバー側で、個別のクライアントとの通信を行う処理クラスです。
このクラスの派生クラスを生成し、アプリケーション独自の処理を記述することができます。

コンストラクタ

#	変数／関数形式	内容
1	CAjxSsvCli();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	VO SetCliHandle (HAJCSSVCLI hCli);	クライアントハンドル設定 ※ 他の関数に先駆けて最初に実行する必要があります。 ※ hCli は、CAjxSsvSvr クラスの OnNtcConnect()で通知された値を指定
2	C_UTP GetIpAddrStr (UTP pBuf, UI lBuf);	ＩＰアドレス文字列取得
3	AJCSSV_TEXTCODE GeActualtRxTextCode();	実際の受信テキストの文字コード取得
4	AJCSSV_TEXTCODE GetActualTxTextCode();	実際の送信テキストの文字コード取得
5	BOOL SendChar (UT code);	クライアントへ１文字送信
6	BOOL SendText (C_UTP pTxt, UI lTxt = -1);	クライアントへテキストデータ送信
7	BOOL SendTextF (C_UTP pFmt, ...);	クライアントへ書式テキスト送信
8	BOOL SendBinData (C_VOP pDat, UI lDat);	クライアントへバイナリデータ送信
9	UI SendPacket (C_VOP pPkt, UI lPkt);	クライアントへパケットデータ送信
10	BOOL PurgeRecvData ();	全受信済データ破棄
11	BOOL PurgeSendData ();	全送信待ちデータ破棄
12	BOOL PurgeAllData ();	全受信済データと送信待ちデータ破棄
13	BOOL SetClientData (UX data);	クライアントにデータを関連付ける
14	UX GetClientData ();	クライアントに関連付けたデータ取得
15	UI GetSeqNo ();	接続順序番号取得
16	UI GetIndex ();	クライアントに割り当てられたインデクス値取得
17	BOOL Disconnect ();	回線切断

仮想関数（サーバ側でのクライアントイベント）

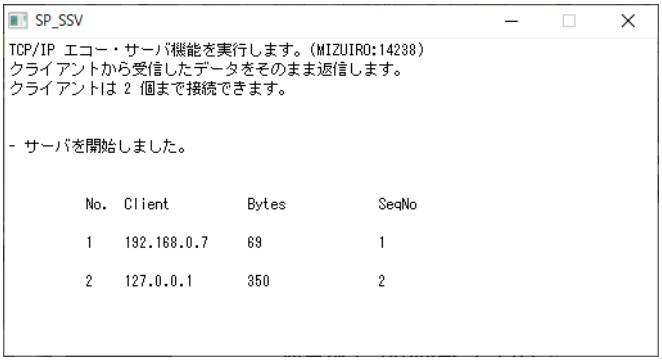
#	変数／関数形式	内容	
1	virtual VO OnNtcConnect ();	クライアント接続通知	AJCSSV_EV_CONNECT
2	virtual VO OnNtcDisconnect (CAjxSsvCli* pCli);	クライアント切断通知	AJCSSV_EV_DISCONNECT
3	virtual VO OnNtcRxTextChunk(C_UTP pText);	テキストチャンク受信通知	AJCSSV_EV_RXCHUNK
4	virtual VO OnNtcRxBinChunk (C_VOP pData, UI Bytes);	バイナリチャンク受信通知	
5	virtual VO OnNtcRxText (C_UTP pText);	テキスト受信通知	AJCSSV_EV_RXTEXT
6	virtual VO OnNtcRxEsc (C_UTP pData);	ＥＳＣシーケンス受信通知	AJCSSV_EV_RXESC
7	virtual VO OnNtcRxCtrl (UI Ctrl);	制御コード受信通知	AJCSSV_EV_RXCTRL
8	virtual VO OnNtcRxPkt (C_VOP pData, UI Bytes);	パケット受信通知	AJCSSV_EV_RXPKT
9	virtual VO OnNtcTxEmpty ();	送信完了通知	AJCSSV_EV_TXEMPTY
10	virtual VO OnNtcRxNoPkt (C_UTP pText);	パケット外データ受信通知	AJCSSV_EV_RXNOPKT
11	virtual VO OnNtcInvChunk (C_VOP pData, UI lData);	不正テキストチャンク受信通知	AJCSSV_EV_INVCHUNK
12	virtual VO OnNtcRxError (UI error);	受信エラー通知	AJCSSV_EV_RXERR
13	virtual VO OnNtcTxError (UI error);	送信エラー通知	AJCSSV_EV_TXERR

※ AJCSSV_EV_*XXXXX*は、対応するイベントコードを示します。

※上記仮想関数は全て、CAjxSsvSvr クラスを継承したクラスの OnNtcConnect()の戻り値に、当該継承クラスのポインタを返した場合に起動されます。

使用例１（Windows アプリ）

このサンプルプログラムは、最大２つのクライアントと接続し、
クライアントから受信したデータをそのまま返信します。
画面には、接続したクライアントのＩＰアドレスと、受信バイト数
を表示します。



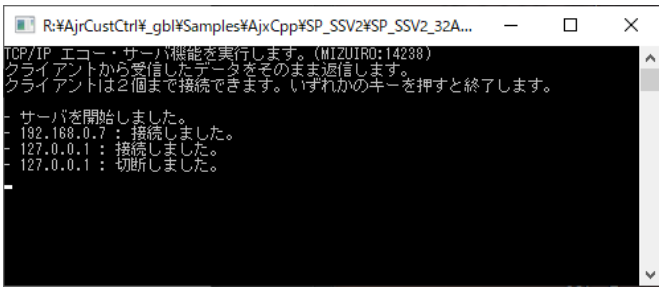

```
 1 : //
 2 : //  SP_SSV1.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : using namespace AjxControl;
 7 :
 8 : #define IDC_VTH     5001
 9 : #define MY_WS       (WS_OVERLAPPEDWINDOW & ~(WS_THICKFRAME | WS_MAXIMIZEBOX))
10 :
11 : static  HINSTANCE   hInst     = NULL;
12 : static  HWND        hWndMain  = NULL;
13 : static  HWND        hWndVth   = NULL;
14 :
15 : AJC_WNDPROC_DEF(Main);
16 : //----- クライアント制御テーブル ----------------------------------------------------------------------------//
17 : #define MAX_CLITBL  2       /*  最大クライアント数  */
18 : typedef struct {
19 :     HAJCSSVCLI  hCli;
20 :     UX          Bytes;
21 :     UI          SeqNo;
22 :     UI          line;
23 : } CLITBL, *PCLITBL;
24 :
25 : //----- 個々のクライアント処理クラス (CAjxSsvCli の派生クラス) ----------------------------------------------//
26 : class CAjxSsvCliEx : public CAjxSsvCli, public CAjxVth {
27 :     HAJCSSVCLI  m_hCli;
28 :     UI          m_line;
29 :     ULL         m_Bytes;
30 :     UI          m_SeqNo;
31 : public:
32 :     //  コンストラクタ
33 :     CAjxSsvCliEx(HAJCSSVCLI hCli = NULL)    {
34 :         CAjxSsvCli::SetCliHandle(m_hCli = hCli);
35 :         Attach(hWndVth);
36 :         m_line =  0;
37 :         m_Bytes = 0;
38 :         m_SeqNo = GetSeqNo();
39 :     }
40 :     //  クライアント接続通知
41 :     VO  OnNtcConnect    ()  override {
42 :         UT  szIpAddr[64];
43 :         UI  ix = GetIndex();
44 :         m_line = 10 + ix * 2;
45 :         GetIpAddrStr(szIpAddr, 64);
46 :         Locate(m_line, 15); PrintF(szIpAddr);
47 :         Locate(m_line, 31); PrintF(TEXT("%d"), m_Bytes);
48 :         Locate(m_line, 48); PrintF(TEXT("%d"), m_SeqNo);
49 :     }
50 :     //  クライアント切断通知
51 :     VO  OnNtcDisconnect ()      override {
52 :         UT  szIpAddr[64];
53 :         UI  ix = GetIndex();
54 :         Locate(m_line, 15); PrintF(TEXT("¥x1b[K"));
55 :         GetIpAddrStr(szIpAddr, 64);
56 :         Locate(m_line, 15); PrintF(TEXT("-"));
57 :         Locate(m_line, 31); PrintF(TEXT("-"));
58 :         Locate(m_line, 48); PrintF(TEXT("-"));
59 :     }
60 :     //  バイナリチャンク受信通知
61 :     VO  OnNtcRxBinChunk (C_VOP pData, UI Bytes)     override {
62 :         UI  ix = GetIndex();
63 :         SendBinData(pData, Bytes);
64 :         m_Bytes += Bytes;
```

```
65 :         Locate(m_line, 31); PrintF(TEXT("%lld"), m_Bytes);
66 :     }
67 : };
68 :
69 : //----- サーバ処理クラス (CAjxSsvSvr + CAjxVth の派生クラス) ------------------------------------------------//
70 : class CAjxSsvEx : public CAjxSsvSvr, public CAjxVth {
71 :     CLITBL      m_CliTbl[MAX_CLITBL];
72 : public:
73 :     //----- サーバイベント ------------------------------------------------------------------------//
74 :     //  サーバ開始通知
75 :     VO      OnNtcStart              ()          override {
76 :         Locate(5, 0);
77 :         PrintF(TEXT("¥r¥x1b[2K"));
78 :         PrintF(TEXT("- サーバを開始しました。¥n"));
79 :         //  クライアント制御テーブル初期化
80 :         int     line = 10;
81 :         memset(m_CliTbl, 0, sizeof m_CliTbl);
82 :         for (int i = 0; i < MAX_CLITBL; i++) {
83 :             m_CliTbl[i].line = line;
84 :             line += 2;
85 :         }
86 :     }
87 :     //  サーバ停止通知
88 :     VO      OnNtcStop               ()          override {
89 :         Locate(14, 0);
90 :         PrintF(TEXT("¥r¥x1b[2K"));
91 :         PrintF(TEXT("- サーバを停止しました。３秒後にプログラムを終了します。¥n"));
92 :         SetTimer(hWndMain, 1, 3000, NULL);
93 :     }
94 :     //  エラー通知
95 :     VO      OnNtcError              (UI err)    override {
96 :         Locate(6, 0);
97 :         PrintF(TEXT("¥r¥x1b[2K"));
98 :         PrintF(TEXT("¥x1b[31m- エラー発生(%d)¥x1b[0m¥n"), err);
99 :     }
100 :    //----- クライアントイベント ------------------------------------------------------------------//
101 :    //  クライアント接続通知
102 :    CAjxSsvCli* OnNtcConnect    (HAJCSSVCLI hCli)                   override {
103 :        //  クライアント処理クラス生成
104 :        CAjxSsvCliEx* pCli = new CAjxSsvCliEx(hCli);
105 :        return pCli;
106 :    }
107 :    //  クライアント切断通知
108 :    VO  OnNtcDisconnect (HAJCSSVCLI hCli, CAjxSsvCli* pCli) override {
109 :        //  クライアント処理クラス消去
110 :        delete (CAjxSsvCliEx*)pCli;
111 :    }
112 : };
113 : static  CAjxSsvEx   ssv_vth;
114 :
115 : //----- WinMain -------------------------------------------------------------------------------------//
116 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
117 : {
118 :     MSG         msg = {0};
119 :     WNDCLASS    wc  = {0};
120 :     ATOM        atm = 0;
121 :
122 :     hInst = hInstance;
123 :
124 :     //  ウインド生成
125 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
126 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
127 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_SSV");
128 :     atm = RegisterClass(&wc);
129 :     hWndMain = CreateWindowEx(0,
130 :                 TEXT("SP_SSV"), TEXT("SP_SSV"),             // window class name, caption
131 :                 MY_WS,                                      // window style
132 :                 CW_USEDEFAULT, CW_USEDEFAULT, 800, 400,     // position, size
133 :                 NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
134 :     ShowWindow(hWndMain, iCmdShow);
135 :     //  メッセージループ
136 :     while (GetMessage(&msg, NULL, 0, 0)) {
137 :         TranslateMessage(&msg);
138 :         DispatchMessage (&msg);
139 :     }
140 :     UnregisterClass((UTP)atm, hInstance);
141 :     return (int)msg.wParam ;
142 : }
143 : //----- WM_CREATE -----------------------------------------------------------------------------------//
144 : AJC_WNDPROC(Main, WM_CREATE     )
145 : {
```

```
146 :     //  ＶＴ１００ウインド生成
147 :     RECT    r;
148 :     GetClientRect(hwnd, &r);
149 :     hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,
150 :                              TEXT("AjcCtrlVT100"),                  // window class
151 :                              TEXT("P: VW=80, VH=16, TS=4, LS=3, FN=ＭＳ ゴシック, LF=12"),  // window caption
152 :                              WS_CHILD | AJCVTHS_NOBORDER,           // window style
153 :                              0, 0, r.right, r.bottom,               // position, size
154 :                              hwnd, (HMENU)IDC_VTH, hInst, NULL);    // parent, menu, instance, param
155 :     //  VT100 ウインドハンドル割り当て
156 :     ssv_vth.Attach(hWndVth);
157 :     //  VT100 ウインドサイズ設定
158 :     UI      w, h;
159 :     ssv_vth.GetVramFitSize(&w, &h);
160 :     GetWindowRect(hwnd, &r);
161 :     r.right  = r.left + w;
162 :     r.bottom = r.top  + h;
163 :     AdjustWindowRect(&r, MY_WS, FALSE);
164 :     SetWindowPos(hwnd, NULL, r.left, r.top, r.right - r.left, r.bottom - r.top, SWP_NOZORDER);
165 :     //  ガイドメッセージ表示
166 :     DWORD   bytes;
167 :     UT      szHostName[256];
168 :     bytes = (sizeof szHostName);
169 :     GetComputerName(szHostName, &bytes);
170 :     ssv_vth.PrintF(TEXT("TCP/IP エコー・サーバ機能を実行します。(%s:14238)¥n")
171 :                    TEXT("クライアントから受信したデータをそのまま返信します。¥n")
172 :                    TEXT("クライアントは %d 個まで接続できます。¥n"), szHostName, MAX_CLITBL);
173 :     //  タイトル表示
174 :     ssv_vth.Locate( 8, 10); ssv_vth.PrintF(TEXT("No.   Client          Bytes           SeqNo"));
175 :     ssv_vth.Locate(10, 10); ssv_vth.PrintF(TEXT("1     -               -               -"));
176 :     ssv_vth.Locate(12, 10); ssv_vth.PrintF(TEXT("2     -               -               -"));
177 :     //  VT100 キャレット非表示
178 :     ssv_vth.ShowCaret(FALSE);
179 :     //  VT100 ウインド表示
180 :     ShowWindow(hWndVth, SW_SHOW);           //  VT100 ウインド表示
181 :     //  サーバ開始
182 :     ssv_vth.Start   (TEXT("14238"), MAX_CLITBL);
183 :     ssv_vth.SetEvtMask(AJCSSV_EV_GENERAL | AJCSSV_EV_RXCHUNK | AJCSSV_EV_ERR);
184 :     ssv_vth.SetChunkMode(AJCSSV_CM_BIN);
185 :     return 0;
186 : }
187 : //----- WM_CLOSE ---------------------------------------------------------------------------------------------//
188 : AJC_WNDPROC(Main, WM_CLOSE      )
189 : {
190 :     ssv_vth.Stop();
191 :     return 0;
192 : }
193 : //----- WM_DESTROY -------------------------------------------------------------------------------------------//
194 : AJC_WNDPROC(Main, WM_DESTROY    )
195 : {
196 :     PostQuitMessage(0);
197 :     return 0;
198 : }
199 : //----- WM_SIZE ----------------------------------------------------------------------------------------------//
200 : AJC_WNDPROC(Main, WM_SIZE       )
201 : {
202 :     MoveWindow(hWndVth, 0, 0, LOWORD(lParam), HIWORD(lParam), FALSE);
203 :     return 0;
204 : }
205 : //----- WM_TIMER ---------------------------------------------------------------------------------------------//
206 : AJC_WNDPROC(Main, WM_TIMER      )
207 : {
208 :     DestroyWindow(hwnd);
209 :     return 0;
210 : }
211 : //------------------------------------------------------------------------------------------------------------//
212 : AJC_WNDMAP_DEF(Main)
213 :     AJC_WNDMAP_MSG(Main, WM_CREATE      )
214 :     AJC_WNDMAP_MSG(Main, WM_CLOSE       )
215 :     AJC_WNDMAP_MSG(Main, WM_DESTROY     )
216 :     AJC_WNDMAP_MSG(Main, WM_SIZE        )
217 :     AJC_WNDMAP_MSG(Main, WM_TIMER       )
218 : AJC_WNDMAP_END
```

使用例２（コンソールアプリ）

このサンプルプログラムは、最大２つのクライアントと接続し、
クライアントから受信したデータをそのまま返信します。
サーバの開始／停止と、接続したクライアントの接続／切断を
ログ表示します。




```
  1 : //
  2 : //  SP_SSV2.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <stdio.h>
  6 : #include    <conio.h>
  7 : #include    <time.h>
  8 : #include    <tchar.h>
  9 : using namespace AjxControl;
 10 :
 11 : //----- ワーク ---------------------------------------------------------------------------------------------//
 12 : static  BOOL    fServEnd = FALSE;       //  サーバ停止フラグ
 13 :
 14 : //----- 個々のクライアント処理クラス（CAjxSsvCli の派生クラス） ---------------------------------------------//
 15 : class CAjxSsvCliEx : public CAjxSsvCli {
 16 :     HAJCSSVCLI  m_hCli;
 17 :     UI          m_line;
 18 :     ULL         m_Bytes;
 19 :     UI          m_SeqNo;
 20 : public:
 21 :     //  コンストラクタ
 22 :     CAjxSsvCliEx(HAJCSSVCLI hCli = NULL)    {
 23 :         CAjxSsvCli::SetCliHandle(m_hCli = hCli);
 24 :         m_line =  0;
 25 :         m_Bytes = 0;
 26 :         m_SeqNo = GetSeqNo();
 27 :     }
 28 :     //  クライアント接続通知
 29 :     VO  OnNtcConnect    ()  override {
 30 :         UT  szAddr[128];
 31 :         GetIpAddrStr(szAddr, 128);
 32 :         SAjxCon::PrintF(TEXT("- %-16s : 接続しました。¥n"), szAddr);
 33 :     }
 34 :     //  クライアント切断通知
 35 :     VO  OnNtcDisconnect ()      override {
 36 :         UT  szAddr[128];
 37 :         GetIpAddrStr(szAddr, 128);
 38 :         SAjxCon::PrintF(TEXT("- %-16s : 切断しました。¥n"), szAddr);
 39 :     }
 40 :     //  バイナリチャンク受信通知
 41 :     VO  OnNtcRxBinChunk (C_VOP pData, UI Bytes)     override {
 42 :         SendBinData(pData, Bytes); // そのまま返信
 43 :     }
 44 : };
 45 : //----- サーバ処理クラス（CAjxSsvSvr の派生クラス） ---------------------------------------------------------//
 46 : class CAjxSsvEx : public CAjxSsvSvr {
 47 : public:
 48 :     //----- サーバイベント ------------------------------------------------------------------------//
 49 :     //  サーバ開始通知
 50 :     VO      OnNtcStart              ()          override {
 51 :         SAjxCon::PrintF(TEXT("- サーバを開始しました。¥n"));
 52 :     }
 53 :     //  サーバ停止通知
 54 :     VO      OnNtcStop               ()          override {
 55 :         SAjxCon::PrintF(TEXT("- サーバを停止しました。¥n"));
 56 :         fServEnd = TRUE;
 57 :     }
 58 :     //  エラー通知
 59 :     VO      OnNtcError              (UI err)    override {
 60 :         SAjxCon::PrintF(TEXT("*** エラー発生(%d) ***¥n"), err);
 61 :     }
 62 :     //  クライアント列挙通知
 63 :     BOOL        OnNtcClients    (HAJCSSVCLI hCli, CAjxSsvCli* pCli)
 64 :     {
 65 :         pCli->Disconnect();
 66 :         return TRUE;  // TRUE:列挙継続
 67 :     }
```

```
68 :     //----- クライアントイベント ------------------------------------------------------------------//
69 :     //  クライアント接続通知
70 :     CAjxSsvCli* OnNtcConnect    (HAJCSSVCLI hCli)                       override {
71 :         //  クライアント処理クラス生成
72 :         CAjxSsvCliEx* pCli = new CAjxSsvCliEx(hCli);
73 :         return pCli;
74 :     }
75 :     //  クライアント切断通知
76 :     VO          OnNtcDisconnect (HAJCSSVCLI hCli, CAjxSsvCli* pCli)     override {
77 :         //  クライアント処理クラス消去
78 :         delete (CAjxSsvCliEx*)pCli;
79 :     }
80 : };
81 : static  CAjxSsvEx   ssv;
82 :
83 : //=============================================================================================================//
84 : int  AjcMain(int argc, UTP argv[])
85 : {
86 :     SAjxCon::SetStdMode();
87 :
88 :     //  ガイドメッセージ表示
89 :     DWORD   bytes;
90 :     UT      szHostName[256];
91 :     bytes = (sizeof szHostName);
92 :     GetComputerName(szHostName, &bytes);
93 :     SAjxCon::PrintF(TEXT("TCP/IP エコー・サーバ機能を実行します。(%s:14238)¥n")
94 :                     TEXT("クライアントから受信したデータをそのまま返信します。¥n")
95 :                     TEXT("クライアントは２個まで接続できます。")
96 :                     TEXT("いずれかのキーを押すと終了します。¥n¥n"), szHostName);
97 :
98 :     //  サーバ開始
99 :     ssv.Start   (TEXT("14238"), 2);
100 :    ssv.SetEvtMask(AJCSSV_EV_GENERAL | AJCSSV_EV_RXCHUNK | AJCSSV_EV_ERR);
101 :    ssv.SetChunkMode(AJCSSV_CM_BIN);
102 :
103 :    //  サーバ停止までイベント待ちループ
104 :    BOOL    fStop = FALSE;
105 :    while (!fServEnd) {
106 :        //  いずれかのキー押下でサーバ停止
107 :        if (_kbhit() && !fStop) {
108 :            ssv.EnumClients();  //  クライアント列挙 (接続中のクライアント切断)
109 :                                //  この処理は特に実行しなくてもＯＫです (サーバ停止で全クライアントは切断されます)
110 :
111 :            ssv.Stop();         //  サーバ停止
112 :            fStop = TRUE;
113 :        }
114 :        //  イベント発生待ち(100ms)
115 :        ssv.WaitEvent(100);
116 :    }
117 :
118 :    SAjxCon::PrintF(TEXT("Hit Enter Key!!"));
119 :    while (_kbhit()) _getch();
120 :    getchar();
121 :
122 :    return 0;
123 : }
124 :
```

2.10. ソケット(TCP/IP)クライアント機能 (CAjxSct)

コンストラクタ

#	変数／関数形式	内容
1	CAjxSct();	コンストラクタ

メンバ変数

#	変数／関数形式	内容
1	HAJCSCT m_hSct;	ＳＣＴインスタンスドハンドル

メンバ関数

#	変数／関数形式	内容
1	#ifdef _CONSOLE // コンソールアプリ BOOL Connect (C_UTP pServ, C_UTP pPort, int AddressFamily = AF_INET, BOOL fUseWaitEvent = **TRUE**); #else // Windows アプリ BOOL Connect (C_UTP pServ, C_UTP pPort, int AddressFamily = AF_INET, BOOL fUseWaitEvent = **FALSE**); #endif	回線接続 pServ --------- サーバ名／ＩＰアドレス pPort --------- ポート番号／サービス名 AddressFamily – アドレスファミリ(AF_INET / AF_INET6) fUseWaitEvent – WaitEvent()関数使用の可否
2	BOOL Disconnect (UI msTimeout = 10000);	回線切断
3	AJCSCT_STATE GetState ();	回線状態取得
4	BOOL SetChunkMode (AJCSCT_CHUNKMODE ChunkMode);	チャンクデータの受信モード設定
5	AJCSCT_CHUNKMODE GetChunkMode ();	チャンクデータの受信モード取得
6	BOOL SetEvtMask (UI EvtMsk);	イベントマスク設定
7	UI GetEvtMask ();	イベントマスク取得
8	BOOL SetRxTextCode (AJCSCT_TEXTCODE code);	受信テキストの文字コード設定
9	AJCSCT_TEXTCODE GetRxTextCode ();	受信テキストの文字コード取得
10	BOOL SetTxTextCode (AJCSCT_TEXTCODE code);	送信テキストの文字コード設定
11	AJCSCT_TEXTCODE GetTxTextCode ();	送信テキストの文字コード取得
12	BOOL WaitEvent (UI msTime);	イベント発生待ち
13	BOOL SetPktCtrlCode (UI stx = 0x02, UI etx = 0x02, UI dle = 0x10);	パケットフレームを認識する為の制御コード設定
14	BOOL GetPktCtrlCode (UIP pStx = NULL, UIP pEtx = NULL, UIP pDle = NULL);	パケットフレームを認識する為の制御コード取得
15	BOOL SetPktTimeout (UI msTime);	パケットフレーム受信タイムアウト値設定
16	BOOL GetPktTimeout (UIP pMsTime);	パケットフレーム受信タイムアウト値取得
17	BOOL SendChar (UT code);	１文字送信
18	BOOL SendText (C_UTP pTxt, UI lTxt = -1);	テキストデータ送信
19	BOOL SendTextF (C_UTP pFmt, ...);	書式テキスト送信
20	BOOL SendBinData (C_VOP pDat, UI lDat);	バイナリデータ送信
21	UI SendPacket (C_VOP pPkt, UI lPkt);	パケットデータ送信
22	BOOL PurgeRecvData ();	全受信済データ破棄
23	BOOL PurgeSendData ();	全送信待ちデータ破棄
24	BOOL PurgeAllData ();	全送受信データ破棄

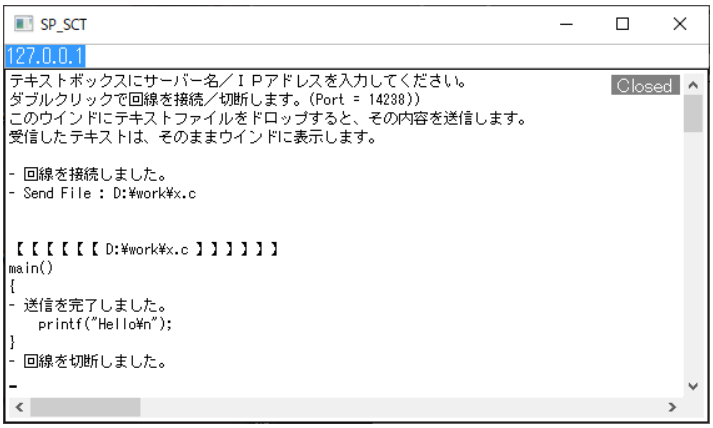
仮想関数（サーバ側でのクライアントイベント）

#	変数／関数形式	内容	
1	virtual VO OnNtcConnect ();	接続通知	
2	virtual VO OnNtcDisconnect ();	切断通知	
3	virtual VO OnNtcRxTextChunk(C_UTP pText);	テキストチャンク受信通知	
4	virtual VO OnNtcRxBinChunk (C_VOP pData, UI Bytes);	バイナリチャンク受信通知	
5	virtual VO OnNtcRxText (C_UTP pText);	テキスト受信通知	AJCSCT_EV_RXTEXT
6	virtual VO OnNtcRxEsc (C_UTP pData);	ＥＳＣシーケンス受信通知	AJCSCT_EV_RXESC
7	virtual VO OnNtcRxCtrl (UI Ctrl);	制御コード受信通知	AJCSCT_EV_RXCTRL
8	virtual VO OnNtcRxPkt (C_VOP pData, UI Bytes);	パケット受信通知	AJCSCT_EV_RXPKT
9	virtual VO OnNtcTxEmpty ();	送信完了通知	AJCSCT_EV_TXEMPTY
10	virtual VO OnNtcRxNoPkt (C_UTP pText);	パケット外データ受信通知	AJCSCT_EV_RXNOPKT
11	virtual VO OnNtcInvChunk (C_VOP pData, UI lData);	不正テキストチャンク受信通知	AJCSCT_EV_INVCHUNK
12	virtual VO OnNtcCnErr (UI param);	接続失敗通知	AJCSCT_EV_CNFAIL
13	virtual VO OnNtcRxErr (UI param);	受信エラー発生通知	AJCSCT_EV_RXERR
14	virtual VO OnNtcTxErr (UI param);	送信エラー発生通知	AJCSCT_EV_TXERR
15	virtual VO OnNtcErr (UI param);	その他のエラー発生通知	AJCSCT_EV_ERR

※ AJCSCT_EV_*XXXXX*は、対応するイベントコードを示します。

使用例１

サーバと接続し、ファイルの送信を行い、サーバから受信したテキストを表示します。
ダブルクリックで回線接続／切断を行います。
ファイルをドロップするとその内容を送信します。




```
  1 : //
  2 : //  SP_SCT1.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <tchar.h>
  6 : using namespace AjxControl;
  7 :
  8 : #define IDC_VTH     5001
  9 : #define WM_MYPOPUP  (WM_APP + 10)
 10 :
 11 : static  HINSTANCE   hInst    = NULL;
 12 : static  HWND        hWndMain = NULL;
 13 : static  HWND        hWndVth  = NULL;
 14 : static  HWND        hTxtServ = NULL;
 15 : static  BOOL        fConnect = FALSE;
 16 :
 17 : static const UT TipMsg[] =  TEXT("テキストボックスにサーバー名／ＩＰアドレスを入力してください。¥n")
 18 :                            TEXT("ダブルクリックで回線を接続／切断します。(Port = 14238))¥n")
 19 :                            TEXT("このウインドにテキストファイルをドロップすると、その内容を送信します。¥n")
 20 :                            TEXT("受信したテキストは、そのままウインドに表示します。");
 21 : AJC_WNDPROC_DEF(Main);
 22 : //----- CAjxSct + CAjxVth の派生クラス ------------------------------------------------------------------------//
 23 : class CAjxSctEx :
 24 :     public CAjxSct,
 25 :     public CAjxVth
 26 : {
 27 : public:
 28 :     VO  OnNtcConnect()      override {                      //  接続通知
 29 :         PrintF(TEXT("- 回線を接続しました。¥n"));
 30 :         fConnect = TRUE;
 31 :     }
 32 :     VO  OnNtcDisconnect()   override {                      //  切断通知
 33 :         PrintF(TEXT("- 回線を切断しました。¥n"));
 34 :         fConnect = FALSE;
 35 :     }
 36 :     VO  OnNtcCnErr(UI param) override {                     //  接続失敗通知
 37 :         PrintF(TEXT("- 回線の接続を失敗しました。(%d)¥n"), param);
 38 :     }
 39 :     VO  OnNtcRxTextChunk(C_UTP pText) override {            //  テキストチャンク受信通知
 40 :         PutText(pText);
 41 :     }
 42 :     VO  OnNtcDblClk     (UI flag)   override    {           //  ダブルクリック通知
 43 :         if (fConnect) {
 44 :             Disconnect();
 45 :         }
 46 :         else {
 47 :             UT  szServ[256];
 48 :             AjcGetCtrlStr(hTxtServ, szServ, 256);
 49 :             Connect (szServ, TEXT("14238"));
 50 :         }
 51 :     }
 52 :     VO  OnNtcDropFile (UI nFiles) override  {               //  ファイルドロップ通知
 53 :         CAjxFile f;
 54 :         UT  path[MAX_PATH];
 55 :         UT  buf[1024];
 56 :         while (GetDroppedFile(path)) {
 57 :             PrintF(TEXT("- Send File : %s¥n¥n"), path);
 58 :             SendTextF(TEXT("¥n【【【【【 %s 】】】】】¥n"), path);
 59 :             if (f.FOpen(path)) {
 60 :                 while (f.FGetS(buf, AJCTSIZE(buf))) {
 61 :                     SendText(buf);
 62 :                 }
 63 :                 f.FClose();
 64 :             }
 65 :         }
 66 :     }
 67 :     VO  OnNtcTxEmpty()      override {                      //  送信完了通知
 68 :         PrintF(TEXT("- 送信を完了しました。¥n"));
 69 :     }
 70 : };
 71 : static  CAjxSctEx   sct_vth;
 72 : //----- WinMain ----------------------------------------------------------------------------------------------//
 73 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
 74 : {
 75 :     MSG         msg = {0};
 76 :     WNDCLASS    wc  = {0};
 77 :     ATOM        atm = 0;
 78 :
 79 :     hInst = hInstance;
 80 :
 81 :     //  ウインド生成
 82 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
```

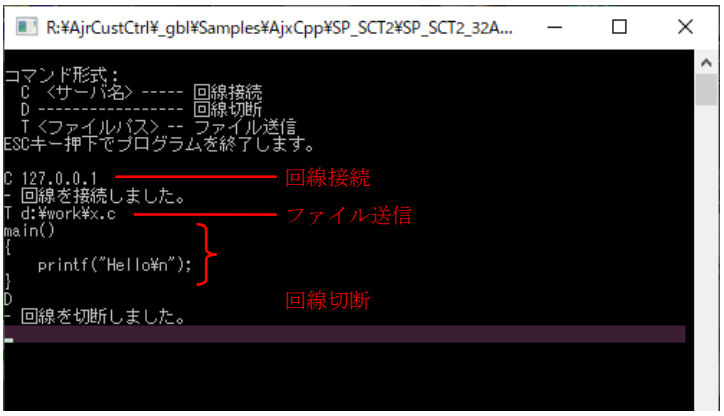
```
 83 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
 84 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_SCT");
 85 :     atm = RegisterClass(&wc);
 86 :     hWndMain = CreateWindowEx(0,
 87 :                 TEXT("SP_SCT"), TEXT("SP_SCT"),             // window class name, caption
 88 :                 WS_OVERLAPPEDWINDOW,                        // window style
 89 :                 CW_USEDEFAULT, CW_USEDEFAULT, 800, 400,     // position, size
 90 :                 NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
 91 :     ShowWindow(hWndMain, iCmdShow);
 92 :     SetFocus(hTxtServ);
 93 :     //  メッセージループ
 94 :     while (GetMessage(&msg, NULL, 0, 0)) {
 95 :         TranslateMessage(&msg);
 96 :         DispatchMessage (&msg);
 97 :     }
 98 :     UnregisterClass((UTP)atm, hInstance);
 99 :     return (int)msg.wParam ;
100 : }
101 : //----- WM_CREATE ----------------------------------------------------------------------------------------------//
102 : AJC_WNDPROC(Main, WM_CREATE     )
103 : {
104 :     //  ＶＴ１００ウインド生成
105 :     RECT    r;
106 :     GetClientRect(hwnd, &r);
107 :     GetClientRect(hwnd, &r);
108 :     hWndVth = CreateWindowEx(WS_EX_ACCEPTFILES,
109 :                              TEXT("AjcCtrlVT100"),                  // window class
110 :                              TEXT("P: VW=512, VH=128, ML=10000, TS=4, LS=3, FN=ＭＳ ゴシック, LF=12"),  // window caption
111 :                              WS_CHILD,                              // window style
112 :                              0, 20, r.right, r.bottom - 20,         // position, size
113 :                              hwnd, (HMENU)IDC_VTH, hInst, NULL);    // parent, menu, instance, param
114 :     ShowWindow(hWndVth, SW_SHOW);           //  VT100 表示
115 :     SAjxTip::Add       (hWndVth, TipMsg);   //  ツールチップ設定
116 :     sct_vth.Attach(hWndVth);                //  VT100 ウインドハンドル割り当て
117 :     sct_vth.PutText(TipMsg); sct_vth.PutText(TEXT("¥n¥n"));
118 :     sct_vth.SetTitleText(TEXT(" Closed "));
119 :     //  テキストボックス生成
120 :     hTxtServ = CreateWindow( TEXT("Edit"),                          // window class
121 :                              TEXT("127.0.0.1"),                     // window caption
122 :                              WS_CHILD | ES_NOHIDESEL,               // window style
123 :                              1, 1, r.right - 2, 18,                 // position, size
124 :                              hwnd, (HMENU)IDC_VTH, hInst, NULL);    // parent, menu, instance, param
125 :     SAjxCtrl::CtrlLoadTextBox(hTxtServ, TEXT("MySect"));
126 :     SendMessage(hTxtServ, EM_SETSEL, 0, -1);
127 :     ShowWindow (hTxtServ, SW_SHOW);         //  テキストボックス表示
128 :     //  イベントマスク設定
129 :     sct_vth.SetEvtMask  (AJCSCT_EV_GENERAL | AJCSCT_EV_RXCHUNK | AJCSCT_EV_TXEMPTY);
130 :     sct_vth.SetChunkMode(AJCSCT_CM_TEXT);
131 :     return 0;
132 : }
133 : //----- WM_DESTROY ---------------------------------------------------------------------------------------------//
134 : AJC_WNDPROC(Main, WM_DESTROY    )
135 : {
136 :     SAjxCtrl::CtrlSaveTextBox(hTxtServ, TEXT("MySect"));
137 :     DestroyWindow(hWndVth );
138 :     DestroyWindow(hTxtServ);
139 :     PostQuitMessage(0);
140 :     return 0;
141 : }
142 : //----- WM_SIZE ------------------------------------------------------------------------------------------------//
143 : AJC_WNDPROC(Main, WM_SIZE       )
144 : {
145 :     MoveWindow(hTxtServ, 1,  1, LOWORD(lParam) - 2, 18                , FALSE);
146 :     MoveWindow(hWndVth , 0, 20, LOWORD(lParam)    , HIWORD(lParam) - 20, FALSE);
147 :     return 0;
148 : }
149 : //--------------------------------------------------------------------------------------------------------------//
150 : AJC_WNDMAP_DEF(Main)
151 :     AJC_WNDMAP_MSG(Main, WM_CREATE      )
152 :     AJC_WNDMAP_MSG(Main, WM_DESTROY     )
153 :     AJC_WNDMAP_MSG(Main, WM_SIZE        )
154 : AJC_WNDMAP_END
```

使用例２

サーバと接続し、ファイルの送信を行い、サーバから受信したテキストを表示します。
ESC キーを押すとプログラムを終了します。

以下のコマンド操作を行います。

C 〈サーバ名〉------ サーバに回線を接続
D ------------------ 回線を切断
T 〈ファイルパス〉-- サーバにファイルを送信




```
 1 : //
 2 : //  SP_SCT2.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : using namespace AjxControl;
 7 :
 8 : //  ガイドテキスト
 9 : static const UT Help[] = TEXT("コマンド形式：¥n")
10 :                         TEXT("  C 〈サーバ名〉----- 回線接続¥n")
11 :                         TEXT("  D ---------------- 回線切断¥n")
12 :                         TEXT("  T 〈ファイルパス〉-- ファイル送信¥n")
13 :                         TEXT("ESC キー押下でプログラムを終了します。");
14 :
15 : //  コンソールウインドハンドル
16 : HWND    hWndCon = NULL;
17 :
18 : //----- CAjxSct + CAjxCon の派生クラス ------------------------------------------------------------------------//
19 : class CAjxSctEx : public CAjxSct, public CAjxCon
20 : {
21 : public:
22 :     //  接続通知
23 :     VO  OnNtcConnect()      override {
24 :         SAjxCon::PrintF(TEXT("- 回線を接続しました。¥n"));
25 :     }
26 :     //  切断通知
27 :     VO  OnNtcDisconnect()   override {
28 :         SAjxCon::PrintF(TEXT("- 回線を切断しました。¥n"));
29 :     }
30 :     //  接続失敗通知
31 :     VO  OnNtcCnErr(UI param) override {
32 :         SAjxCon::PrintF(TEXT("- 回線の接続を失敗しました。(%d)¥n"), param);
33 :     }
34 :     //  テキストチャンク受信通知
35 :     VO  OnNtcRxTextChunk(C_UTP pText) override {
36 :         SAjxCon::PutS(pText);
37 :     }
38 :     //  入力文字列と引数リスト通知
39 :     VO  OnNtcStr(int argc, UT *argv[], C_UTP pTxt) override {
40 :         //  入力文字列表示
41 :         SAjxCon::PrintF(TEXT("%s¥n"), pTxt);
42 :         //  コマンド実行
43 :         if (argc != 0) {
44 :             AJCSCT_STATE state = GetState();
45 :             if      (argc == 2 && _tcsicmp(argv[0], TEXT("C")) == 0) {  //  C 〈サーバ名〉
46 :                 if (state == AJCSCT_DISCONNECT) Connect(argv[1], TEXT("14238"));
47 :                 else                            SAjxCon::PrintF(TEXT("*** 回線は接続済です。¥n"));
48 :             }
49 :             else if (argc == 1 && _tcsicmp(argv[0], TEXT("D")) == 0) {  //  D
50 :                 if (state == AJCSCT_CONNECT   ) Disconnect();
51 :                 else                            SAjxCon::PrintF(TEXT("*** 回線は接続されていません。¥n"));
52 :             }
53 :             else if (argc == 2 && _tcsicmp(argv[0], TEXT("T")) == 0) {  //  T 〈ファイルパス〉
54 :                 if (state == AJCSCT_CONNECT) {
55 :                     CAjxFile f;
56 :                     UT  buf[1024];
57 :                     if (f.FOpen(argv[1])) {
58 :                         while (f.FGetS(buf, AJCTSIZE(buf))) {
59 :                             SendText(buf);
60 :                         }
61 :                         f.FClose();
62 :                     }
63 :                 }
64 :                 else {SAjxCon::PrintF(TEXT("*** 回線は接続されていないか、データ送信中です。¥n"));  }
65 :             }
66 :             else SAjxCon::PrintF(TEXT("- 不正なコマンドです。¥n"));
```

```
67 :         }
68 :     }
69 : };
70 : static CAjxSctEx    sct_con;
71 :
72 :
73 : int  AjcMain(int argc, UTP argv[])
74 : {
75 :     SAjxCon::SetStdMode();
76 :
77 :     //  コンソールのウインドハンドル取得
78 :     UT WndTxt[256];
79 :     GetConsoleTitle(WndTxt, 256);
80 :     hWndCon = FindWindow(NULL, WndTxt);
81 :     //  ＳＣＴ初期化
82 :     sct_con.SetEvtMask  (AJCSCT_EV_GENERAL | AJCSCT_EV_RXCHUNK);
83 :     sct_con.SetChunkMode(AJCSCT_CM_TEXT);
84 :     SAjxCon::PrintF(TEXT("¥n%s¥n¥n"), Help);
85 :     //  コンソール入力ループ
86 :     while (sct_con.Input(Help)) {
87 :         //  最初のイベント待ち（5秒）
88 :         if (sct_con.WaitEvent(5000)) {
89 :             //  ２つ目以降のイベント待ち（100ms）
90 :             while (sct_con.WaitEvent(100));
91 :         }
92 :         AjcSetFocusWindow(hWndCon);
93 :     }
94 :
95 :     return 0;
96 : }
97 :
```

2.11. ダイアログボックス／ウインド項目のアクセス (CAjxDlg, SAjxCtrl)

2.11.1. ダイアログボックス項目のアクセス(CAjxDlg)

メンバ変数

#	変数／関数形式	内容
1	HWND m_hDlg;	自ダイアログハンドル

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxDlg(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxDlg(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数（ダイアログ項目の取得）

#	変数／関数形式		内容
1	UI	GetDlgItemUInt (int id);	ダイアログ項目の取得（符号なし整数）
2	SI	GetDlgItemSInt (int id);	ダイアログ項目の取得（符号付き整数）
3	UI	GetDlgItemHex (int id);	ダイアログ項目の取得（１６進整数）
4	double	GetDlgItemReal (int id);	ダイアログ項目の取得（実数）
5	ULL	GetDlgItemUI64 (int id);	ダイアログ項目の取得（符号なし長整数）
6	SLL	GetDlgItemSI64 (int id);	ダイアログ項目の取得（符号付き長整数）
7	ULL	GetDlgItemH64 (int id);	ダイアログ項目の取得（１６進長整数）
8	UI	GetDlgItemChk (int id);	ダイアログ項目の取得（チェックボックス／ラジオボタン）
9	UI	GetDlgItemStr (int id, UTP pBuf, UI lBuf);	ダイアログ項目の取得（文字列）
10	UI	GetDlgItemStrLen (int id);	ダイアログ項目の取得（文字列長）
11	int	GetDlgItemSldPos (int id);	ダイアログ項目の取得（スライダの位置）
12	UI	GetDlgItemCboIx (int id);	ダイアログ項目の取得（コンボボックス・インデクス）
13	UI	GetDlgItemCboCount (int id);	ダイアログ項目の取得（コンボボックス・項目数）
14	UI	GetDlgItemCboLen (int id, UI ix);	ダイアログ項目の取得（コンボボックス・文字長）
15	UI	GetDlgItemCboItem (int id, UI ix, UTP pBuf, UI lBuf);	ダイアログ項目の取得（コンボボックス・項目の文字列）
16	SX	GetDlgItemCboData (int id, UI ix);	ダイアログ項目の取得（コンボボックス・データ値）
17	UI	GetDlgItemCboFind (int id, UI ix, C_UTP pStr, UI flag);	ダイアログ項目の取得（コンボボックス・文字列検索）
18	UI	GetDlgItemCboMaxLen (int id);	ダイアログ項目の取得（コンボボックス・項目群のの最大文字列長）

メンバ関数（ダイアログ項目の設定）

#	変数／関数形式		内容
1	BOOL	SetDlgItemUInt (int id, UI value);	ダイアログ項目の設定（符号なし整数）
2	BOOL	SetDlgItemSInt (int id, SI value);	ダイアログ項目の設定（符号付き整数）
3	BOOL	SetDlgItemHex (int id, UI value);	ダイアログ項目の設定（１６進整数）
4	BOOL	SetDlgItemReal (int id, double value, int prec);	ダイアログ項目の設定（実数）
5	BOOL	SetDlgItemUI64 (int id, ULL value);	ダイアログ項目の設定（符号なし長整数）
6	BOOL	SetDlgItemSI64 (int id, SLL value);	ダイアログ項目の設定（符号付き長整数）
7	BOOL	SetDlgItemH64 (int id, ULL value);	ダイアログ項目の設定（１６進長整数）
8	BOOL	SepDlgItemUInt (int id, UI value);	ダイアログ項目の設定（符号なし整数, ３桁区切り）
9	BOOL	SepDlgItemSInt (int id, SI value);	ダイアログ項目の設定（符号付き整数, ３桁区切り）
10	BOOL	SepDlgItemHex (int id, UI value, int col);	ダイアログ項目の設定（１６進整数 , 桁数指定）
11	BOOL	SepDlgItemReal (int id, double value, int prec);	ダイアログ項目の設定（実数, ３桁区切り）
12	BOOL	SepDlgItemUI64 (int id, ULL value);	ダイアログ項目の設定（符号なし長整数, ３桁区切り）
13	BOOL	SepDlgItemSI64 (int id, SLL value);	ダイアログ項目の設定（符号付き長整数, ３桁区切り）
14	BOOL	SepDlgItemH64 (int id, ULL value, int col);	ダイアログ項目の設定（１６進長整数 , 桁数指定）
15	BOOL	SetDlgItemChk (int id, UI state);	ダイアログ項目の設定（チェックボックス／ラジオボタン）
16	BOOL	SetDlgItemRbt (int id, int idFirst, int idLast);	ダイアログ項目の設定（ラジオボタン）

メンバ関数（ダイアログ項目の設定）

#	変数／関数形式	内容
17	BOOL SetDlgItemStr (int id, C_UTP pStr);	ダイアログ項目の設定（文字列）
18	BOOL SetDlgItemFStr (int id, C_UTP pFmt, ...);	ダイアログ項目の設定（書式文字列）
19	BOOL SetDlgItemEdtLimit(int id, UI limit);	ダイアログ項目の設定（エディットコントロールの桁数）
20	BOOL SetDlgItemCboIx (int id, UI ix);	ダイアログ項目の設定（コンボボックス・インデクス）
21	UI SetDlgItemCboAdd (int id, UI ix, C_UTP pItem, UI flag = AJCCBF_ALL);	ダイアログ項目の設定（コンボボックス・項目の文字列）
22	UI SetDlgItemCboIns (int id, UI ix, C_UTP pItem, UI flag = AJCCBF_ALL);	ダイアログ項目の設定（コンボボックス・項目挿入）
23	BOOL SetDlgItemCboDel (int id, UI ix);	ダイアログ項目の設定（コンボボックス・項目削除）
24	BOOL SetDlgItemCboData (int id, UI ix, SX data);	ダイアログ項目の設定（コンボボックス・データ値）
25	BOOL SetDlgItemCboList (int id, C_UTP pList);	ダイアログ項目の設定（コンボボックス・項目リスト）
26	BOOL SetDlgItemCboLimit(int id, UI limit);	ダイアログ項目の設定（コンボボックス・入力可能文字数）
27	BOOL SetDlgItemCboReset(int id);	ダイアログ項目の設定（コンボボックス・リセット）
28	BOOL SetDlgItemPgsRange (int id, int low, int high);	ダイアログ項目の設定（プログレスバーのレンジ）
29	BOOL SetDlgItemPgsPos (int id, int pos);	ダイアログ項目の設定（プログレスバーの位置）
30	BOOL SetDlgItemSldRange (int id, int low, int high, int page);	ダイアログ項目の設定（スライダのレンジ）
31	BOOL SetDlgItemSldPos (int id, int pos);	ダイアログ項目の設定（スライダの位置）
32	BOOL SetDlgItemSpnInfo (int id, int low, int high, int idBuddy);	ダイアログ項目の設定（スピンボタンのレンジとバディウインド
33	BOOL SetDlgItemSpnRange (int id, int low, int high);	ダイアログ項目の設定（スピンボタンのレンジ
34	BOOL SetDlgItemSpnBuddy (int id, int idBuddy);	ダイアログ項目の設定（スピンボタンのバディウインド
35	BOOL EnableDlgItemToDrop (int id, UI optDrop= AJCDROP_DIR_AND_FILE);	EDIT コントロールにフォルダ／ファイルをドロップ可能にする

メンバ関数（ダイアログ項目のグレィ設定／表示／移動）

#	変数／関数形式	内容
1	BOOL EnableDlgItem (int id , BOOL fEnable);	ダイアログ項目の有効化／無効化
2	BOOL EnableDlgGroup (int idGrp, BOOL fEnaGrpBox, BOOL fEnaCtrls);	グループボックス内の全コントロール有効化／無効化
3	BOOL ShowDlgGroup (int idGrp, BOOL fShow);	グループボックスとグループボックス内の全コントロール表示／非表示
4	BOOL ShowDlgGroup (int idGrp, BOOL fShowGrpBox, BOOL fShowCtrls);	グループボックスやグループボックス内の全コントロール表示／非表示（拡張）
5	BOOL MoveDlgGroupToLoc (int idGrp, int x, int y);	グループボックスとグループボックス内の全コントロールを指定位置へ移動
6	BOOL MoveDlgGroupToCtl (int idGrp, int idCtl);	グループボックスとグループボックス内の全コントロールを指定コントロールのへ移動
7	BOOL MoveDlgGroupToOrg (int idGrp);	グループボックスとグループボックス内の全コントロールを移動前の位置に戻す

メンバ関数（ダイアログ項目群の永続化）

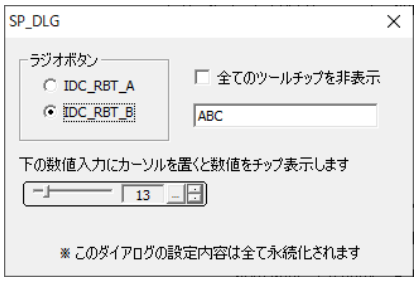
#	変数／関数形式	内容
1	BOOL DelAllCtrlPermAtt ();	全コントロールの永続化属性解除
2	BOOL LoadAllControlSettings (C_UTP pSect, UI act = AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);	全コントロールの設定内容の読み出し
3	BOOL SaveAllControlSettings ();	全コントロールの設定内容の書き込み
4	BOOL DlgItemSetPermAtt (UI id, UI att);	各コントロールの永続化属性の設定
5	BOOL DlgItemLoadTextBox (UI id, C_UTP pSect, BOOL fExcRdOnly = TRUE);	テキストボックスの読み出し(読出専用の動作を指定)
6	BOOL DlgItemSaveTextBox (UI id, C_UTP pSect, BOOL fExcRdOnly = TRUE);	テキストボックスの書き込み(読出専用の動作を指定)
7	BOOL DlgItemLoadChkBox (UI id, C_UTP pSect, BOOL fNtc = FALSE);	チェックボックス／ラジオボタンの読み出し
8	BOOL DlgItemSaveChkBox (UI id, C_UTP pSect);	チェックボックス／ラジオボタンの書き込み
9	BOOL DlgItemLoadComboBox (UI id, C_UTP pSect, UI sel = 0, BOOL fNtc = FALSE);	コンボボックスの読み出し
10	BOOL DlgItemSaveComboBox (UI id, C_UTP pSect、UI sel = 0);	コンボボックスの書き込み
11	BOOL DlgItemLoadListBox (UI id, C_UTP pSect, UI sel = 0);	リストボックスの読み出し
12	BOOL DlgItemSaveListBox (UI id, C_UTP pSect, UI sel = 0);	リストボックスの書き込み

仮想関数

#	変数／関数形式	内容
1	virtual VO Attach(HWND hDlg);	ダイアログハンドル設定

使用例１

ダイアログ項目のアクセスと、全項目の永続化の例を示します。




```
 1 : //
 2 : //  SP_DLG.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : #include    "resource.h"
 7 : using namespace AjxControl;
 8 :
 9 : static  HINSTANCE   hInst = NULL;
10 : static  HWND        hDlgMain = NULL;
11 : static  CAjxDlg     dlg;
12 : AJC_DLGPROC_DEF(Main);
13 :
14 : //----- CAjxTip の派生クラス ----------------------------------------------------------------------------------//
15 : class CAjxTipEx :
16 :     public CAjxTip
17 : {
18 : public:
19 :     VO  OnNeedText(HWND hCtrl, UTP pBuf, UI lBuf) override  {       //  チップテキスト取得通知
20 :         AjcSnPrintF(pBuf, lBuf, TEXT("設定値 = %u"), dlg.GetDlgItemUInt(IDC_INP));
21 :     }
22 : };
23 : static  CAjxTipEx   tip;
24 :
25 : //----- WinMain ----------------------------------------------------------------------------------------------//
26 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
27 : {
28 :     MSG         msg = {0};
29 :
30 :     hInst = hInstance;
31 :
32 :     //  ダイアログ生成
33 :     hDlgMain = CreateDialog(hInst, MAKEINTRESOURCE(IDD_MAIN), NULL, AJC_DLGPROC_NAME(Main));
34 :     ShowWindow(hDlgMain, iCmdShow);
35 :     //  メッセージループ
36 :     while (GetMessage(&msg, NULL, 0, 0)) {
37 :         do {
38 :             if (IsDialogMessage(hDlgMain, &msg)) break;
39 :             TranslateMessage(&msg);
40 :             DispatchMessage (&msg);
41 :         } while(0);
42 :     }
43 :     return (int)msg.wParam ;
44 : }
45 : //----- WM_CREATE --------------------------------------------------------------------------------------------//
46 : AJC_DLGPROC(Main, WM_INITDIALOG     )
47 : {
48 :     //  ダイアログハンドル設定
49 :     dlg.Attach(hDlg);
50 :     //  数値入力のハンドル設定
51 :     tip.Attach(GetDlgItem(hDlg, IDC_INP));
52 :     //  ツールチップ設定
53 :     SAjxTip::Add(GetDlgItem(hDlg, IDC_CHK  ), TEXT("全てのチップテキストを表示しないようにします。"));
54 :     SAjxTip::Add(GetDlgItem(hDlg, IDC_RBT_A), TEXT("ラジオボタンＡ"));
55 :     SAjxTip::Add(GetDlgItem(hDlg, IDC_RBT_B), TEXT("ラジオボタンＢ"));
56 :     SAjxTip::Add(GetDlgItem(hDlg, IDC_TXT  ), TEXT("ここにテキストを入力してください"));
57 :     //  デフォルト値設定
58 :     dlg.SetDlgItemChk (IDC_RBT_A, TRUE);
59 :     dlg.SetDlgItemStr (IDC_TXT  , TEXT("ここにテキストを入力"));
60 :     dlg.SetDlgItemUInt(IDC_INP  , 12);
61 :     //  プロファイルから設定値読み出し
62 :     dlg.LoadAllControlSettings(TEXT("Settings"));
63 :
```

```
 64 :     return TRUE;
 65 : }
 66 : //----- WM_DESTROY ------------------------------------------------------------------------------------------//
 67 : AJC_DLGPROC(Main, WM_DESTROY    )
 68 : {
 69 :     //  プロファイルへ設定値書き込み
 70 :     dlg.SaveAllControlSettings();
 71 :
 72 :     PostQuitMessage(0);
 73 :     return TRUE;
 74 : }
 75 : //----- RBT_A -----------------------------------------------------------------------------------------------//
 76 : AJC_DLGPROC(Main, IDC_RBT_A     )
 77 : {
 78 :     SAjxTip::ShowCenter(hDlg, TEXT("ラジオボタン(RBT_A)が選択されました。"));
 79 :     return TRUE;
 80 : }
 81 : //----- RBT_B -----------------------------------------------------------------------------------------------//
 82 : AJC_DLGPROC(Main, IDC_RBT_B     )
 83 : {
 84 :     SAjxTip::ShowCenter(hDlg, TEXT("ラジオボタン(RBT_B)が選択されました。"));
 85 :     return TRUE;
 86 : }
 87 : //----- CHK -------------------------------------------------------------------------------------------------//
 88 : AJC_DLGPROC(Main, IDC_CHK       )
 89 : {
 90 :     SAjxTip::EnableAll(!dlg.GetDlgItemChk(IDC_CHK));
 91 :     return TRUE;
 92 : }
 93 : //----- CANCEL ----------------------------------------------------------------------------------------------//
 94 : AJC_DLGPROC(Main, IDCANCEL      )
 95 : {
 96 :     DestroyWindow(hDlg);
 97 :     return TRUE;
 98 : }
 99 : //-----------------------------------------------------------------------------------------------------------//
100 : AJC_DLGMAP_DEF(Main)
101 :     AJC_DLGMAP_MSG(Main, WM_INITDIALOG  )
102 :     AJC_DLGMAP_MSG(Main, WM_DESTROY     )
103 :     AJC_DLGMAP_CMD(Main, IDC_RBT_A      )
104 :     AJC_DLGMAP_CMD(Main, IDC_RBT_B      )
105 :     AJC_DLGMAP_CMD(Main, IDC_CHK        )
106 :     AJC_DLGMAP_CMD(Main, IDCANCEL       )
107 : AJC_DLGMAP_END
```

2.11.2. ウインド項目のアクセス(SAjxCtrl)

表中の「fUnicode = ***XXXXX***;」の「***XXXXX***;」は、コンパイルオプション「UNICODE」が指定されている場合は「TRUE」、指定されていない場合は「FALSE」となります。

スタティック関数（コントロールの取得）

#	変数／関数形式			内容
1	static UI	GetCtrlUInt	(HWND hCtrl);	コントロールの取得（符号なし整数）
2	static SI	GetCtrlSInt	(HWND hCtrl);	コントロールの取得（符号付き整数）
3	static UI	GetCtrlHex	(HWND hCtrl);	コントロールの取得（１６進整数）
4	static double	GetCtrlReal	(HWND hCtrl);	コントロールの取得（実数）
5	static ULL	GetCtrlUI64	(HWND hCtrl);	コントロールの取得（符号なし長整数）
6	static SLL	GetCtrlSI64	(HWND hCtrl);	コントロールの取得（符号付き長整数）
7	static ULL	GetCtrlH64	(HWND hCtrl);	コントロールの取得（１６進長整数）
8	static UI	GetCtrlChk	(HWND hCtrl);	コントロールの取得（チェックボックス）
9	static UI	GetCtrlStr	(HWND hCtrl, UTP pBuf, UI lBuf);	コントロールの取得（文字列）
10	static UI	GetCtrlStrLen	(HWND hCtrl, BOOL fUnicode=***XXXXX***);	コントロールの取得（文字列長）
11	static int	GetCtrlSldPos	(HWND hCtrl);	コントロールの取得（スライダの位置）
12	static UI	GetCtrlCboIx	(HWND hCtrl);	コントロールの取得（コンボボックス・インデクス）
13	static UI	GetCtrlCboCount	(HWND hCtrl);	コントロールの取得（コンボボックス・項目数）
14	static UI	GetCtrlCboLen	(HWND hCtrl, UI ix, BOOL fUnicode = ***XXXXX***);	コントロールの取得（コンボボックス・文字長）
15	static UI	GetCtrlCboItem	(HWND hCtrl, UI ix, UTP pBuf, UI lBuf);	コントロールの取得（コンボボックス・項目の文字列）
16	static SX	GetCtrlCboData	(HWND hCtrl, UI ix);	コントロールの取得（コンボボックス・データ値）
17	static UI	GetCtrlCboFind	(HWND hCtrl, UI ix, C_UTP pStr, UI flag);	コントロールの取得（コンボボックス・文字列検索）
18	static UI	GetCtrlCboMaxLen	(HWND hCtrl, BOOL fUnicode = ***XXXXX***);	コントロールの取得（コンボボックス・項目群のの最大文字列長）

スタティック関数（コントロールの設定）

#	変数／関数形式			内容
1	static BOOL	SetCtrlUInt	(HWND hCtrl, UI value);	コントロールの設定（符号なし整数）
2	static BOOL	SetCtrlSInt	(HWND hCtrl, SI value);	コントロールの設定（符号付き整数）
3	static BOOL	SetCtrlHex	(HWND hCtrl, UI value);	コントロールの設定（１６進整数）
4	static BOOL	SetCtrlReal	(HWND hCtrl, double value, int prec);	コントロールの設定（実数）
5	static BOOL	SetCtrlUI64	(HWND hCtrl, ULL value);	コントロールの設定（符号なし長整数）
6	static BOOL	SetCtrlSI64	(HWND hCtrl, SLL value);	コントロールの設定（符号付き長整数）
7	static BOOL	SetCtrlH64	(HWND hCtrl, ULL value);	コントロールの設定（１６進長整数）
8	static BOOL	SepCtrlUInt	(HWND hCtrl, UI value);	コントロールの設定（符号なし整数, ３桁区切り）
9	static BOOL	SepCtrlSInt	(HWND hCtrl, SI value);	コントロールの設定（符号付き整数, ３桁区切り）
10	static BOOL	SepCtrlHex	(HWND hCtrl, UI value, int col);	コントロールの設定（１６進整数 , 桁数指定）
11	static BOOL	SepCtrlReal	(HWND hCtrl, double value, int prec);	コントロールの設定（実数, ３桁区切り）
12	static BOOL	SepCtrlUI64	(HWND hCtrl, ULL value);	コントロールの設定（符号なし長整数, ３桁区切り）
13	static BOOL	SepCtrlSI64	(HWND hCtrl, SLL value);	コントロールの設定（符号付き長整数, ３桁区切り）
14	static BOOL	SepCtrlH64	(HWND hCtrl, ULL value, int col);	コントロールの設定（１６進長整数 , 桁数指定）
15	static BOOL	SetCtrlChk	(HWND hCtrl, UI state);	コントロールの設定（チェックボックス）
16	static BOOL	SetCtrlRbt	(HWND hCtrl, int idFirst, int idLast);	コントロールの設定（ラジオボタン）
17	static BOOL	SetCtrlStr	(HWND hCtrl, C_UTP pStr);	コントロールの設定（文字列）
18	static BOOL	SetCtrlFStr	(HWND hCtrl, C_UTP pFmt, ...);	コントロールの設定（書式文字列）
19	static BOOL	SetCtrlEdtLimit	(HWND hCtrl, UI limit, BOOL fUnicode= ***XXXXX***);	コントロールの設定（エディットコントロールの桁数）

スタティック関数（コントロールの設定）

#	変数／関数形式	内容
20	static BOOL SetCtrlCboIx (HWND hCtrl, UI ix);	コントロールの設定（コンボボックス・インデクス）
21	static UI SetCtrlCboAdd (HWND hCtrl, UI ix, C_UTP pItem, UI flag = AJCCBF_ALL);	コントロールの設定（コンボボックス・項目の文字列）
22	static UI SetCtrlCboIns (HWND hCtrl, UI ix, CUTP pItem, UI flag = AJCCBF_ALL);	コントロールの設定（コンボボックス・項目挿入）
23	static BOOL SetCtrlCboDel (HWND hCtrl, UI ix);	コントロールの設定（コンボボックス・項目削除）
24	static BOOL SetCtrlCboData (HWND hCtrl, UI ix, SX data);	コントロールの設定（コンボボックス・データ値）
25	static BOOL SetCtrlCboList (HWND hCtrl, C_UTP pList);	コントロールの設定（コンボボックス・項目リスト）
26	static BOOL SetCtrlCboLimit (HWND hCtrl, UI limit, BOOL fUnicode = ***XXXXX***);	コントロールの設定（コンボボックス・入力可能文字数）
27	static BOOL SetCtrlCboReset (HWND hCtrl);	コントロールの設定（コンボボックス・リセット）
28	static BOOL SetCtrlPgsRange (HWND hCtrl, int low, int high);	コントロールの設定（プログレスバーのレンジ）
29	static BOOL SetCtrlPgsPos (HWND hCtrl, int pos);	コントロールの設定（プログレスバーの位置）
30	static BOOL SetCtrlSldRange (HWND hCtrl, int low, int high, int page);	コントロールの設定（スライダのレンジ）
31	static BOOL SetCtrlSldPos (HWND hCtrl, int pos);	コントロールの設定（スライダの位置）
32	static BOOL SetCtrlSpnInfo (HWND hCtrl, int low, int high, HWND hBuddy);	コントロールの設定（スピンボタンのレンジとバディウインド
33	static BOOL SetCtrlSpnRange (HWND hCtrl, int low, int high);	コントロールの設定（スピンボタンのレンジ）
34	static BOOL SetCtrlSpnBuddy (HWND hCtrl, HWND hBuddy);	コントロールの設定（スピンボタンのバディウインド）
35	static HWND SetCtrlTopMostInSiblings (HWND hCtrl);	コントロールを兄弟ウインド中で再前面に設定する
36	static HWND SetCtrllBackMostInSiblings (HWND hCtrl);	コントロールを兄弟ウインド中で再背面に設定する
37	static BOOL EnableCtrlToDrop(HWND hCtrl, UI optDrop = AJCDROP_DIR_AND_FILE);	EDIT コントロールにフォルダ／ファイルをドロップ可能にする

スタティック関数（コントロールのグレィ設定／表示／移動）

#	変数／関数形式	内容
1	static BOOL EnableCtrl (HWND hCtrl, BOOL fEnable);	コントロールの有効化／無効化
2	static BOOL EnableGroup (HWND hCtrl, BOOL fEnaGrpBox, BOOL fEnaCtrls);	グループボックス内の全コントロール有効化／無効化
3	static BOOL ShowGroup (HWND hCtrl, BOOL fShow);	グループボックスとグループボックス内の全コントロール表示／非表示
4	static BOOL ShowGroup (HWND hCtrl, BOOL fShowGrpBox, BOOL fShowCtrls);	グループボックスやグループボックス内の全コントロール表示／非表示 (拡張)
5	static BOOL MoveGroupToLoc (HWND hCtrl, int x, int y);	グループボックスとグループボックス内の全コントロールを指定位置へ移動
6	static BOOL MoveGroupToCtl (HWND hCtrl, HWND hctl);	グループボックスとグループボックス内の全コントロールを指定コントロールの位置へ移動
7	static BOOL MoveGroupToOrg (HWND hCtrl);	グループボックスとグループボックス内の全コントロールを移動前の位置に戻す

スタティック関数（コントロール群の永続化）

#	変数／関数形式	内容	
1	static BOOL DelAllCtrlPermAtt (HWND hWnd);	全コントロールの永続化属性解除	コントロール群の親ウインドを指定する
2	static BOOL LoadAllControlSettings (HWND hWnd, C_UTP pSect, UI act = AJCCTL_SELACT_ALL | AJCCTL_SELACT_NTCALL);	全コントロールの設定内容の読み出し	
3	static BOOL SaveAllControlSettings (HWND hWnd);	全コントロールの設定内容の書き込み	
4	static BOOL CtrlSetPermAtt (HWND hCtrl, UI att);	各コントロールの永続化属性の設定	
5	static BOOL CtrlLoadTextBox (HWND hCtrl, C_UTP pSect, BOOL fExcRdOnly = TRUE);	テキストボックスの読み出し(読出専用の動作を指定)	
6	static BOOL CtrlSaveTextBox (HWND hCtrl, C_UTP pSect, BOOL fExcRdOnly = TRUE);	テキストボックスの書き込み(読出専用の動作を指定)	
7	static BOOL CtrlLoadChkBox (HWND hCtrl, C_UTP pSect);	チェックボックス／ラジオボタンの読み出し	
8	static BOOL CtrlSaveChkBox (HWND hCtrl, C_UTP pSect);	チェックボックス／ラジオボタンの書き込み	
9	static BOOL CtrlLoadComboBox(HWND hCtrl, C_UTP pSect, UI sel = 0);	コンボボックスの読み出し	
10	static BOOL CtrlSaveComboBox(HWND hCtrl, C_UTP pSect, UI sel = 0);	コンボボックスの書き込み	
11	static BOOL CtrlLoadListBox (HWND hCtrl, C_UTP pSect, UI sel = 0);	リストボックスの読み出し	
12	static BOOL CtrlSaveListBox (HWND hCtrl, C_UTP pSect, UI sel = 0);	リストボックスの書き込み	

2.12. 拡張ツールチップ (SAjxTip)

2.12.1. スタティックファンクション

メンバ関数（スタティック関数）

#	変数／関数形式	内容
1	static BOOL EnableMultiThread(BOOL fEnable);	マルチスレッドの許可／禁止
2	static AJCTIP_POSMODE SetTipPosMode (AJCTIP_POSMODE PosMode);	ツールチップ表示位置モードの設定
3	static BOOL Add(HWND hCtrl, C_UTP pTxt, int msDelay = 1000, int msShow = 3000, HFONT hFont = NULL, COLORREF TextColor = (COLORREF)-1, COLORREF BackGround = (COLORREF)-1, COLORREF BorderColor = (COLORREF)-1);	ツールチップの関連付け
4	static BOOL SetShowAlways(HWND hCtrl, BOOL fShowAlways);	ツールチップ表示条件の設定
5	static BOOL GetShowAlways(HWND hCtrl);	ツールチップ表示条件の取得
6	static VO SetShowForActive(BOOL fShowForActive);	全ツールチップ表示条件の設定
7	static BOOL GetShowForActive(VO);	全ツールチップ表示条件の取得
8	static BOOL Enable(HWND hCtrl, BOOL fEnable);	チップテキストの表示許可／禁止設定
9	static BOOL GetEnableState(HWND hCtrl);	チップテキストの表示許可／禁止状態取得
10	static VO EnableAll(BOOL fEnable);	全てのチップテキストの表示許可／禁止設定
11	static BOOL GetEnableAllState();	全てのチップテキストの表示許可／禁止状態取得
12	static BOOL Remove(HWND hCtrl);	特定のツールチップの関連付け解除
13	static BOOL Remove();	全てのツールチップの関連付け解除
14	static UI GetInfo(HWND hCtrl, PAJCTIPINFO pInfoBuf, UTP pTxtBuf, UI lTxtBuf);	ツールチップの関連付け情報取得
15	static BOOL SetNtcSubclass(HWND hCtrl, HWND hNtc, UI msg, UX lParam);	サブクラス化の通知情報設定
16	static BOOL ClrNtcSubclass(HWND hCtrl);	サブクラス化の通知情報消去
17	static VO SetWindowTime(int msTime);	コントロール間移動猶予時間設定
18	static int GetWindowTime();	コントロール間移動猶予時間取得
19	static VO SetDefMsDelay(int msDelay);	デフォルトの表示遅延時間の設定
20	static int GetDefMsDelay();	デフォルトの表示遅延時間の取得
21	static BOOL Show(int x, int y, C_UTP pTxt, int msTime = 3000, HFONT hFont = NULL, COLORREF TextColor = (COLORREF)-1, COLORREF BackGround = (COLORREF)-1, COLORREF BorderColor = (COLORREF)-1);	ツールチップの表示
22	static BOOL Show(int x, int y, int minWidth, int height, C_UTP pTxt, int msTime = 3000, HFONT hFont = NULL, COLORREF TextColor = (COLORREF)-1, COLORREF BackGround = (COLORREF)-1, COLORREF BorderColor = (COLORREF)-1);	ツールチップの表示(最小チップサイズ指定)
23	static BOOL ShowCenter(HWND hwnd, C_UTP pTxt, int msTime = 3000, HFONT hFont = NULL, COLORREF TextColor = (COLORREF)-1, COLORREF BackGround = (COLORREF)-1, COLORREF BorderColor = (COLORREF)-1);	ツールチップをウインド中央に表示
24	static BOOL ShowCenter(HWND hwnd, int minWidth, int height, C_UTP pTxt, int msTime = 3000, HFONT hFont = NULL, COLORREF TextColor = (COLORREF)-1, COLORREF BackGround = (COLORREF)-1, COLORREF BorderColor = (COLORREF)-1);	ツールチップをウインド中央に表示(最小チップサイズ指定)
25	static BOOL GetSize(int w, int h, C_UTP pTxt, HFONT hFont, BOOL fBorder, LPSIZE pBuf);	ツールチップの表示サイズ取得

つづく

#	変数／関数形式	内容
26	static VO Move(int x, int y);	ツールチップウインドを移動
27	static VO MoveCursor();	マウスカーソルをツールチップの中央へ移動
28	static VO MoveCursor(AJCTIP_CURMOVE cm);	マウスカーソルをツールチップの指定位置へ移動
29	static VO Hide();	ツールチップ非表示
30	static BOOL GetRect(LPRECT pRect);	ツールチップウインドの矩形情報取得
31	static BOOL SetPalette(int ix, COLORREF color);	パレット色の設定
32	static COLORREF GetPalette(int ix);	パレット色の取得
33	static VO SetDefTextColor (COLORREF color);	デフォルトのテキスト表示色設定
34	static COLORREF GetDefTextColor ();	デフォルトのテキスト表示色取得
35	static VO SetDefBorderColor(COLORREF color);	デフォルトの外枠表示色設定
36	static COLORREF GetDefBorderColor();	デフォルトの外枠表示色取得
37	static VO SetDefBkColor (COLORREF color);	デフォルトの背景表示色設定
38	static COLORREF GetDefBkColor ();	デフォルトの背景表示色取得
39	static VO SetDefFont (HFONT hFont);	デフォルトフォントの設定
40	static HFONT GetDefFont ();	デフォルトフォントの取得
41	static VO SetDefMsShow (int msShow);	デフォルトの表示時間の設定
42	static int GetDefMsShow ();	デフォルトの表示時間の取得

2.12.2. 状況依存ツールチップ (CAjxTip)

メンバ変数

#	変数／関数形式	内容
1	HWND m_hCtrl;	自コントロールのウインドハンドル

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxTip(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxTip(BOOL fUnicode = TRUE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	VO Attach(HWND hCtrl);	ハンドルを関連付け

仮想関数

#	変数／関数形式	内容	
1	virtual BOOL OnChkCursor(HWND hCtrl, LPPOINT ptClient);	カーソル位置チェック要求	cbChkCursor
2	virtual VO OnNeedText (HWND hCtrl, UTP pBuf, UI lBuf);	ツールチップテキスト取得要求	cbGetText

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例は、「ダイアログボックス／ウインド項目のアクセス (CAjxDlg, SAjxCtrl)」の「使用例１」を参照。

2.13. コンソール

2.13.1. スタティックファンクション (SAjxCon)

メンバ関数（スタティック関数）

#	変数／関数形式	内容
1	static BOOL SetStdMode(EAJCTEC tec = AJCTEC_MBC, BOOL fBom = FALSE);	標準出力のモード設定
2	static int PutS (C_UTP pStr);	文字列をコンソール(標準出力)へ出力
3	static int PrintF (C_UTP pFmt, ...);	書式文字列をコンソール(標準出力)へ出力
4	static int ErrPrintF (C_BCP pFmt, ...);	書式文字列をコンソール(標準エラー)へ出力
5	static BOOL GetScreenBufferInfo (LPSIZE pBufSize, LPRECT pRcWnd = NULL, LPSIZE pMaxWndSize = NULL);	コンソール情報取得
6	static BOOL GetMaxWndSize (int *pCx , int *pCy);	ウインド最大サイズ取得
7	static BOOL SetBufSize (int cx, int cy);	コンソールバッファサイズ設定
8	static BOOL GetBufSize (int *pCx , int *pCy);	コンソールバッファサイズ取得
9	static BOOL SetWndRect(int left, int top, int righ, int botto);	コンソールウインド矩形設定
10	static BOOL GetWndRect(int *pLeft, int *pTop, int *pRight, int *pBottom);	コンソールウインド矩形取得
11	static BOOL SetColor (COLORREF ForeColor, COLORREF BackColor = (COLORREF)-1);	表示色設定
12	static BOOL GetColor (LPCOLORREF pForeColor, LPCOLORREF pBackColor = NULL);	表示色取得
13	static BOOL SelPalette (UI ForePalette, UI BackPalette);	表示色パレット選択
14	static COLORREF GetForeColor ();	文字色パレット値の取得
15	static COLORREF GetBackColor ();	背景色パレット値の取得
16	static BOOL SetPalette (const COLORREF Palette[16]);	パレット値設定（全パレット）
17	static BOOL SetPalByIx (UI ix, COLORREF color);	パレット値設定（単一パレット）
18	static UI GetPalette (COLORREF Palette[16]);	パレット値取得（全パレット）
19	static COLORREF GetPalByIx (UI ix);	パレット値取得（単一パレット）
20	static BOOL SetCursor (int x, int y);	カーソル位置設定
21	static BOOL GetCursor (int *pX, int *pY);	カーソル位置取得

2.13.2. コンソール入力 (CAjxCon)

コンストラクタ

#	変数／関数形式	内容
1	CAjxCon()	コンストラクタ

メンバ関数

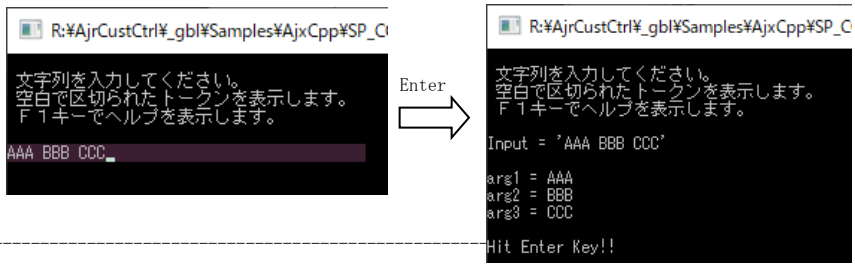
#	変数／関数形式	内容
1	BOOL Input (C_UTP pHelpText, // ヘルプテキスト UI lInpField = 80, // 入力域長 UI lBuf = 256, // バッファ長 UI fOpt = 0, // オプション COLORREF TextColor = (COLORREF)-1, //文字色 COLORREF BackColor = (COLORREF)-1); //背景色	コンソールからテキスト入力（入力バッファ指定を内部で生成） 入力したテキストは、OnNtcStr()で通知します。 ※第１引数（pHelpText）は必須 （ヘルプテキストが無い場合は空文字列（""）を指定）
2	BOOL Input (UI lInpField, // 入力域長 UTP pBuf, // 入力バッファ UI lBuf, // バッファ長 C_UTP pHelpText = NULL, // オプション UI fOpt = 0, // ヘルプテキスト COLORREF TextColor = (COLORREF)-1, // 文字色 COLORREF BackColor = (COLORREF)-1); // 背景色	コンソールからテキスト入力（初期テキスト＝入力バッファ）
3	BOOL Input (C_UTP pInitialText, // 初期テキスト UI lInpField, // 入力域長 UTP pBuf, // 入力バッファ UI lBuf, // バッファ長 C_UTP pHelpText = NULL, // ヘルプテキスト UI fOpt = 0, // オプション COLORREF TextColor = (COLORREF)-1, // 文字色 COLORREF BackColor = (COLORREF)-1); // 背景色	コンソールからテキスト入力（初期テキスト指定）

仮想関数

#	変数／関数形式	内容	
1	virtual VO OnNtcStr(int argc, BC *argv[]);	入力したテキストを空白で区切ったトークンを通知します。 ex.　入力：”AAA　BBB” 　　通知：argc = 2 　　argv[0] = “AAA” 　　argv[1] = “BBB”	cbNtcStr

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例




```
 1 : //
 2 : //  SP_CON.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : #include    <conio.h>
 7 : using namespace AjxControl;
 8 :
 9 : //----- CAjxCon の派生クラス ----------------------------------------------------------------------
10 : class CAjxConEx : public CAjxCon
11 : {
12 : public:
13 :     VO  OnNtcStr(int argc, UT *argv[], C_UTP pTxt) override {   //  引数リストと入力文字列通知
14 :         SAjxCon::PrintF(TEXT("Input = '%s'¥n¥n"), pTxt);
15 :         for (int i = 0; i < argc; i++) {
16 :             SAjxCon::PrintF(TEXT("arg%d = %s¥n"), i + 1, argv[i]);
17 :         }
18 :     }
19 :
20 : };
21 : static CAjxConEx    con;
22 :
23 :
24 : int  AjcMain(int argc, UTP argv[])
25 : {
26 :
27 :     MAjcAllocMainArgs(argc, argv);
28 :
29 :     //----- コマンドガイド表示 ---------------------------------------------------------------------------//
30 :     SAjxCon::PrintF(TEXT("¥n"));
31 :     SAjxCon::PrintF(TEXT(" 文字列を入力してください。¥n"));
32 :     SAjxCon::PrintF(TEXT(" 空白で区切られたトークンを表示します。¥n"));
33 :     SAjxCon::PrintF(TEXT(" Ｆ１キーでヘルプを表示します。¥n¥n"));
34 :
35 :     if (con.Input(TEXT(""))) {
36 :         SAjxCon::PrintF(TEXT("¥nHit Enter Key!!"));
37 :         getchar();
38 :     }
39 :     MAjcFreeMainArgs(argc, argv);
40 :
41 :     return 0;
42 : }
43 :
```

2.14. ファイル検索（CAjxFSrh）

コンストラクタ

#	変数／関数形式	内容
1	CAjxFsr();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	int SearchFiles(C_UTP pDir, C_UTP pWild = "*.*", BOOL fSubDir = TRUE, BOOL fNtcDir = TRUE);	ファイル／ディレクトリ検索
	int SearchMyComputer(C_BCP pPath, C_BCP pWild = "*.*", BOOL fRemote = FALSE, BOOL fNtcDir = TRUE);	マイコンピュータ検索

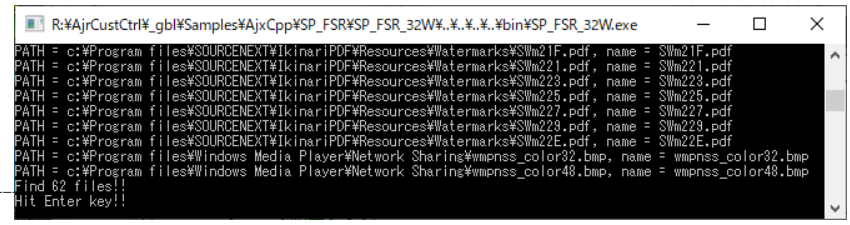
仮想関数

#	変数／関数形式	内容	
1	virtual BOOL OnFindDir (UTP pPath);	ディレクトリ通知	cbFind
2	virtual BOOL OnFindFile (UI nest, WCP pPath, WCP pName, UI att, UI wtime);	ファイル通知	

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例

「c:¥Program files」下から、ファイル（「*.bmp」と「*.pdf」）を検索します。




```
  1 : //
  2 : //  SP_FSR.cpp
  3 : //
  4 : #include    <AjrCstXX.h>
  5 : #include    <AjxCpp.h>
  6 : using namespace AjxControl;
  7 :
  8 : //----- CAjxFsr の派生クラス ----------------------
  9 : class CAjxFsrEx : public CAjxFsr
 10 : {
 11 : public:
 12 :     //----- 仮想関数 ---------------------------------------------------------------------------------------//
 13 :     //  ディレクトリ通知
 14 :     BOOL    OnFindDir   (UTP pPath) override    {
 15 :     //  SAjxCon::PrintF(TEXT("---------- %s ----------¥n"), pPath);
 16 :         return TRUE;
 17 :     }
 18 :     //  ファイル通知
 19 :     BOOL    OnFindFile  (UI nest, UTP pPath, UTP pName, UI att, UI wtime)   override    {
 20 :         SAjxCon::PrintF(TEXT("PATH = %s, name = %s¥n"), pPath, pName);
 21 :         return TRUE;
 22 :     }
 23 : };
 24 : static CAjxFsrEx    fsr;
 25 :
 26 : int  AjcMain(int argc, UTP argv[])
 27 : {
 28 :     SAjxCon::SetStdMode();
 29 :
 30 :     int n = fsr.SearchFiles(TEXT("c:¥¥Program files"), TEXT("*.bmp;*.pdf"));
 31 :
 32 :     SAjxCon::PrintF(TEXT("Find %d files!!¥n"), n);
 33 :     SAjxCon::PrintF(TEXT("Hit Enter key!!"));
 34 :     getchar();
 35 :
 36 :     return 0;
 37 : }
 38 :
```

2.15. テキストファイル・アクセス (CAjxFile)

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxFile(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxFile(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数（テキストファイル入力）

#	変数／関数形式	内容
1	BOOL FOpen (C_UTP pPath, EAJCTEC tec = AJCTEC_AUTO);	入力テキストファイルオープン
2	BOOL FOpenShae(C_UTP pPath, EAJCTEC tec = AJCTEC_AUTO, UI share=FILE_SHARE_READ | FILE_SHARE_WRITE);	入力テキストファイルオープン（共有設定）
3	BCP FGetS (UTP pBuf, UI lBuf);	文字列入力
4	BCP FGetS (UTP pBuf, UI lBuf);	文字列入力
5	UI FGetC ();	１文字入力（UNICODE モード時はワイド文字を、その他はバイト文字を返す）
6	ULL FSavePoint();	ファイル読み出しポイント退避
7	BOOL FRecvPoint();	ファイル読み出しポイント回復
8	BOOL FEof ();	入力ファイルのＥＯＦチェック

メンバ関数（テキストファイル出力）

#	変数／関数形式	内容
1	BOOL FCreate (C_UTP pPath, EAJCTEC tec = AJCTEC_AUTO, BOOL fBOM = FALSE);	出力テキストファイル生成
2	BOOL FCreateShare(C_UTP pPath, EAJCTEC tec = AJCTEC_AUTO, BOOL fBOM = FALSE, UI share = FILE_SHARE_READ | FILE_SHARE_WRITE);	出力テキストファイル生成（共有設定）
3	BOOL FAppend (C_UTP pPath, EAJCTEC tec = AJCTEC_AUTO, BOOL fBOM = FALSE);	出力テキストファイル追記
4	BOOL FAppendShare(C_UTP pPath, EAJCTEC tec = AJCTEC_AUTO, BOOL fBOM = FALSE, UI share = FILE_SHARE_READ | FILE_SHARE_WRITE);	出力テキストファイル追記（共有設定）
5	BOOL FPutS (C_UTP pText, UI lText = -1);	文字列出力
6	BOOL FPutC (UT c);	１文字出力
7	BOOL FPrintF (C_UTP pFmt, ...);	書式文字列出力

メンバ関数（その他）

#	変数／関数形式	内容
1	BOOL FClose ();	テキストファイルクローズ
2	EAJCTEC FGetTec ();	テキストエンコード種別取得
3	BOOL FGetBOM();	入力ファイルのＢＯＭ有無の取得
4	BOOL FIsActive();	ファイルが活性状態(オープン／生成状態)かチェック
5	BOOL FIsModeInput();	入出力モード取得
6	BOOL FSetFmtLen (UI len);	書式文字列バッファ長設定
7	HANDLE FGetFileHandle();	ファイルハンドル取得
8	BOOL FAttachOpened(HANDLE hFile, EAJCTEC tec);	オープン済ファイルのハンドルを使用してインスタンス生成
9	BOOL FAttachCreated(HANDLE hFile, EAJCTEC tec, BOOL fBOM);	生成済ファイルのハンドルを使用してインスタンス生成
10	BOOL FDetach();	ファイルをクローズしないでインスタンス解放

使用例　　ＶＴ－１００エミュレーション・ウインド コントロール (CAjxVth) 参照

2.16. 平衡２分木（ＡＶＬ木） (CAjxAvl)

コンストラクタ

#	変数／関数形式	内容
1	CAjxAvl();	コンストラクタ

メンバ関数（ノードアクセス）

#	変数／関数形式	内容
1	BOOL InsNode (UX key, C_VOP pNodeData, UI len);	ノード挿入
2	int GetNode (UX key, VOP pBuf, UI lBuf);	ノード取得
3	VOP GetNodePtr (UX key, UIP pLen);	ノードアドレス取得
4	VOP GetNodePtr (UX key);	ノードアドレス取得
5	BOOL RepNode (UX key, C_VOP pNodeData, UI len);	ノード置換
6	BOOL InsOrRepNode (UX key, C_VOP pNodeData, UI len);	ノード挿入／置換
7	BOOL DelNode (UX key);	ノード削除
8	BOOL DelAllNodes ();	全ノードを削除

メンバ関数（文字列ノードアクセス）

#	変数／関数形式	内容
11	UX InsStrNode (C_UTP pStr);	文字列ノード挿入
2	UX GetStrNode (C_UTP pStr);	文字列ノード取得
3	UX InsOrGetStrNode (C_UTP pStr);	文字列ノード挿入／取得
4	UX DelStrNode (C_UTP pStr);	文字列ノード削除

メンバ関数（その他）

#	変数／関数形式	内容
1	BOOL EnableMultiThread (BOOL fEnable);	マルチスレッドの許可／禁止
2	UI GetCount ();	ノード数取得
3	UI EnumNodes (BOOL fDownSeq = FALSE);	全ノードのシーケンシャルな読み出し（列挙）
4	PCAJCAVLPTR CreatePtrArr (UIP pNum = NULL, BOOL fDownSeq = FALSE);	全ノードへのポインタ配列生成
5	BOOL ReleasePtrArr (PCAJCAVLPTR pArr);	全ノードへのポインタ配列解放

仮想関数

#	変数／関数形式	内容	
1	virtual int OnKeyComp (UX key1, UX key2);	キーの比較	cbComp
2	virtual VO OnNtcRemove (UX key, VOP pNodeData, UI len, UI nest);	ノード削除通知 **(※1)**	cbRemove
3	virtual BOOL OnNtcNode (UX key, VOP pNodeData, UI len, UI nest);	ノード読み出し通知	cbNtcNode

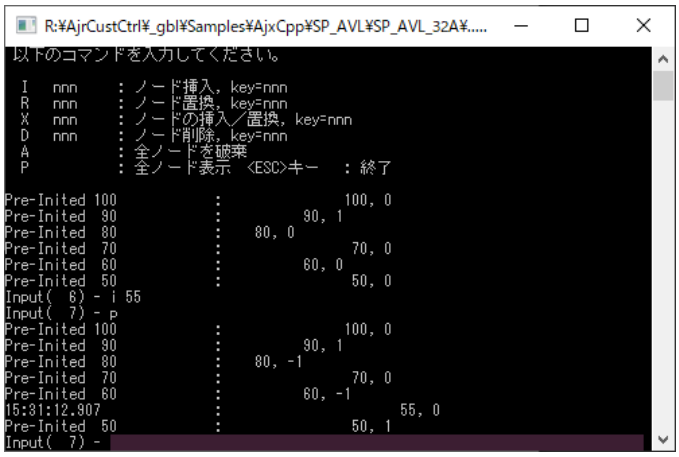
※ cb*XXXXX*は、対応するコールバック関数を示します。

※1：残留ノードがある状態でデストラクタが実行された場合、残留ノードは全て破棄されますが、デストラクト中にOnNtcRemove() は実行されません。
デストラクト中にOnNtcRemove() を実行したい場合は、デストラクタ内でDelAllNode() メソッドを実行してください。

使用例

以下のサンプルプログラムは、コマンド入力により、２文木にノード挿入や削除等を行います。
ノードデータは、時刻文字列へのポインタです。
時刻文字列は動的なメモリに格納し、ノードの消去通知時に解放します。
起動時は初期ノードとしてキー＝50, 60, 70, 80, 90, 100 の６つのノードを挿入します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	I nnn	ノード挿入, key=nnn
2	R nnn	ノード置換, key=nnn
3	X nnn	ノードの挿入／置換, key=nnn
4	D nnn	ノード削除, key=nnn
5	A	全ノードを破棄
6	P	全ノード表示



```
 1 : //
 2 : //  SP_AVL.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : using namespace AjxControl;
 7 :
 8 : //  ノードデータ形式
 9 : typedef struct {
10 :     UTP     pTimStr;            //  ヒープ上の時刻文字列へのポインタ
11 : } NODEDATA, *PNODEDATA;
12 :
13 : //---- CAjxAvl, CAjxCon の派生クラス ----------------------------------------------------------------//
14 : class CAjxAvlEx : public CAjxAvl, public CAjxCon
15 : {
16 : private:
17 :     NODEDATA    m_Node;
18 : public:
19 :     //  コンストラクタ (初期ノード登録)
20 :     CAjxAvlEx()
21 :     {
22 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited  50")); InsNode( 50, &m_Node, sizeof m_Node);
23 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited  60")); InsNode( 60, &m_Node, sizeof m_Node);
24 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited  70")); InsNode( 70, &m_Node, sizeof m_Node);
25 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited  80")); InsNode( 80, &m_Node, sizeof m_Node);
26 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited  90")); InsNode( 90, &m_Node, sizeof m_Node);
27 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited 100")); InsNode(100, &m_Node, sizeof m_Node);
28 :     }
29 :     //  デストラクタ
30 :     ~CAjxAvlEx()
31 :     {
32 :         //  デストラクト中に OnNtcRemove() を発生させるには DelAllNodes() メソッドを実行します
33 :         DelAllNodes();
34 :     }
35 :     //  ノード削除通知
36 :     VO OnNtcRemove (UX key, VOP pNodeData, UI len, UI nest) override
37 :     {
38 :         SAjxCon::PrintF(TEXT("ノードが削除されました - key=%4u, len=%d, nest=%d, data='%s'¥n"), (UI)key, len, nest, ((PNODEDATA)pNodeData)->pTimStr);
39 :         AjcTFree(((PNODEDATA)pNodeData)->pTimStr);
40 :     }
41 :
42 :     //  ノード列挙通知
43 :     BOOL OnNtcNode  (UX key, VOP pNodeData, UI len, UI nest) override
44 :     {
45 :         PAJCAVLNODE pNode = ((PAJCAVLNODE)pNodeData) - 1;
46 :         SAjxCon::PrintF(TEXT("%-26s:%*u, %d¥n"), ((PNODEDATA)pNodeData)->pTimStr, nest * 6, (UI)key, pNode->bal);
47 :         return TRUE;
48 :     }
49 :     //  入力テキスト通知(ASCII)
50 :     VO      OnNtcStr(int argc, UT *argv[], C_UTP pTxt) override
51 :     {
52 :         //  入力テキスト表示
53 :         SAjxCon::PrintF(TEXT("%s¥n"), pTxt);
54 :         //  コマンド実行
55 :         if (argc >= 1) {
56 :             //  ノード挿入
57 :             if      (_tcsicmp(argv[0], TEXT("I")) == 0) {
58 :                 if (argc == 2) {
59 :                     if (!InsNode(_ttoi(argv[1]), &(m_Node.pTimStr = GetTimeStr()), sizeof m_Node)) {
60 :                         if (m_Node.pTimStr != NULL) AjcTFree(m_Node.pTimStr); m_Node.pTimStr = NULL;
61 :                         SAjxCon::PrintF(TEXT("ノードの挿入を失敗しました。¥n"));
62 :                     }
63 :                 }
64 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
65 :             }
66 :             //  ノード置換
67 :             else if (_tcsicmp(argv[0], TEXT("R")) == 0) {
68 :                 if (argc == 2) {
69 :                     if (!RepNode(_ttoi(argv[1]), &(m_Node.pTimStr = GetTimeStr()), sizeof m_Node)) {
70 :                         if (m_Node.pTimStr != NULL) AjcTFree(m_Node.pTimStr); m_Node.pTimStr = NULL;
71 :                         SAjxCon::PrintF(TEXT("ノードの置換を失敗しました。¥n"));
72 :                     }
73 :                 }
74 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
75 :             }
76 :             //  ノードの挿入／置換
77 :             else if (_tcsicmp(argv[0], TEXT("X")) == 0) {
78 :                 if (argc == 2) {
79 :                     if (!InsOrRepNode(_ttoi(argv[1]), &(m_Node.pTimStr = GetTimeStr()), sizeof m_Node)) {
80 :                         if (m_Node.pTimStr != NULL) AjcTFree(m_Node.pTimStr); m_Node.pTimStr = NULL;
81 :                         SAjxCon::PrintF(TEXT("ノードの挿入／置換を失敗しました。¥n"));
82 :                     }
83 :                 }
84 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
85 :             }
86 :             //  ノード削除
87 :             else if (_tcsicmp(argv[0], TEXT("D"))  == 0) {
88 :                 if (argc == 2) {
89 :                     if (!DelNode(_ttoi(argv[1]))) {
```

```
90 :                         SAjxCon::PrintF(TEXT("ノードの削除を失敗しました。¥n"));
91 :                     }
92 :                 }
93 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
94 :             }
95 :             //  全ノード破棄
96 :             else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
97 :                 if (!DelAllNodes()) {
98 :                     SAjxCon::PrintF(TEXT("全ノードの削除を失敗しました。¥n"));
99 :                 }
100:             }
101:             //  全ノード表示
102:             else if (_tcsicmp(argv[0], TEXT("P"))  == 0) {
103:                 EnumNodes(TRUE);
104:             }
105:             //  その他
106:             else {
107:                 SAjxCon::PrintF(TEXT("*** Invalid command.¥n"));
108:             }
109:         }
110:     }
111: 
112: private:
113:     //  現在時刻文字列生成 (現在時刻文字列を格納した、動的メモリへのポインタを返す)
114:     UTP     GetTimeStr()
115:     {
116:         UTP         rc;
117:         SYSTEMTIME  st;
118:         UI          len;
119:         UT          szTim[64];
120: 
121:         GetLocalTime(&st);
122:         AjcSnPrintF(szTim, AJCTSIZE(szTim), TEXT("%02d:%02d:%02d.%03d"), st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
123:         len = (UI)_tcslen(szTim) + 1;
124:         MAjcStrCpy(rc = AjcTAlloc(len), len, szTim);
125:         return rc;
126:     }
127: };
128: 
129: //---- m a i n ---------------------------------------------------------------------------------//
130: int AjcMain()
131: {
132:     CAjxAvlEx avl;
133: 
134:     SAjxCon::SetStdMode();
135: 
136:     //  コマンドメニュー
137:     SAjxCon::PrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
138:     SAjxCon::PrintF(TEXT("  I   nnn     : ノード挿入, key=nnn¥n"));
139:     SAjxCon::PrintF(TEXT("  R   nnn     : ノード置換, key=nnn¥n"));
140:     SAjxCon::PrintF(TEXT("  X   nnn     : ノードの挿入／置換, key=nnn¥n"));
141:     SAjxCon::PrintF(TEXT("  D   nnn     : ノード削除, key=nnn¥n"));
142:     SAjxCon::PrintF(TEXT("  A           : 全ノードを破棄¥n"));
143:     SAjxCon::PrintF(TEXT("  P           : 全ノード表示"));
144:     SAjxCon::PrintF(TEXT("  <ESC>キー   : 終了¥n"));
145:     SAjxCon::PrintF(TEXT("¥n"));
146: 
147:     //  初期ノード表示
148:     avl.EnumNodes(TRUE);
149:     //  コマンドを入力しメニュー処理実行
150:     SAjxCon::PrintF(TEXT("Input(%3u) - "), avl.GetCount());
151:     while (avl.Input(TEXT(""))) {
152:         SAjxCon::PrintF(TEXT("Input(%3u) - "), avl.GetCount());
153:     }
154:     return 0;
155: }
```

2.17. 平衡２分木・文字列キー(CAjx Avs)

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxAvs(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxAvs(BOOL fUnicode = FALSE); // コンストラクタ(UNICODE)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	BOOL InsNode (C_UTP key, C_VOP pNodeData, UI len);	ノード挿入
2	int GetNode (C_UTP key, VOP pBuf, UI lBuf);	ノード取得
3	VOP GetNodePtr (C_UTP key, UIP pLen = NULL);	ノードアドレス取得
4	BOOL RepNode (C_UTP key, C_VOP pNodeData, UI len);	ノード置換
5	BOOL InsOrRepNode (C_UTP key, C_VOP pNodeData, UI len);	ノード挿入／置換
6	BOOL DelNode (C_UTP key)	ノード削除
7	BOOL DelAllNodes ();	全ノードを削除
8	UI GetCount ();	ノード数取得
9	UI EnumNodes (BOOL fDownSeq = FALSE);	全ノードのシーケンシャルな読み出し（列挙）
10	BOOL CreatePtrArr (PCAJCAVSPTR *ppArr, UIP pNum = NULL, BOOL fDownSeq = FALSE);	全ノードへのポインタ配列生成 ・ppArr は配列へのポインタを格納する変数のアドレス ・戻り値＝TRUE:成功, FALSE:失敗
11	BOOL ReleasePtrArr (PCAJCAVSPTRA pArr);	全ノードへのポインタ配列解放
12	BOOL EnableMultiThread (BOOL fEnable);	マルチスレッドの許可／禁止

仮想関数

#	変数／関数形式	内容	
1	virtual VO OnNtcRemove (C_UTP key, VOP pNodeData, UI len, UI nest);	ノード削除通知 （※１）	cbRemove
2	virtual BOOL OnNtcNode (C_UTP key, VOP pNodeData, UI len, UI nest);	ノード読み出し通知	cbNtcNode

※ cb*XXXXX*は、対応するコールバック関数を示します。

※１：残留ノードがある状態でデストラクタが実行された場合、残留ノードは全て破棄されますが、デストラクト中に OnNtcRemove() は実行されません。
デストラクト中に OnNtcRemove() を実行したい場合は、デストラクタ内で DelAllNode() メソッドを実行してください。

使用例

以下のサンプルプログラムは、コマンド入力により、２文木にノード挿入や削除等を行います。
ノードデータは、時刻文字列へのポインタです。
時刻文字列は動的なメモリに格納し、ノードの消去通知時に解放します。
起動時は初期ノードとしてキー＝”AAA”, “BBB”, “CCC”, “DDD”, “EEE”, “FFF” の６つのノードを挿入します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	I　<キー文字列>	ノード挿入, key=<キー文字列>
2	R　<キー文字列>	ノード置換, key=<キー文字列>
3	X　<キー文字列>	ノードの挿入／置換, key=<キー文字列>
4	D　<キー文字列>	ノード削除, key=<キー文字列>
5	A	全ノードを破棄
6	P	全ノード表示


```
1 : //
2 : //  SP_AVL.cpp
3 : //
4 : #include    <AjxCpp.h>
5 : #include    <tchar.h>
6 : using namespace AjxControl;
7 :
8 : //  ノードデータ形式
9 : typedef struct {
10 :     UTP     pTimStr;            //  ヒープ上の時刻文字列へのポインタ
11 : } NODEDATA, *PNODEDATA;
12 :
13 : //----- CAjxAvs, CAjxCon の派生クラス ----------------------------------------------------------------------//
14 : class CAjxAvsEx : public CAjxAvs, public CAjxCon
15 : {
16 : private:
17 :     NODEDATA    m_Node;
18 : public:
19 :     //  コンストラクタ（初期ノード登録）
20 :     CAjxAvsEx()
21 :     {
22 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited AAA")); InsNode(TEXT("AAA"), &m_Node, sizeof m_Node);
23 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited BBB")); InsNode(TEXT("BBB"), &m_Node, sizeof m_Node);
24 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited CCC")); InsNode(TEXT("CCC"), &m_Node, sizeof m_Node);
25 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited DDD")); InsNode(TEXT("DDD"), &m_Node, sizeof m_Node);
26 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited EEE")); InsNode(TEXT("EEE"), &m_Node, sizeof m_Node);
27 :         _tcscpy_s((m_Node.pTimStr = AjcTAlloc(16)), 16, TEXT("Pre-Inited FFF")); InsNode(TEXT("FFF"), &m_Node, sizeof m_Node);
28 :     }
29 :     //  デストラクタ
30 :     ~CAjxAvsEx()
31 :     {
32 :         //  デストラクト中に OnNtcRemove() を発生させるには DelAllNodes() メソッドを実行します
33 :         DelAllNodes();
34 :     }
```

```
35 :     //  ノード削除通知
36 :     VO OnNtcRemove (C_UTP pKey, VOP pNodeData, UI len, UI nest) override
37 :     {
38 :         SAjxCon::PrintF(TEXT("ノードが削除されました - key=%s, len=%d, nest=%d, data='%s'¥n"), pKey, len, nest, ((PNODEDATA)pNodeData)->pTimStr);
39 :         AjcTFree(((PNODEDATA)pNodeData)->pTimStr);
40 :     }
41 :
42 :     //  ノード列挙通知
43 :     BOOL OnNtcNode  (C_UTP pKey, VOP pNodeData, UI len, UI nest) override
44 :     {
45 :         PAJCAVLNODE pNode = ((PAJCAVLNODE)pNodeData) - 1;
46 :         SAjxCon::PrintF(TEXT("%-26s:%*s, %d¥n"), ((PNODEDATA)pNodeData)->pTimStr, nest * 6, pKey, pNode->bal);
47 :         return TRUE;
48 :     }
49 :     //  入力テキスト通知(ASCII)
50 :     VO      OnNtcStr(int argc, UT *argv[], C_UTP pTxt) override
51 :     {
52 :         //  入力テキスト表示
53 :         SAjxCon::PrintF(TEXT("%s¥n"), pTxt);
54 :         //  コマンド実行
55 :         if (argc >= 1) {
56 :             //  ノード挿入
57 :             if      (_tcsicmp(argv[0], TEXT("I")) == 0) {
58 :                 if (argc == 2) {
59 :                     if (!InsNode(argv[1], &(m_Node.pTimStr = GetTimeStr()), sizeof m_Node)) {
60 :                         if (m_Node.pTimStr != NULL) AjcTFree(m_Node.pTimStr); m_Node.pTimStr = NULL;
61 :                         SAjxCon::PrintF(TEXT("ノードの挿入を失敗しました。¥n"));
62 :                     }
63 :                 }
64 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
65 :             }
66 :             //  ノード置換
67 :             else if (_tcsicmp(argv[0], TEXT("R")) == 0) {
68 :                 if (argc == 2) {
69 :                     if (!RepNode(argv[1], &(m_Node.pTimStr = GetTimeStr()), sizeof m_Node)) {
70 :                         if (m_Node.pTimStr != NULL) AjcTFree(m_Node.pTimStr); m_Node.pTimStr = NULL;
71 :                         SAjxCon::PrintF(TEXT("ノードの置換を失敗しました。¥n"));
72 :                     }
73 :                 }
74 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
75 :             }
76 :             //  ノードの挿入／置換
77 :             else if (_tcsicmp(argv[0], TEXT("X")) == 0) {
78 :                 if (argc == 2) {
79 :                     if (!InsOrRepNode(argv[1], &(m_Node.pTimStr = GetTimeStr()), sizeof m_Node)) {
80 :                         if (m_Node.pTimStr != NULL) AjcTFree(m_Node.pTimStr); m_Node.pTimStr = NULL;
81 :                         SAjxCon::PrintF(TEXT("ノードの挿入／置換を失敗しました。¥n"));
82 :                     }
83 :                 }
84 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
85 :             }
86 :             //  ノード削除
87 :             else if (_tcsicmp(argv[0], TEXT("D"))  == 0) {
88 :                 if (argc == 2) {
89 :                     if (!DelNode(argv[1])) {
90 :                         SAjxCon::PrintF(TEXT("ノードの削除を失敗しました。¥n"));
91 :                     }
92 :                 }
93 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
94 :             }
95 :             //  全ノード破棄
96 :             else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
97 :                 if (!DelAllNodes()) {
98 :                     SAjxCon::PrintF(TEXT("全ノードの削除を失敗しました。¥n"));
99 :                 }
100 :            }
101 :            //  全ノード表示
102 :            else if (_tcsicmp(argv[0], TEXT("P"))  == 0) {
103 :                EnumNodes(TRUE);
104 :            }
105 :            //  その他
106 :            else {
```

```
107 :                 SAjxCon::PrintF(TEXT("*** Invalid command.¥n"));
108 :             }
109 :         }
110 :     }
111 :
112 : private:
113 :     //  現在時刻文字列生成 (現在時刻文字列を格納した、動的メモリへのポインタを返す)
114 :     UTP     GetTimeStr()
115 :     {
116 :         UTP         rc;
117 :         SYSTEMTIME  st;
118 :         UI          len;
119 :         UT          szTim[64];
120 :
121 :         GetLocalTime(&st);
122 :         AjcSnPrintF(szTim, AJCTSIZE(szTim), TEXT("%02d:%02d:%02d.%03d"), st.wHour, st.wMinute, st.wSecond, st.wMilliseconds);
123 :         len = (UI)_tcslen(szTim) + 1;
124 :         MAjcStrCpy(rc = AjcTAlloc(len), len, szTim);
125 :         return rc;
126 :     }
127 : };
128 :
129 : //----- m a i n ---------------------------------------------------------------------------------------------//
130 : int AjcMain()
131 : {
132 :     CAjxAvsEx avs;
133 :
134 :     SAjxCon::SetStdMode();
135 :
136 :     //  コマンドメニュー
137 :     SAjxCon::PrintF(TEXT("  I   SSSSS  : ノード挿入, key=¥"SSSSS¥"¥n"));
138 :     SAjxCon::PrintF(TEXT("  R   SSSSS  : ノード置換, key=¥"SSSSS¥"¥n"));
139 :     SAjxCon::PrintF(TEXT("  X   SSSSS  : ノードの挿入／置換, key=¥"SSSSS¥"¥n"));
140 :     SAjxCon::PrintF(TEXT("  D   SSSSS  : ノード削除, key=¥"SSSSS¥"¥n"));
141 :     SAjxCon::PrintF(TEXT("  A          : 全ノードを破棄¥n"));
142 :     SAjxCon::PrintF(TEXT("  P          : 全ノード表示¥n"));
143 :     SAjxCon::PrintF(TEXT("  <ESC>キー  : 終了¥n"));
144 :     SAjxCon::PrintF(TEXT("¥n"));
145 :
146 :     //  初期ノード表示
147 :     avs.EnumNodes(TRUE);
148 :     //  コマンドを入力しメニュー処理実行
149 :     SAjxCon::PrintF(TEXT("Input(%3u) - "), avs.GetCount());
150 :     while (avs.Input(TEXT(""))) {
151 :         SAjxCon::PrintF(TEXT("Input(%3u) - "), avs.GetCount());
152 :     }
153 :     return 0;
154 : }
```

2.18. 線形リスト制御（CAjxQue）

メンバ変数

#	変数／関数形式	内容
1	HAJCVQUE m_hQue;	可変長キュー インスタンスドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxQue();	コンストラクタ

メンバ関数

#	変数／関数形式	内容	
1	VO SetDefSize(UI DefSize);	ノードデータの規定バイト数設定	規定バイト数の初期値は0
2	UI GetDefSize();	ノードデータの規定バイト数取得	
3	VOP Enque (C_VOP pDat, UI lDat = -1);	ノードを末尾へ追加	lDat/lBuf = -1 の場合は、規定バイト数を仮定
4	VOP EnqTop (C_VOP pDat, UI lDat = -1);	ノードを先頭へ挿入	
5	UI Deque (VOP pBuf, UI lBuf = -1);	先頭ノード取り出し ノードデータのバイト数を返す(-1:ノード無し)	
6	BOOL Purge ();	全ノード消去	
7	UI GetCount();	ノードの個数取得	
8	BOOL Merge (CAjxQue* pSrc);	２つのリストをマージ	
9	VOP TopNode (UIP pBytes);	先頭ノードアドレス取得	
10	VOP NextNode(C_VOP pCurrentNode, UIP pBytes = NULL);	次のノードアドレス取得	
11	PCAJCVQUEPTR CreatePtrArr (UIP pNum = NULL);	全ノードへのポインタ配列生成	
12	VO ReleasePtrArr(PCAJCVQUEPTR pArr);	全ノードへのポインタ配列解放	
13	BOOL EnableMultiThread (BOOL fEnable);	マルチスレッドの許可／禁止	

仮想関数

#	変数／関数形式	内容	
1	virtual VO OnNtcRemove(VOP pNodeDat、UI len);	ノード削除通知 **（※１）**	cbRemove

※ cb*XXXXX*は、対応するコールバック関数を示します。

※１：残留ノードがある状態でデストラクタが実行された場合、残留ノードは全て破棄されますが、デストラクト中に OnNtcRemove() は実行されません。
デストラクト中に OnNtcRemove() を実行したい場合は、デストラクタ内で Purge() メソッドを実行してください。

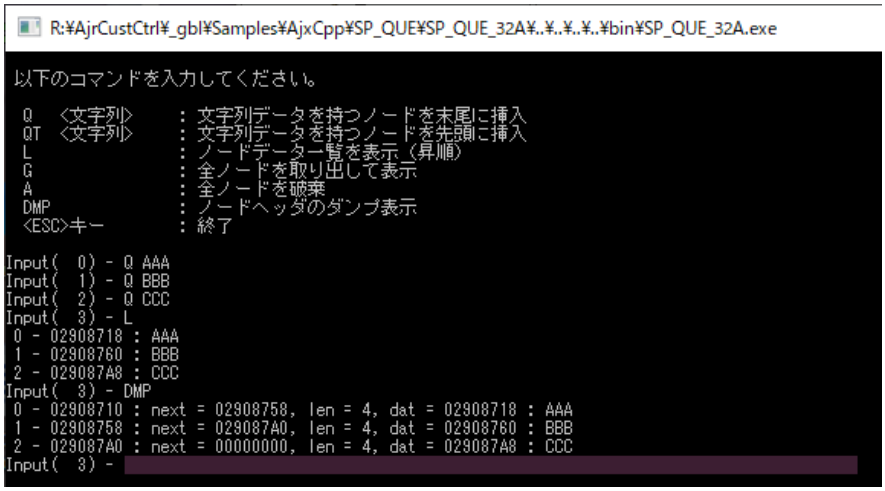
使用例

以下のサンプルプログラムは、コマンド入力により、双方向リストのノード挿入や削除等を行います。
ノードデータは、入力した文字列へのポインタです。
文字列は動的なメモリに格納し、ノードの消去通知時に解放します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	Q 〈文字列〉	文字列データを持つノードを末尾に挿入
2	QT 〈文字列〉	文字列データを持つノードを先頭に挿入
3	L	ノードデーター覧を表示（昇順）
4	G	全ノードを取り出して表示
5	A	全ノードを破棄
6	DMP	ノードヘッダのダンプ表示

※＜文字列＞に空白を含む場合はダブルクォート(“)で囲みます（ex. “A B C”）

実行イメージ




```
 1 : //
 2 : //  SP_QUE.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 :
 7 : using namespace AjxControl;
 8 :
 9 : static const UTP IniTxt[] = {TEXT("200"), TEXT("300"), TEXT("400")};
10 :
11 : //  ノードデータ
12 : typedef struct {
13 :     UTP     pStr;
14 : } NODE, *PNODE;
15 :
16 : class CAjxQueEx : public CAjxQue, public CAjxCon
17 : {
18 : public:
19 :     //  コンストラクタ（ノードのデフォルトサイズ設定）
20 :     CAjxQueEx() {
21 :         SetDefSize(sizeof(NODE));
22 :     }
23 :     //  デストラクタ
24 :     ~CAjxQueEx()
25 :     {
26 :         //  デストラクト中に OnNtcRemove() を発生させるには Purge() メソッドを実行します
27 :         Purge();
28 :     }
29 :     //  ノード消去通知
30 :     VO OnNtcRemove  (VOP pNodeData, UI len) override
31 :     {
32 :         PNODE   pNode = (PNODE)pNodeData;
33 :         SAjxCon::PrintF(TEXT(" ノード削除 %p -> %p : %s, len = %d¥n"), pNode, pNode->pStr, pNode->pStr, len);
34 :         AjcTFree(pNode->pStr);
35 :     }
36 :     //  入力テキスト通知(ASCII)
37 :     VO      OnNtcStr(int argc, UT *argv[], C_UTP pTxt)
38 :     {
39 :         UI      bytes, len, seq;
40 :         NODE    node;
41 :         PNODE   pNode;
42 :
43 :         //  入力テキスト表示
44 :         SAjxCon::PrintF(TEXT("%s¥n"), pTxt);
```

```
        //  コマンド実行
        if (argc >= 1) {
            //  文字列へのポインタを持つノードを末尾に挿入
            if      (_tcsicmp(argv[0], TEXT("Q")) == 0) {
                if (argc == 2) {
                    node.pStr = AjcTAlloc(len = (UI)_tcslen(argv[1]) + 1);
                    _tcscpy_s(node.pStr, len, argv[1]);
                    Enque(&node);
                }
                else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
            }
            //  文字列へのポインタを持つノードを先頭に挿入
            else if (_tcsicmp(argv[0], TEXT("QT")) == 0) {
                if (argc == 2) {
                    node.pStr = AjcTAlloc(len = (UI)_tcslen(argv[1]) + 1);
                    _tcscpy_s(node.pStr, len, argv[1]);
                    EnqTop(&node);
                }
                else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
            }
            //  ノードデータ一覧を表示
            else if (_tcsicmp(argv[0], TEXT("L")) == 0) {
                seq = 0;
                if (pNode = (PNODE)TopNode(&bytes)) {
                    do {
                        SAjxCon::PrintF(TEXT("%2d - %p -> %p : %s¥n"), seq++, pNode, pNode->pStr, pNode->pStr);
                    } while(pNode = (PNODE)NextNode(pNode, &bytes));
                }
            }
            //  全ノードを取り出して表示
            else if (_tcsicmp(argv[0], TEXT("G"))  == 0) {
                seq = 0;
                while ((Deque(&node, sizeof node)) != -1) {
                    SAjxCon::PrintF(TEXT("%2d - %p : %s¥n"), seq++, node.pStr, node.pStr);
                }
            }
            //  全ノードを破棄
            else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
                Purge();
            }
            //  ノードヘッダのダンプ表示
            else if (_tcsicmp(argv[0], TEXT("DMP"))  == 0) {
                seq = 0;
                if (pNode = (PNODE)TopNode(&bytes)) {
                    do {
                        PAJCVQNODE pH = ((PAJCVQNODE)pNode) - 1;
                        SAjxCon::PrintF(TEXT("%2d - %p : next = %p, len = %d, dat = %p -> %p : %s¥n"),
                                                            seq++, pH, pH->pNxt, pH->len, pNode, pNode->pStr, pNode->pStr);
                    } while(pNode = (PNODE)NextNode(pNode, &bytes));
                }
            }
            //  その他
            else {
                SAjxCon::PrintF(TEXT("*** Invalid command.¥n"));
            }
        }
    }
};

int AjcMain()
{
    CAjxQueEx   que;

    SAjxCon::SetStdMode();
    SAjxCon::SetBufSize(128, 64);
    SAjxCon::SetWndRect(0, 0, 127, 30);

    //  コマンドメニュー
    SAjxCon::PrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
    SAjxCon::PrintF(TEXT("  Q   <文字列>     : 文字列データを持つノードを末尾に挿入¥n"));
    SAjxCon::PrintF(TEXT("  QT  <文字列>     : 文字列データを持つノードを先頭に挿入¥n"));
    SAjxCon::PrintF(TEXT("  L                : ノードデータ一覧を表示 (昇順) ¥n"));
    SAjxCon::PrintF(TEXT("  G                : 全ノードを取り出して表示¥n"));
    SAjxCon::PrintF(TEXT("  A                : 全ノードを破棄¥n"));
    SAjxCon::PrintF(TEXT("  DMP              : ノードヘッダのダンプ表示¥n"));
    SAjxCon::PrintF(TEXT("  <ESC>キー        : 終了¥n"));
    SAjxCon::PrintF(TEXT("¥n"));

    //  コマンドを入力しメニュー処理実行
    SAjxCon::PrintF(TEXT("Input(%3u) - "), que.GetCount());
    while (que.Input(TEXT(""))) {
        SAjxCon::PrintF(TEXT("Input(%3u) - "), que.GetCount());
    }

    return 0;
}
```

2.19. 双方向リスト制御 (CAjxXQue)

メンバ変数

#	変数／関数形式	内容
1	HAJCVQUE m_hQue;	可変長キュー インスタンスドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxQue();	コンストラクタ

メンバ関数

#	変数／関数形式	内容	
1	VO SetDefSize(UI DefSize);	ノードデータの規定バイト数設定	規定バイト数の初期値は 0
2	UI GetDefSize();	ノードデータの規定バイト数取得	
3	VOP Enque (C_VOP pDat, UI lDat = -1);	ノードを末尾へ追加	lDat/lBuf = -1 の場合は、規定バイト数を仮定
4	VOP EnqTop (C_VOP pDat, UI lDat = -1);	ノードを先頭へ挿入	
5	VOP Insert (C_VOP pDat, UI lDat = -1, C_VOP pIns = NULL);	ノードを挿入	
7	UI Deque (VOP pBuf, UI lBuf = -1);	先頭ノード取り出し ノードデータのバイト数を返す(-1:ノード無し)	
8	UI Remove (C_VOP pDat);	ノードを削除	
9	BOOL Purge ();	全ノード消去	
10	UI GetCount();	ノードの個数取得	
11	BOOL Merge (CAjxQue* pSrc);	２つのリストをマージ	
12	VOP TopNode (UIP pBytes);	先頭ノードアドレス取得	
13	VOP LastNode(UIP pBytes);	末尾ノードアドレス取得	
14	VOP NextNode(C_VOP pCurrentNode, UIP pBytes = NULL);	次のノードアドレス取得	
15	VOP PrevNode(C_VOP pCurrentNode, UIP pBytes = NULL);	直前のノードアドレス取得	
16	PCAJCVQUEPTR CreatePtrArr (UIP pNum = NULL);	全ノードへのポインタ配列生成	
17	VO ReleasePtrArr(PCAJCVQUEPTR pArr);	全ノードへのポインタ配列解放	
18	BOOL EnableMultiThread (BOOL fEnable);	マルチスレッドの許可／禁止	

仮想関数

#	変数／関数形式	内容	
1	virtual VO OnNtcRemove(VOP pNodeDat、UI len);	ノード削除通知 **(※１)**	cbRemove

※ cb*XXXXX* は、対応するコールバック関数を示します。

※１：残留ノードがある状態でデストラクタが実行された場合、残留ノードは全て破棄されますが、デストラクト中に OnNtcRemove() は実行されません。
デストラクト中に OnNtcRemove() を実行したい場合は、デストラクタ内で Purge() メソッドを実行してください。

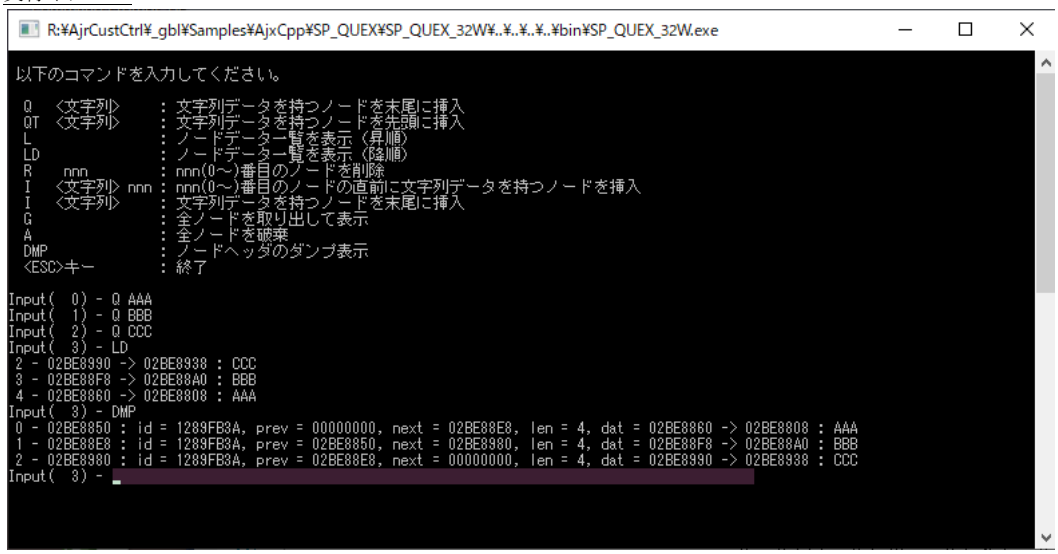
使用例

以下のサンプルプログラムは、コマンド入力により、双方向リストのノード挿入や削除等を行います。
ノードデータは、入力した文字列へのポインタです。
文字列は動的なメモリに格納し、ノードの消去通知時に解放します。
コマンドの形式は以下の通りです。 ESC キーを押すとプログラムを終了します。

#	コマンド形式	内容
1	Q 〈文字列〉	文字列データを持つノードを末尾に挿入
2	QT 〈文字列〉	文字列データを持つノードを先頭に挿入
3	L	ノードデーター覧を表示（昇順）
4	LD	ノードデーター覧を表示（降順）
5	R nnn	nnn(0～)番目のノードを削除
6	I 〈文字列〉 nnn	nnn(0～)番目のノードの直前に文字列データを持つノードを挿入
7	I 〈文字列〉	文字列データを持つノードを末尾に挿入
8	G	全ノードを取り出して表示
9	A	全ノードを破棄
10	DMP	ノードヘッダのダンプ表示

※＜文字列＞に空白を含む場合はダブルクォート(“)で囲みます（ex. “A B C”）

実行イメージ




```
 1 : //
 2 : //  SP_QUEX.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 :
 7 : using namespace AjxControl;
 8 :
 9 : static const UTP IniTxt[] = {TEXT("200"), TEXT("300"), TEXT("400")};
10 :
11 : //  ノードデータ
12 : typedef struct {
13 :     UTP     pStr;
14 : } NODE, *PNODE;
15 :
16 :
17 : class CAjxXQueEx : public CAjxXQue, public CAjxCon
18 : {
19 : public:
20 :     //  コンストラクタ（ノードのデフォルトサイズ設定）
21 :     CAjxXQueEx() {
22 :         SetDefSize(sizeof(NODE));
23 :     }
24 :     //  デストラクタ
25 :     ~CAjxXQueEx()
26 :     {
27 :         //  デストラクト中に OnNtcRemove() を発生させるには Purge() メソッドを実行します
28 :         Purge();
29 :     }
```

```
30 :     //  ノード消去通知
31 :     VO OnNtcRemove  (VOP pNodeData, UI len) override
32 :     {
33 :         PNODE   pNode = (PNODE)pNodeData;
34 :         SAjxCon::PrintF(TEXT(" ノード削除 %p -> %p : %s, len = %d¥n"), pNode, pNode->pStr, pNode->pStr, len);
35 :         AjcTFree(pNode->pStr);
36 :     }
37 :     //  入力テキスト通知(ASCII)
38 :     VO      OnNtcStr(int argc, UT *argv[], C_UTP pTxt)
39 :     {
40 :         UI      bytes, len, seq;
41 :         NODE    node;
42 :         PNODE   pNode;
43 :
44 :         //  入力テキスト表示
45 :         SAjxCon::PrintF(TEXT("%s¥n"), pTxt);
46 :         if (argc >= 1) {
47 :             //  文字列データを持つノードを末尾に挿入
48 :             if      (_tcsicmp(argv[0], TEXT("Q")) == 0) {
49 :                 if (argc == 2) {
50 :                     node.pStr = AjcTAlloc(len = (UI)_tcslen(argv[1]) + 1);
51 :                     _tcscpy_s(node.pStr, len, argv[1]);
52 :                     Enque(&node);
53 :                 }
54 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
55 :             }
56 :             //  文字列データを持つノードを先頭に挿入
57 :             else if (_tcsicmp(argv[0], TEXT("QT")) == 0) {
58 :                 if (argc == 2) {
59 :                     node.pStr = AjcTAlloc(len = (UI)_tcslen(argv[1]) + 1);
60 :                     _tcscpy_s(node.pStr, len, argv[1]);
61 :                     EnqTop(&node);
62 :                 }
63 :                 else SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
64 :             }
65 :             //  ノードデータ一覧を表示 (昇順)
66 :             else if (_tcsicmp(argv[0], TEXT("L")) == 0) {
67 :                 seq = 0;
68 :                 if (pNode = (PNODE)TopNode(&bytes)) {
69 :                     do {
70 :                         SAjxCon::PrintF(TEXT("%2d - %p -> %p : %s¥n"), seq++, pNode, pNode->pStr, pNode->pStr);
71 :                     } while(pNode = (PNODE)NextNode(pNode, &bytes));
72 :                 }
73 :             }
74 :             //  ノードデータ一覧を表示 (降順)
75 :             else if (_tcsicmp(argv[0], TEXT("LD")) == 0) {
76 :                 seq = GetCount()- 1;
77 :                 if (pNode = (PNODE)LastNode(&bytes)) {
78 :                     do {
79 :                         SAjxCon::PrintF(TEXT("%2d - %p -> %p : %s¥n"), seq++, pNode, pNode->pStr, pNode->pStr);
80 :                     } while(pNode = (PNODE)PrevNode(pNode, &bytes));
81 :                 }
82 :             }
83 :             //  nnn 番目のノードを削除
84 :             else if (_tcsicmp(argv[0], TEXT("R"))  == 0) {
85 :                 if (argc == 2 && (pNode = (PNODE)GetNodeAddr(_ttoi(argv[1])))) Remove(pNode);
86 :                 else                                                           SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));
87 :             }
88 :             //  nnn 番目のノードの直前／末尾にノードを挿入
89 :             else if (_tcsicmp(argv[0], TEXT("I"))  == 0) {
90 :                 node.pStr = AjcTAlloc(len = (UI)_tcslen(argv[1]) + 1);
91 :                 _tcscpy_s(node.pStr, len, argv[1]);
92 :                 if      (argc == 3 && (pNode = (PNODE)GetNodeAddr(_ttoi(argv[2])))) Insert(&node, (UI)sizeof node, pNode);
93 :                 else if (argc == 2                                                 ) Insert(&node, (UI)sizeof node, NULL);
94 :                 else  {AjcTFree(node.pStr); SAjxCon::PrintF(TEXT("*** Invalid parameter.¥n"));}
95 :             }
96 :             //  全ノードを取り出して表示
97 :             else if (_tcsicmp(argv[0], TEXT("G"))  == 0) {
98 :                 seq = 0;
99 :                 while ((Deque(&node, sizeof node)) != -1) {
100 :                    SAjxCon::PrintF(TEXT("%2d - %p : %s¥n"), seq++, node.pStr, node.pStr);
101 :                }
102 :            }
103 :            //  全ノードを破棄
104 :            else if (_tcsicmp(argv[0], TEXT("A"))  == 0) {
105 :                Purge();
106 :            }
107 :            //  ノードヘッダのダンプ表示
108 :            else if (_tcsicmp(argv[0], TEXT("DMP"))  == 0) {
109 :                seq = 0;
110 :                if (pNode = (PNODE)TopNode(&bytes)) {
```

```
111 :                     do {
112 :                         PAJCXQNODE pH = ((PAJCXQNODE)pNode) - 1;
113 :                         SAjxCon::PrintF(TEXT("%2d - %p : id = %08X, prev = %p, next = %p, ")
114 :                                         TEXT("len = %d, dat = %p -> %p : %s¥n"),
115 :                                         seq++, pH, pH->id, pH->pBfr, pH->pNxt, pH->len, pNode, pNode->pStr, pNode->pStr);
116 :                     } while(pNode = (PNODE)NextNode(pNode, &bytes));
117 :                 }
118 :             }
119 :             //  その他
120 :             else {
121 :                 SAjxCon::PrintF(TEXT("*** Invalid command.¥n"));
122 :             }
123 :         }
124 :     }
125 :     //  ノードアドレス取得
126 :     VOP     GetNodeAddr(UI ix)
127 :     {
128 :         VOP         rc = NULL;
129 :         UI          n;
130 :         MAJCXQUEPTR(PTRARR, UTP)
131 :         PPTRARR pArr = (PPTRARR)CreatePtrArr(&n);
132 :         if (ix < n) {
133 :             rc = (VOP)pArr[ix].pNode;
134 :         }
135 :         ReleasePtrArr((PCAJCXQUEPTR)pArr);
136 :         return rc;
137 :     }
138 : };
139 :
140 : int AjcMain()
141 : {
142 :     CAjxXQueEx  que;
143 :
144 :     SAjxCon::SetStdMode();
145 :     SAjxCon::SetBufSize(128, 64);
146 :     SAjxCon::SetWndRect(0, 0, 127, 30);
147 :
148 :     //  コマンドメニュー
149 :     SAjxCon::PrintF(TEXT("¥n 以下のコマンドを入力してください。¥n¥n"));
150 :     SAjxCon::PrintF(TEXT("  Q   <文字列>     : 文字列データを持つノードを末尾に挿入¥n"));
151 :     SAjxCon::PrintF(TEXT("  QT  <文字列>     : 文字列データを持つノードを先頭に挿入¥n"));
152 :     SAjxCon::PrintF(TEXT("  L                : ノードデータ一覧を表示 (昇順) ¥n"));
153 :     SAjxCon::PrintF(TEXT("  LD               : ノードデータ一覧を表示 (降順) ¥n"));
154 :     SAjxCon::PrintF(TEXT("  R    nnn         : nnn(0～)番目のノードを削除¥n"));
155 :     SAjxCon::PrintF(TEXT("  I   <文字列> nnn : nnn(0～)番目のノードの直前に文字列データを持つノードを挿入¥n"));
156 :     SAjxCon::PrintF(TEXT("  I   <文字列>     : 文字列データを持つノードを末尾に挿入¥n"));
157 :     SAjxCon::PrintF(TEXT("  G                : 全ノードを取り出して表示¥n"));
158 :     SAjxCon::PrintF(TEXT("  A                : 全ノードを破棄¥n"));
159 :     SAjxCon::PrintF(TEXT("  DMP              : ノードヘッダのダンプ表示¥n"));
160 :     SAjxCon::PrintF(TEXT("  <ESC>キー        : 終了¥n"));
161 :     SAjxCon::PrintF(TEXT("¥n"));
162 :
163 :     //  コマンドを入力しメニュー処理実行
164 :     SAjxCon::PrintF(TEXT("Input(%3u) - "), que.GetCount());
165 :     while (que.Input(TEXT(""))) {
166 :         SAjxCon::PrintF(TEXT("Input(%3u) - "), que.GetCount());
167 :     }
168 :
169 :     return 0;
170 : }
```

2.20. スレッド間メールデータ通信（CAjxMbx）

メンバ変数

#	変数／関数形式	内容
1	HAJCVMBX m_hMbx;	可変長メールボックス インスタンスドハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxMbx();	コンストラクタ

メンバ関数

#	変数／関数形式	内容	
1	VO SetDefSize(UI DefSize);	メールデータの規定バイト数設定	規定バイト数の初期値は 0
2	UI GetDefSize();	メールデータの規定バイト数取得	
3	BOOL Enque (C_VOP pDat, UI lDat = -1);	メールデータを末尾へ追加	lDat/lBuf = -1 の場合は、規定バイト数を仮定
4	BOOL EnqTop (C_VOP pDat, UI lDat = -1);	メールデータを先頭へ挿入	
5	UI Deque (VOP pBuf, UI lBuf = -1, UI MmsTime = 1000);	先頭メールデータ取り出し	
6	VO Purge ();	全メールデータ消去	
7	UI GetCount();	メールデータの個数取得	

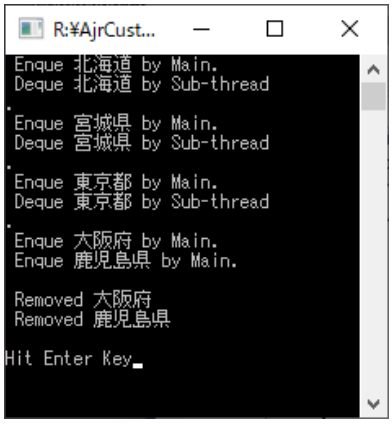
仮想関数

#	変数／関数形式	内容	
1	virtual VO OnNtcRemove(VOP pNodeDat);	メール削除通知 **（※１）**	cbRemove

※ cb*XXXXX* は、対応するコールバック関数を示します。

※１：残留メールデータがある状態でデストラクタが実行された場合、残留メールデータは全て破棄されますが、デストラクト中に OnNtcRemove() は実行されません。
デストラクト中に OnNtcRemove() を実行したい場合は、デストラクタ内で Purge() メソッドを実行してください。

使用例



サブスレッドを生成し、５つのメールデータをキューイングします。
サブスレッドは、３つのメールデータを取り出して表示後、サブスレッドを終了します。
サブスレッドが終了したら、残りの２つのメールデータを消去します。
消去するノードデータは、仮想関数(OnNtcRemove)へ通知されます。

```
 1 : //
 2 : //  SP_MBX.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : #include    <process.h>
 7 :
 8 : using namespace AjxControl;
 9 :
10 : static const UTP pTxt[] = {TEXT("北海道"), TEXT("宮城県"), TEXT("東京都"), TEXT("大阪府"), TEXT("鹿児島県")};
11 : static  BOOL    fThreadBusy = FALSE;
12 : static  CRITICAL_SECTION    cs;
13 :
14 : //  ノードデータ
15 : typedef struct {
16 :     UT      str[32];
17 : } NODE, *PNODE;
18 :
19 : //  CAjxMbx の派生クラス
20 : class CAjxMbxEx : public CAjxMbx
21 : {
22 :     //  ノード消去通知
23 :     VO OnNtcRemove  (VOP pNodeData) override
24 :     {   PNODE pNode = (PNODE)pNodeData;
25 :         SAjxCon::PrintF(TEXT(" Removed %s¥n"), pNode->str);
26 :     }
27 : };
```

```
28 :
29 : //----- メールデータ表示 ----------------------------------------------------------------------------------//
30 : VO      PrintData(PNODE pNode, C_UTP pAct, C_UTP pName)
31 : {
32 :     EnterCriticalSection(&cs);
33 :     SAjxCon::PrintF(TEXT(" %s %s by %s.¥n"), pAct, pNode->str, pName);
34 :     LeaveCriticalSection(&cs);
35 : }
36 :
37 : //----- サブスレッド --------------------------------------------------------------------------------------//
38 : VO      SubThread(VOP vop)
39 : {
40 :     CAjxMbx* p = (CAjxMbx*)vop;
41 :     NODE    node;
42 :
43 :     for (;;) {
44 :         //  メインスレッドからの通知データを表示
45 :         if (p->Deque(&node)) {
46 :             PrintData(&node, TEXT("Deque"), TEXT("Sub-thread¥n"));
47 :             //  データが "東京都" ならば、スレッド終了
48 :             if (_tcscmp(node.str, TEXT("東京都")) == 0) {
49 :                 break;
50 :             }
51 :         }
52 :     }
53 :     //  サブスレッド終了
54 :     fThreadBusy = FALSE;
55 :     _endthread();
56 : }
57 :
58 :
59 : //----- m a i n -------------------------------------------------------------------------------------------//
60 : int AjcMain()
61 : {
62 :     CAjxMbxEx   mbx;
63 :
64 :     SAjxCon::SetStdMode();
65 :
66 :     InitializeCriticalSection(&cs);
67 :
68 :     //  サブスレッド生成
69 :     fThreadBusy = TRUE;
70 :     _beginthread(SubThread, 0, (VOP)&mbx);
71 :     //  ノードのデフォルトサイズ設定
72 :     mbx.SetDefSize(sizeof(NODE));
73 :     //  ノード挿入
74 :     for (int i = 0; i < 5; i++) {
75 :         NODE        node;
76 :         _tcscpy_s(node.str, 32, pTxt[i]);
77 :         mbx.Enque(&node);
78 :         PrintData(&node, TEXT("Enque"), TEXT("Main"));
79 :         Sleep(100);
80 :     }
81 :     SAjxCon::PrintF(TEXT("¥n"));
82 :     //  サブスレッドの終了待ち
83 :     while (fThreadBusy) Sleep(1);
84 :     //  残ノード消去
85 :     mbx.Purge();
86 :
87 :     DeleteCriticalSection(&cs);
88 :
89 :     SAjxCon::PrintF(TEXT("¥nHit Enter Key"));
90 :     getchar();
91 :     return 0;
92 : }
```

2.21. C言語の字句分解（CAjxCtk）

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxCtk(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxCtk(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式	内容	
1	BOOL Reset ();	リセット	
2	BOOL SetFlag (UI flag);	機能フラグの設定	
3	UI GetFlag ();	機能フラグの取得	
4	BOOL GetToken (UTP pStrBuf, UI lStrBuf);	字句読み出し	
5	BOOL PeekToken (UTP pStrBuf, UI lStrBuf);	現在の字句取得	
6	BOOL Push ();	読み出し位置の退避	
7	BOOL Pop ();	読み出し位置の回復	
8	C_UTP GetTokenString (EAJCTKCODE tkn);	トークンコードに対応する文字列の取得	
9	EAJCTKCODE Token ();	トークンコード取得	AJCTK_TOKEN
10	EAJCTKSUF Suffix ();	数値定数のサフィックスコード取得	AJCTK_SUFFIX
11	UI Flag ();	フラグ情報取得	AJCTK_FLAG
12	UI Line ();	当該字句が存在する、ソースプログラム上の行番号取得（タブも１文字として計算）	AJCTK_LINE
13	UI Loc ();	当該字句が存在する、ライン上の桁位置を返します	AJCTK_LOC
14	UI Pos (UI TabStep);	当該字句が存在する、タブ文字(0x09)を考慮したライン上の桁位置を返します。(TabStep はタブステップ幅)	AJCTK_POS
15	UI Error ();	エラーコード取得	AJCTK_ERROR
16	BOOL IsUsrSym (EAJCTKCODE tkn);	トークンチェック（ユーザシンボル）	AJCTKIS_USRSYM
17	BOOL IsRsvSym (EAJCTKCODE tkn);	トークンチェック（予約名）	AJCTKIS_RSVSYM
18	BOOL IsValue (EAJCTKCODE tkn);	トークンチェック（数値定数）	AJCTKIS_VALUE
19	BOOL IsString (EAJCTKCODE tkn);	トークンチェック（文字列）	AJCTKIS_STRING
20	BOOL IsPathName (EAJCTKCODE tkn);	トークンチェック（#include のパス名）	AJCTKIS_PATHNAME
21	BOOL IsDelimit (EAJCTKCODE tkn);	トークンチェック（デリミタや演算子）	AJCTKIS_DELIMIT
22	BOOL IsSymbol (EAJCTKCODE tkn);	トークンチェック（シンボル）	AJCTKIS_SYMBOL
23	BOOL IsValSym (EAJCTKCODE tkn);	トークンチェック（シンボル／数値定数）	AJCTKIS_VALSYM
24	BOOL IsSpace (EAJCTKCODE tkn);	トークンチェック（空白）	AJCTKIS_SPACE

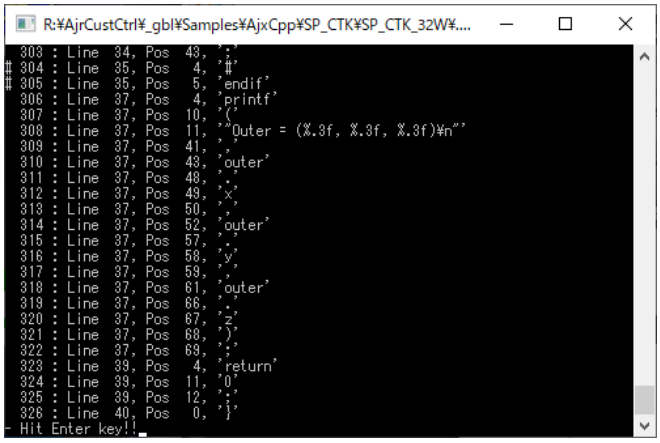
※ AJCTK_XXXXX / AJCTKIS_XXXXX は、対応するマクロを示します。

仮想関数

#	変数／関数形式	内容	
1	virtual BOOL OnGetLine (UTP pBuf, UI lBuf);	１行読み出し要求	cbGetLine

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例　コマンドラインの第一引数で指定したＣ言語ソースファイルを入力し、分解した字句を表示します。
プリプロセス文中の字句である場合は、先頭に「#」を表示します。




```
 1 : //
 2 : //  SP_CTK.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : using namespace AjxControl;
 7 : #define RED     RGB(255, 0, 0)
 8 : #define WHITE   RGB(255, 255, 255)
 9 :
10 : class CAjxCtkEx : public CAjxCtk, public CAjxFile
11 : {
12 :     UT          m_Path[MAX_PATH];
13 : public:
14 :     CAjxCtkEx(UTP pPath) {      //  コンストラクタ
15 :         _tcscpy_s(m_Path, MAX_PATH, pPath);
16 :         if (!AjcPathIsFile(pPath)) throw std::runtime_error("File not found.");
17 :     }
18 :     ~CAjxCtkEx()    {           //  デストラクタ
19 :         if (FIsActive()) {
20 :             FClose();
21 :         }
22 :     }
23 : private:
24 :     //  １行読み出し要求
25 :     BOOL  OnGetLine (UTP pBuf, UI lBuf) override
26 :     {
27 :         BOOL    rc = FALSE;
28 :         //  初回ならばファイルオープン
29 :         if (!FIsActive()) {
30 :             FOpen(m_Path);
31 :         }
32 :         //  １行読み出し，EOF ならばファイルクローズ
33 :         if (FIsActive()) {
34 :             if (FGetS(pBuf, lBuf) != NULL) rc = TRUE;
35 :             else                           FClose();
36 :         }
37 :         else throw std::runtime_error("File Open failure.");
38 :         return rc;
39 :     }
40 : };
41 : int AjcMain(int argc, UTP argv[])
42 : {
43 :     SAjxCon::SetStdMode();
44 :     try {
45 :         UI          cnt = 0;
46 :         UT          syl[256];
47 :         CAjxCtkEx   ctk(argv[1]);
48 :         while (ctk.GetToken(syl, AJCTSIZE(syl))) {
49 :             _tprintf(TEXT("%c%4d : Line %3d, Pos %3d, '%s'¥n"), (ctk.Flag() & AJCTKF_PREPRO) ? '#' : ' ',
50 :                                                                 ++cnt, ctk.Line(), ctk.Pos(4), syl);
51 :         }
52 :     }
53 :     catch (std::exception e) {
54 :         printf("Exception: %s¥n", e.what());
55 :     }
56 :
57 :     _tprintf(TEXT("- Hit Enter key!!"));
58 :     getchar();
59 :     return 0;
60 : }
```

2.22. C言語のプリコンパイル (CAjxPpc)

コンストラクタ

#	変数／関数形式	内容
1	CAjxPpc();	コンストラクタ

メンバ関数

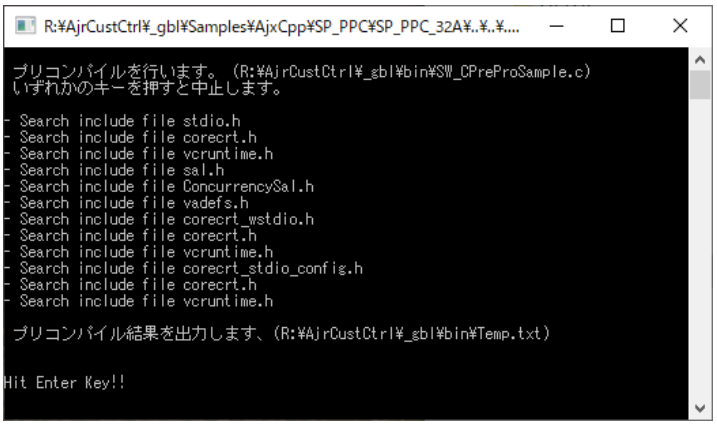
#	変数／関数形式	内容
1	AJCPPCRESULT PreCompile(C_BCP pSrcPath, EAJCTEC tecSrc = AJCTEC_AUTO, C_BCP pBasePath = NULL, PAJCLBXITEMA pIncPath = NULL, PAJCLBXITEMA pOptSym = NULL, UI opt = AJCPPC_FLG_AUTO_SEARCH | AJCPPC_FLG_GENALL););	プリコンパイル実行
2	PAJCPPCTKNNODE GetObject (PAJCPPCTKNNODE *ppNoGen);	プリコンパイル オブジェクト（トークンストリーム）の取得
3	BOOL Stop ();	プリコンパイル中止
4	BOOL GetMacroInfo (C_BCP pMacName, PAJCPPCMACINFO pMacInfo);	マクロ情報取得
5	UI EnumMacro ();	マクロ列挙
6	BOOL TokenStreamToFile (C_BCP pOutPath, EAJCTEC tecOut = AJCTEC_AUTO, BOOL bomOut = FALSE, BOOL fExpInc = TRUE, UI CommOutOfkndPP = AJCPPC_PPK_ALL);	プリコンパイル結果をファイル出力

仮想関数

#	変数／関数形式	内容
1	virtual VO OnNtcAnyEvt (AJCPPCNOTIFY evt);	いずれかのイベント発生
2	virtual VO OnNtcFileLno (C_BCP pFile, UI lno, UI nest);	ファイル名，行番号通知
3	virtual VO OnNtcSrhStart (C_BCP pIncFile, C_BCP pDirPath);	インクルードファイル検索開始通知
4	virtual VO OnNtcSrhDir (C_BCP pDirPath);	インクルードファイル検索中のフォルダ通知
5	virtual VO OnNtcSrhEnd (C_BCP pIncFile, BOOL fFind);	インクルードファイル検索終了通知
6	virtual VO OnNtcOptSym (PCAJCPPCTKNNODE pTkn);	プリプロセス用オプションシンボル通知
7	virtual VO OnNtcMacDef (PCAJCPPCTKNNODE pTkn, PCAJCPPCMACINFO pMac);	マクロ定義通知
8	virtual VO OnNtcMacRef (PCAJCPPCTKNNODE pTkn, PCAJCPPCMACINFO pMac);	マクロ参照通知
9	virtual VO OnNtcOutLoop (VO);	トークンストリームをファイルへ出力ループ中
10	virtual VO OnNtcSrcTec (C_BCP pFilePath, EAJCTEC tec, BOOL fBom);	ソースファイルのエンコード通知
11	virtual VO OnNtcToken (EAJCTKCODE tkn, C_BCP pTkn);	トークン通知
12	virtual VO OnNtcError (AJCPPCERROR err, C_BCP pFile, UI lno, C_BCP pErrTxt);	エラー通知
13	virtual BOOL OnNtcMacInfo (PCAJCPPCMACINFO pMacInfo);	マクロ列挙通知

使用例

実行ファイルと同じフォルダ中の「SW_CPreProSample.c」をプリコンパイルし、結果を同フォルダの「Temp.txt」へ出力します。




```
 1 : //
 2 : //  SP_PPC.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : #include    <conio.h>
 7 : using namespace AjxControl;
 8 :
 9 : static  AJCLBXITEM  OptSym[] = {{"_MSC_VER=1400"             }, {"_MSC_FULL_VER=140050727"},
10 :                               {"_INTEGRAL_MAX_BITS=64"     }, {"_WIN32"                 },
11 :                               {"_MBCS"                     }, {"_CRTBLD"                },
12 :                               {"_M_IX86"                   }, {"__STDC__=0"             },
13 :                               {"__STDC_WANT_SECURE_LIB__=0"}, {NULL                     },
14 : };
15 :
16 : static  AJCLBXITEM  IncPath[] = {{"C:¥¥Program Files (x86)¥¥Microsoft Visual Studio*.*"},
17 :                                {"C:¥¥Program Files (x86)¥¥Microsoft SDKs¥¥*.*"       },
18 :                                {"C:¥¥Program Files (x86)¥¥Windows Kits¥¥*.*"         },
19 :                                {NULL}
20 : };
21 :
22 : class CAjxPpcEx : public CAjxPpc
23 : {
24 :     //  いずれかのイベント発生
25 :     VO      OnNtcAnyEvt     (AJCPPCNOTIFY evt)                                  override {
26 :         if (_kbhit()) Stop();
27 :     }
28 :     //  ファイル名，行番号通知
29 :     VO      OnNtcFileLno    (C_BCP pFile, UI lno, UI nest)                      override {
30 :     }
31 :     //  インクルードファイル検索開始通知
32 :     VO      OnNtcSrhStart   (C_BCP pIncFile, C_BCP pDirPath)                    override {
33 :         SAjxCon::PrintF("- Search include file %s¥n", pIncFile);
34 :     }
35 :     //  インクルードファイル検索中のフォルダ通知
36 :     VO      OnNtcSrhDir     (               C_BCP pDirPath)                     override {
37 :     }
38 :     //  インクルードファイル検索終了通知
39 :     VO      OnNtcSrhEnd     (C_BCP pIncFile, BOOL fFind)                        override {
40 :     }
41 :     //  プリプロセス用オプションシンボル通知
42 :     VO      OnNtcOptSym     (PCAJCPPCTKNNODE pTkn)                              override {
43 :     }
44 :     //  マクロ定義通知
45 :     VO      OnNtcMacDef     (PCAJCPPCTKNNODE pTkn, PCAJCPPCMACINFO pMac)        override {
46 :     }
47 :     //  マクロ参照通知
48 :     VO      OnNtcMacRef     (PCAJCPPCTKNNODE pTkn, PCAJCPPCMACINFO pMac)        override {
49 :     }
50 :     //  トークンストリームをファイルへ出力ループ中
51 :     VO      OnNtcOutLoop    (VO)                                                override {
52 :     }
53 :     //  ソースファイルのエンコード通知
54 :     VO      OnNtcSrcTec     (C_BCP pFilePath, EAJCTEC tec, BOOL fBom)           override {
55 :     }
```

```
56 :     //  トークン通知
57 :     VO      OnNtcToken      (EAJCTKCODE tkn, C_BCP pTkn)                            override {
58 :     }
59 :     //  エラー通知
60 :     VO      OnNtcError      (AJCPPCERROR err, C_BCP pFile, UI lno, C_BCP pErrTxt)   override {
61 :         SAjxCon::PrintF("%s¥n", pErrTxt);
62 :     }
63 :     //  マクロ列挙通知
64 :     BOOL    OnNtcMacInfo    (PCAJCPPCMACINFO pMacInfo)                              override {
65 :         return TRUE;
66 :     }
67 : };
68 :
69 : int AjcMain()
70 : {
71 :     CAjxPpcEx ppc;
72 :
73 :     BC      src[MAX_PATH];
74 :     BC      out[MAX_PATH];
75 :
76 :     SAjxCon::SetStdMode();
77 :     SAjxCon::SetBufSize(160, 256);
78 :     SAjxCon::SetWndRect(0, 0, 159, 30);
79 :
80 :     AjcGetAppPath(src, sizeof src);
81 :     strcat_s(src, sizeof src, "SW_CPreProSample.c");
82 :     SAjxCon::PrintF("¥n プリコンパイルを行います。 (%s)¥n いずれかのキーを押すと中止します。¥n¥n", src);
83 :     //  プリコンパイル実行
84 :     if (ppc.PreCompile(src,                                 //  ソースファイル
85 :                        AJCTEC_AUTO,                         //  ソースファイルのテキストコード
86 :                        NULL,                                //  ベースパス
87 :                        IncPath,                             //  インクルードパス
88 :                        OptSym) == AJCPPCR_OK) {             //  オプションシンボル
89 :         //  ファイル出力
90 :         AjcGetAppPath(out, sizeof out);
91 :         strcat_s(out, sizeof out, "Temp.txt");
92 :         SAjxCon::PrintF("¥n プリコンパイル結果を出力します、(%s)¥n¥n", out);
93 :         ppc.TokenStreamToFile(out,                          //  出力ファイル
94 :                               AJCTEC_AUTO,                  //  出力ファイルのテキストコード
95 :                               FALSE,                        //  出力ファイルのＢＯＭなし
96 :                               TRUE,                         //  インクルードの展開内容を含める
97 :                               AJCPPC_PPK_ALL);              //  全プリプロセス文出力
98 :     }
99 :
100 :     SAjxCon::PrintF("¥nHit Enter Key!!");
101 :     getchar();
102 :     return 0;
103 : }
```

2.23. 文字列プール (CAjxSpl)

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxSpl(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxSpl(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	BOOL SetCompMode (EAJCCMPMODE CmpMode);	比較モード設定
2	EAJCCMPMODE GetCompMode ();	比較モード取得
3	C_UTP Regist (C_UTP pStr);	文字列プールへ文字列登録
4	C_UTP Find (C_UTP pStr);	文字列プールから文字列を検索
5	C_UTP PartStrInPool(C_UTP pPartStr, AJCSPL_INSTROPT opt = AJCSPL_ISP_MATCHFIRST);	部分文字列一致検索 （文字列プールから指定部分文字列が一致するノード検索）
6	C_UTP PoolStrInStr (C_UTP pStr, AJCSPL_INSTROPT opt = AJCSPL_ISP_MATCHFIRST);	部分文字列一致検索 （指定文字列が文字列プール中の部分文字列と一致するかチェック）
7	BOOL Remove (C_UTP pStr);	文字列プールから文字列を削除
8	UI GetCount ();	登録済文字列数取得
9	BOOL Reset ();	文字列プールをリセット
10	UI EnumStr (BOOL fDownSeq = FALSE);	登録済み全文字列の取得(列挙)
11	HAJCAVL GetAvlHandle ();	ＡＶＬ２分木のハンドルを取得

仮想関数

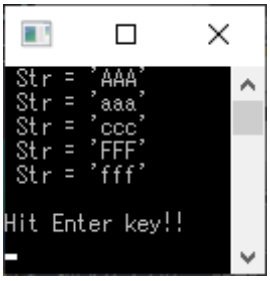
#	変数／関数形式	内容	
1	virtual BOOL OnNtcStr (C_UTP pStr);	列挙文字列通知	cbNtcStr

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例

７つの文字列 ("AAA", "FFF", "CCC", "aaa", "fff", "ccc") を登録し、文字列 ("CCC") を削除後、全文字列を列挙表示します。

```
 1 : //
 2 : //  SP_SPL.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : using namespace AjxControl;
 7 :
 8 : class CAjxSplEx : public CAjxSpl
 9 : {
10 :     //  文字列列挙通知
11 :     BOOL OnNtcStr    (C_UTP pStr) override
12 :     {
13 :         _tprintf(TEXT(" Str = '%s'¥n"), pStr);
14 :         return TRUE;
15 :     }
16 : };
17 :
18 : int AjcMain()
19 : {
20 :     CAjxSplEx avs;
21 :
22 :     SAjxCon::SetStdMode();
23 :
24 :     //  文字列登録
25 :     avs.Regist(TEXT("AAA"));
26 :     avs.Regist(TEXT("FFF"));
27 :     avs.Regist(TEXT("CCC"));
28 :     avs.Regist(TEXT("aaa"));
29 :     avs.Regist(TEXT("fff"));
30 :     avs.Regist(TEXT("ccc"));
31 :     avs.Regist(TEXT("AAA"));
32 :     //  文字列消去
33 :     avs.Remove(TEXT("CCC"));
34 :     //  全文字列列挙
35 :     avs.EnumStr();
36 :
37 :     _tprintf(TEXT("¥nHit Enter key!!¥n"));
38 :     getchar();
39 :     return 0;
40 : }
```



2.24. ＬＺＨ書庫データのアクセス (CAjxLzh)

コンストラクタ

#	変数／関数形式	内容
1	CAjxLzh();	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	HAJCLZH FindFirst (C_UTP pWild, UI flag = AJCLZHF_SUBDIR);	書庫内のファイル名検索（初回）
2	BOOL FindNext (HAJCLZH hLzh);	書庫内のファイル名検索（２回目以降）
3	HAJCLZH Open (C_UTP pFilePath, UI flag = AJCLZHF_SUBDIR);	書庫内のファイルオープン
4	UI Read (HAJCLZH hLzh, VOP pBuf, UI len);	書庫内のファイル読み出し
5	VO Close (HAJCLZH hLzh);	書庫アクセスのクローズ
6	BOOL GetFileInfo (HAJCLZH hLzh, PAJCLZHFILEINFO pBuf);	検索したファイル情報取得
7	AJCLZHERR GetLastError(VO);	検索したファイル情報取得

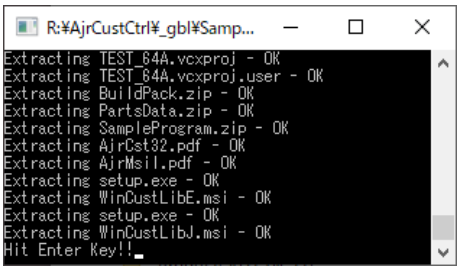
純粋仮想関数

#	変数／関数形式	内容	
1	virtual UI OnRead (VOP pBuf, UI lBuf) = 0;	ファイル読み出し要求	cbRead
2	virtual VO OnSeek (UL ofs) = 0;	ファイル読み出し位置設定要求	cbSeek

※ cb*XXXXX* は、対応するコールバック関数を示します。

使用例

コマンドラインで指定した LZH 書庫ファイルを解凍します。
解答したファイルは、指定した LZH 書庫ファイルと同じディレクトリ下に作成します。




```
 1 : //
 2 : //  SP_LZH.cpp
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : #include    <direct.h>
 7 : using namespace AjxControl;
 8 :
 9 : //----- CAjxLzh の派生クラス ------------------------------------------------------------------------------------//
10 : class CAjxLzhEx : public CAjxLzh
11 : {
12 : public:
13 :     FILE*   m_hFileLzh;
14 :     //  コンストラクタ
15 :     CAjxLzhEx()
16 :     {
17 :         m_hFileLzh = NULL;
18 :     }
19 :     //  LZH ファイルオープン
20 :     errno_t OpenLzhFile(UTP pPath)
21 :     {
22 :         return _tfopen_s(&m_hFileLzh, pPath, TEXT("rb"));
23 :     }
24 :     //  LZH ファイルクローズ
25 :     VO      CloseLzhFile()
26 :     {
27 :         if (m_hFileLzh != NULL) {
28 :             fclose(m_hFileLzh);
29 :             m_hFileLzh = NULL;
30 :         }
31 :     }
32 :     //  ファイル読み出し要求
33 :     UI      OnRead      (VOP pBuf, UI lBuf) override
34 :     {
35 :         return (UI)fread(pBuf, 1, lBuf, m_hFileLzh);
36 :     }
37 :     //  ファイル読み出し位置設定要求
38 :     VO      OnSeek      (UL  ofs)   override
39 :     {
40 :         fseek(m_hFileLzh, ofs, SEEK_SET);
41 :     }
42 : };
43 :
44 : CAjxLzhEx   lzh;
45 :
46 : //----- m a i n ------------------------------------------------------------------------------------------------//
47 : int AjcMain(int argc, UTP argv[])
48 : {
49 :     UW          crc;                    //  ＣＲＣ算出ワーク
50 :     AJCLZHERR   err;                    //  ＬＺＨデコード エラーコード
51 :     HAJCLZH     hLzhSearch = NULL;      //  ＬＺＨハンドル（検索用）
52 :     HAJCLZH     hLzhRead   = NULL;      //  ＬＺＨハンドル（読出用）
```

```
 53 :     FILE        *hFileLzh  = NULL;          //  ＬＺＨ書庫ファイル・ハンドル
 54 :     FILE        *hFileOut  = NULL;          //  出力ファイルハンドル
 55 :     UT          szPath    [MAX_PATH];       //  パス編集ワーク
 56 :     UT          szTopDir  [MAX_PATH];       //  先頭フォルダパス
 57 :     BOOL        fIgnoreDir = TRUE;          //  ディレクトリ無視フラグ
 58 :     AJCLZHFILEINFO  FInfo;                  //  ファイル情報
 59 :     UT          drv[_MAX_DRIVE], dir[_MAX_DIR], fname[_MAX_FNAME], fext[_MAX_EXT];
 60 :     UI          rbytes;                     //  読み出したバイト数
 61 :     UB          buf[1024];                  //  読み出しバッファ
 62 :
 63 :     SAjxCon::SetStdMode();
 64 :
 65 :     //  ＬＺＨ書庫(.lzh)ファイルオープン
 66 :     if (argc >= 2 && lzh.OpenLzhFile(argv[1]) == 0) {
 67 :         //  トップフォルダ名設定
 68 :         MAjcSplitPath(argv[1] , drv, dir, fname, fext);
 69 :         MAjcMakePath (szTopDir, drv, dir, NULL , NULL);
 70 :         //  書庫内のファイル検索初期化
 71 :         if ((hLzhSearch = lzh.FindFirst(TEXT("*.*"))) != NULL) {
 72 :             //  書庫内の全ファイルループ
 73 :             do {
 74 :                 //----- ファイル情報取得 ----------------------------------------------------------------------------//
 75 :                 lzh.GetFileInfo(hLzhSearch, &FInfo);
 76 :                 //----- フォルダ作成 --------------------------------------------------------------------------------//
 77 :                 if (FInfo.szDirName[0] != 0) {
 78 :                     UTP     p;
 79 :                     UT      tmp [MAX_PATH];
 80 :                     MAjcStrCpy(szPath, AJCTSIZE(szPath), szTopDir);
 81 :                     MAjcStrCpy(tmp, AJCTSIZE(tmp), FInfo.szDirName);
 82 :                     if (p = MAjcStrTok(tmp, TEXT("¥¥"))) {
 83 :                         do {
 84 :                             AjcPathCat(szPath, p, MAX_PATH);
 85 :                              _tmkdir(szPath);
 86 :                         } while (p = MAjcStrTok(NULL, TEXT("¥¥")));
 87 :                     }
 88 :                 }
 89 :                 //----- ファイル出力 --------------------------------------------------------------------------------//
 90 :                 if (hLzhRead = lzh.Open(FInfo.szPathName, 0)) {
 91 :                     SAjxCon::PrintF(TEXT("Extracting %s - "), FInfo.szFileName);
 92 :                     MAjcStrCpy(szPath, MAX_PATH, szTopDir);
 93 :                     AjcPathCat(szPath, FInfo.szPathName, MAX_PATH);
 94 :                     if (_tfopen_s(&hFileOut, szPath, TEXT("wb")) == 0) {
 95 :                         //  ファイル出力とＣＲＣ算出
 96 :                         crc = 0;
 97 :                         while (rbytes = lzh.Read(hLzhRead, buf, sizeof buf)) {
 98 :                             crc = AjcPartCrc16R(buf, rbytes, crc);
 99 :                             fwrite(buf, 1, rbytes, hFileOut);
100 :                         }
101 :                         //  ＣＲＣチェック
102 :                         if ((err = lzh.GetLastError()) == AJCLZHERR_OK) {
103 :                             if (crc == hLzhRead->crc) SAjxCon::PrintF(TEXT("OK¥n"));
104 :                             else                      SAjxCon::PrintF(TEXT("NG (CRC error)¥n"));
105 :                         }
106 :                         else {
107 :                             SAjxCon::PrintF(TEXT("NG (err = 0x%04X)¥n"), err);
108 :                         }
109 :                         //  出力ファイル・クローズ
110 :                         fclose(hFileOut);
111 :                     }
112 :                     else {
113 :                         SAjxCon::PrintF(TEXT("NG (Creation failure)¥n"));
114 :                     }
115 :                     // ＬＺＨ読み出しハンドル・クローズ
116 :                     lzh.Close(hLzhRead);
117 :                 }
118 :                 else {
119 :                     SAjxCon::PrintF(TEXT("File not found (%s)¥n"), FInfo.szPathName);
120 :                 }
121 :             } while(lzh.FindNext(hLzhSearch));
122 :             // ＬＺＨ検索ハンドル・クローズ
123 :             lzh.Close(hLzhSearch);
124 :         }
125 :         else {
126 :             SAjxCon::PrintF(TEXT("ＬＺＨ書庫内にファイルが見つかりません¥n"));
127 :         }
128 :         //  ＬＺＨ書庫(.lzh)ファイル クローズ
129 :         lzh.CloseLzhFile();
130 :     }
131 :     else {
132 :         SAjxCon::PrintF(TEXT("ＬＺＨ書庫ファイルのオープン失敗¥n"));
133 :     }
134 :
135 :     SAjxCon::PrintF(TEXT("Hit Enter Key!!"));
136 :     getchar();
137 :     return 0;
138 : }
```

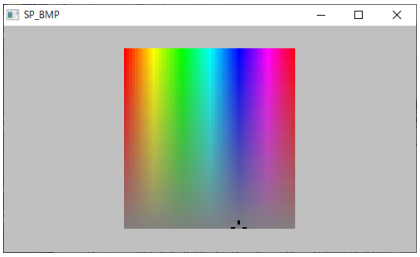
2.25. ビットマップ操作（SAjxBmp）

メンバ関数（スタティック関数）

#	変数／関数形式	内容
1	static BOOL ChangeBitmapColor (HBITMAP hBmp, COLORREF before, COLORREF after);	ビットマップの色変更
2	static BOOL SetBitmapToClipboard (HBITMAP hBitmap);	ビットマップデータをクリップボードへ格納する
3	static HBITMAP GetBitmapFromClipboard ();	クリップボードからビットマップデータを取得する
4	static BOOL WriteBitmapToFile (HBITMAP hBitmap, C_BCP pFilePath);	ビットマップファイルを作成する(ASCII)
5	static BOOL WriteBitmapToFile (HBITMAP hBitmap, C_WCP pFilePath);	ビットマップファイルを作成する(UNICODE)
6	static HBITMAP CreateDibFromFile (C_UTP pFilePath, UIP pWidth = NULL, UIP pHeight = NULL, PAJCBITMAPINFO pBmpInfo = NULL, VOP *ppImage = NULL);	ビットマップファイルを読み出してＤＩＢセクションを作成

使用例

自プログラム(.exe)と同一フォルダ上の、ビットマップファイル(SW_DIBSect.bmp)を読み出して表示します。




```
//
//  SP_BMP.c
//
#include    <AjxCpp.h>
#include    <tchar.h>
using namespace AjxControl;

static  HWND            hWndMain = NULL;
static  HBITMAP         hBmp     = NULL;
static  UI              width, height;
AJC_WNDPROC_DEF(Main);

//----- WinMain ----------------------------------------------------------------------------------------------//
int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
{
    MSG         msg = {0};
    WNDCLASS    wc  = {0};

    //  ウインド生成
    wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
    wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
    wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_BMP");
    RegisterClass(&wc);
    hWndMain = CreateWindow( TEXT("SP_BMP"), TEXT("SP_BMP"),    // window class name, caption
                    WS_OVERLAPPEDWINDOW & ~WS_THICKFRAME,       // window style
                    CW_USEDEFAULT, CW_USEDEFAULT, 500, 300,     // position, size
                    NULL, NULL, hInstance, NULL);               // parent, menu, instance, param
    ShowWindow(hWndMain, iCmdShow);
    //  メッセージループ
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage (&msg);
    }
    return (int)msg.wParam ;
}
//----- WM_CREATE --------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_CREATE     )
{
    UT      drv[_MAX_DRIVE], dir[_MAX_DIR];
    UT      path[MAX_PATH];

    AjcGetAppPath(path, MAX_PATH);
    MAjcSplitPath(path, drv, dir, NULL, NULL);
    MAjcMakePath (path, drv, dir, TEXT("SW_DIBSect"), TEXT(".bmp"));
    //  ビットマップファイル読み出し
    hBmp = SAjxBmp::CreateDibFromFile(path, &width, &height);
    return 0;
}
//----- WM_DESTROY -------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_DESTROY    )
{
    if (hBmp != NULL) DeleteObject(hBmp);   //  ビットマップ破棄
    PostQuitMessage(0);                     //  プログラム終了
    return 0;
}
//----- WM_PAINT ---------------------------------------------------------------------------------------------//
AJC_WNDPROC(Main, WM_PAINT      )
{
    PAINTSTRUCT ps;
    HDC         hdc, hmd;
    RECT        r;

    hdc   = BeginPaint(hwnd, &ps);
    if (hBmp != NULL) {
        GetClientRect(hwnd, &r);
        hmd = CreateCompatibleDC(hdc);
        SelectObject(hmd, hBmp);
        BitBlt(hdc, (r.right - width) / 2, (r.bottom - height) / 2, width, height, hmd, 0, 0, SRCCOPY);
    }
    EndPaint(hwnd, &ps);
    return 0;
}
//------------------------------------------------------------------------------------------------------------//
AJC_WNDMAP_DEF(Main)
    AJC_WNDMAP_MSG(Main, WM_CREATE  )
    AJC_WNDMAP_MSG(Main, WM_DESTROY )
    AJC_WNDMAP_MSG(Main, WM_PAINT   )
AJC_WNDMAP_END
```

2.26. XMODEM／YMODEM(CAjxXYM)

メンバ変数

#	変数／関数形式	内容
1	HAJCXYM m_hXym;	XMODEM/YMODEM インスタンスハンドル

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxXym(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxXym(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	VO GetTxTimeInfo(UIP t0, UIP r0, UIP t1, UIP r1);	送信時タイマ情報取得
2	VO SetTxTimeInfo(UI t0, UI r0, UI t1, UI r1);	送信時タイマ情報設定
3	VO GetRxTimeInfo(UIP t0, UIP r0, UIP t1, UIP r1);	受信時タイマ情報取得
4	VO SetRxTimeInfo(UI t0, UI r0, UI t1, UI r1);	受信時タイマ情報設定
5	BOOL TxStart (AJCXYMPROTOCOL Protocol = AJCXYP_XMODEM_SUM);	ファイル送信開始
6	BOOL RxStart (AJCXYMPROTOCOL Protocol = AJCXYP_XMODEM_SUM);	ファイル受信開始
7	VO Stop ();	ファイル転送中止
8	VO PutRxChar (UI rxd);	シリアル回線からの受信データ投与
9	VO TxEnd ();	シリアル回線への送信完了を通知（ＹＭＯＤＥＭ－Ｇ専用）

仮想関数

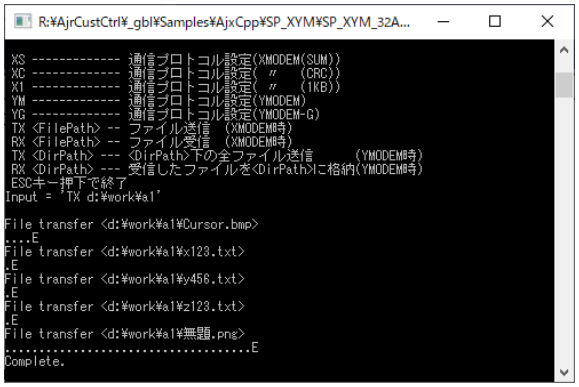
#	変数／関数形式	内容	
1	virtual VO OnNotice (UI knd, UX Param);	イベント通知	cbNotice
2	virtual VO OnGetFile(PAJCXYMFILEINFO pBuf);	ファイル情報取得要求	cbGetFile
3	virtual VO OnGetData(VOP pBuf, UI lBuf, UIP pBytes);	ファイルデータ取得要求	cbGetData
4	virtual VO OnSend (C_VOP pTxD, UI lTxD);	データ送出要求	cbSend

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例

以下のコマンドを投入し、ファイル転送を行います。

```
〈コマンド形式〉      〈内容〉
 XS ------------- 通信プロトコル設定(XMODEM(SUM))
 XC ------------- 通信プロトコル設定( 〃    (CRC))
 X1 ------------- 通信プロトコル設定( 〃    (1KB))
 YM ------------- 通信プロトコル設定(YMODEM)
 YG ------------- 通信プロトコル設定(YMODEM-G)
 TX 〈FilePath〉 -- ファイル送信  (XMODEM 時)
 RX 〈FilePath〉 -- ファイル受信  (XMODEM 時)
 TX 〈DirPath〉 --- 〈DirPath〉下の全ファイル送信       (YMODEM 時)
 RX 〈DirPath〉 --- 受信したファイルを〈DirPath〉に格納(YMODEM 時)
```



例) YMODEM で「d:¥work¥inp」下のファイルを送信し、受信したファイルを「d:¥work¥out」へ格納する場合、各々で以下の２つのコマンドを入力します。

送信側

```
YM
TX  d:¥work¥inp
```

受信側

```
YM
RX  d:¥work¥out
```

```
 1 : //
 2 : //  SP_XYM.cpp
 3 : //
 4 : #include    〈AjxCpp.h〉
 5 : #include    〈tchar.h〉
 6 : #include    〈direct.h〉
 7 : #include    〈conio.h〉
 8 : using namespace AjxControl;
 9 :
10 : //----- CAjxXym, CAjxScp, CAjxCon の派生クラス ----------------------------------------------------------------//
11 : class CAjxXymEx : public CAjxXym, public CAjxScp, public CAjxCon
12 : {
13 :     AJCXYMPROTOCOL          m_Protocol;             //  通信プロトコル
14 :     BOOL                    m_fBusy;                //  ファイル転送中フラグ
15 :     HANDLE                  m_hFile;                //  ファイルハンドル
16 :     SX                      m_hFind;                //  ファイル検索ハンドル
17 :     BOOL                    m_fFindFirst;           //  ファイル検索初回フラグ
18 :     struct _tfinddata64_t   m_FindData;             //  ファイル検索結果
19 :     BOOL                    m_fReceiving;           //  ファイル受信中を示すフラグ
20 :     ULL                     m_RxBytes;              //  ファイル受信サイズ (残受信バイト数)
21 :     ULL                     m_TxBytes;              //  ファイル送信サイズ (残送信バイト数)
22 :     UT                      m_Path[MAX_PATH];       //  送信ファイル／ディレクトリ
23 :     UT                      m_drv[_MAX_DRIVE];      //  ドライブ名
24 :     UT                      m_dir[_MAX_DIR];        //  ディレクトリパス名
25 :
26 :     //----- ファイル転送終了待ち ---------------------------------------------//
27 :     VO  WaitForEnd() {
28 :         m_fBusy = TRUE;
29 :         while (m_fBusy) {
30 :             //  終了チェック (ESC キー入力チェック)
31 :             if (_kbhit()) {
32 :                 int c = _getch();
33 :                 if (c == 0x1B) Stop();
34 :             }
35 :             //  ＳＣＰイベント待ち
36 :             WaitEvent(100);
37 :             //  システムメッセージ処理
38 :             AjcDoEvent();
39 :         }
40 :     }
41 :
42 : public:
43 :     //  コンストラクタ
44 :     CAjxXymEx()
45 :     {
46 :         m_Protocol   = AJCXYP_XMODEM_SUM;
47 :         m_fBusy      = FALSE;
48 :         m_hFile      = NULL;
49 :         m_hFind      = -1;
50 :         m_fFindFirst = TRUE;
51 :         //  ＳＣＰ初期化
52 :         Init(TEXT("MyScpSect"), AJCSCP_CM_BIN);
53 :         SetEvtMask(AJCSCP_EV_PORTSTATE | AJCSCP_EV_RXCHUNK | AJCSCP_EV_TXEMPTY);
54 :     }
55 :     //----- ファイル送信 ----------------------------------------------------//
56 :     VO  SendFile()
57 :     {
58 :         m_fReceiving = FALSE;
59 :         m_fFindFirst = TRUE;
```

```
60 :         if (TxStart(m_Protocol)) {
61 :             WaitForEnd();
62 :         }
63 :     }
64 :     //----- ファイル受信 ----------------------------------------------------//
65 :     VO  RecvFile()
66 :     {
67 :         m_fReceiving = TRUE;
68 :         if (RxStart(m_Protocol)) {
69 :             WaitForEnd();
70 :         }
71 :     }
72 :     //----- 受信ファイル書き込み --------------------------------------------//
73 :     VO  SubWrite(VOP pDat, ULL len) {
74 :         UL  bytes;
75 :         if (m_hFile != INVALID_HANDLE_VALUE) {
76 :             if (m_RxBytes > len) {WriteFile(m_hFile, pDat, (UI)len      , &bytes, NULL); m_RxBytes -= len;}
77 :             else                 {WriteFile(m_hFile, pDat, (UI)m_RxBytes, &bytes, NULL); m_RxBytes  = 0;  }
78 :         }
79 :     }
80 :     //----- XYM Override ----------------------------------------------------//
81 :     //  イベント通知
82 :     VO  OnNotice (UI knd, UX Param              )   override {
83 :         PCAJCXYMFILEINFO pfi;
84 :         UT      path[MAX_PATH];
85 :         //  イベントアクション
86 :         switch (knd) {
87 :             //  受信ファイル情報通知（ファイル受信時）
88 :             case AJCXYN_RXFILEINFO:     pfi = (PCAJCXYMFILEINFO)Param;
89 :                                         MAjcMakePath(path, m_drv, m_dir, pfi->pFName, NULL);
90 :                                         SAjxCon::PrintF(TEXT("File receive '%s'¥n"), path);
91 :                                         m_RxBytes = pfi->size;
92 :                                         if ((m_hFile = CreateFile(path, GENERIC_WRITE, 0, NULL,
93 :                                             CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL)) == INVALID_HANDLE_VALUE) {
94 :                                             SAjxCon::PrintF(TEXT("File creation failure <%s>¥n"), path);
95 :                                         }
96 :                                         break;
97 :             //  １レコード送信完了
98 :             case AJCXYN_TXREC:          SAjxCon::PrintF(TEXT("."));
99 :                                         break;
100 :            //  １レコード受信（１２８バイトレコード）
101 :            case AJCXYN_RXREC_128:      SAjxCon::PrintF(TEXT(".")); SubWrite((VO*)Param, 128);
102 :                                        break;
103 :            //  １レコード受信（１ＫＢレコード）
104 :            case AJCXYN_RXREC_1K:       SAjxCon::PrintF(TEXT(".")); SubWrite((VO*)Param, 1024);
105 :                                        break;
106 :            //  再送発生
107 :            case AJCXYN_RETRY:          SAjxCon::PrintF(TEXT("R"));
108 :                                        break;
109 :            //  ＥＯＦ通知
110 :            case AJCXYN_EOF:            SAjxCon::PrintF(TEXT("E¥n"));
111 :                                        if (m_hFile != INVALID_HANDLE_VALUE) {
112 :                                            CloseHandle(m_hFile);
113 :                                            m_hFile = INVALID_HANDLE_VALUE;
114 :                                        }
115 :                                        break;
116 :            //  通信プロトコル変更通知
117 :            case AJCXYN_PROTOCOL:       SAjxCon::PrintF(TEXT("Protocol change.¥n"));
118 :                                        break;
119 :            //  ファイル転送完了
120 :            case AJCXYN_COMPLETE:       SAjxCon::PrintF(TEXT("Complete.¥n"));
121 :                                        break;
122 :            //  ファイル転送中止（ＣＡＮ受信            ）
123 :            case AJCXYN_RX_CAN:         SAjxCon::PrintF(TEXT("Stop by Received CAN¥n"));
124 :                                        break;
125 :            //  ファイル転送中止（ユーザからの中止要求  ）
126 :            case AJCXYN_USERSTOP:       SAjxCon::PrintF(TEXT("Stop by user¥n"));
127 :                                        break;
128 :            //  ファイル転送中止（タイムアウト          ）
129 :            case AJCXYN_TIMEOUT:        SAjxCon::PrintF(TEXT("Stop by Timeout¥n"));
130 :                                        break;
131 :        }
132 :        //  転送終了
133 :        if ((knd & AJCXYN_END) != 0) {
134 :            if (m_hFind != -1                ) {_findclose (m_hFind); m_hFind = -1;                 }
135 :            if (m_hFile != INVALID_HANDLE_VALUE) {CloseHandle(m_hFile); m_hFile = INVALID_HANDLE_VALUE;}
136 :            m_fBusy = FALSE;
137 :        }
138 :    }
139 :    //  ファイル情報取得要求（ファイル送信時）
140 :    VO  OnGetFile(PAJCXYMFILEINFO pBuf          )   override {
141 :        BOOL    fEndOfFind  = FALSE;
142 :        //----- XMODEM の場合、指定ファイルオープン ------------------------------//
143 :        if (m_Protocol & AJCXYP_XMODEM) {
144 :            m_hFind  = -1;
145 :            //  送信ファイルオープン
146 :            if ((m_hFile = CreateFile(m_Path, GENERIC_READ, FILE_SHARE_READ, NULL,
147 :                                      OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)) != INVALID_HANDLE_VALUE) {
148 :                //  送信ファイルサイズ設定
149 :                m_TxBytes = AjcGetFileSize(m_Path);
150 :                //  ファイル情報設定
```

```
151 :                 pBuf->pFName = m_Path;
152 :                 pBuf->size   = AjcGetFileSize    (m_Path);
153 :                 pBuf->time   = AjcGetFileTime1970(m_Path);
154 :                 SAjxCon::PrintF(TEXT("File transfer <%s>¥n"), m_Path);
155 :             }
156 :             else {
157 :                 SAjxCon::PrintF(TEXT("File open failure <%s>¥n"), m_Path);
158 :             }
159 :         }
160 :         //----- YMODEM の場合、ディレクトリ下のファイル検索 ---------------------//
161 :         else {
162 :             //  前回のファイルクローズ
163 :             if (m_hFile != INVALID_HANDLE_VALUE) {
164 :                 CloseHandle(m_hFile);
165 :                 m_hFile = INVALID_HANDLE_VALUE;
166 :             }
167 :             //  次の送信ファイル設定
168 :             while (!fEndOfFind  &&  m_hFile == INVALID_HANDLE_VALUE) {
169 :                 //  次のファイル検索
170 :                 do {
171 :                     UT wild[MAX_PATH];
172 :                     MAjcMakePath(wild, m_drv, m_dir, TEXT("*"), TEXT(".*"));
173 :                     if (m_fFindFirst) {if ((m_hFind = _tfindfirst64(wild, &m_FindData)) == -1) fEndOfFind = TRUE;}
174 :                     else              {if (_tfindnext64(m_hFind, &m_FindData)                == -1) fEndOfFind = TRUE;}
175 :                     m_fFindFirst = FALSE;
176 :                 } while (!fEndOfFind && (m_FindData.attrib & (_A_HIDDEN | _A_SUBDIR | _A_SYSTEM)));
177 :                 //  ファイルデータ／ＥＯＦ設定
178 :                 if (!fEndOfFind) {
179 :                     //  送信ファイルサイズ設定
180 :                     m_TxBytes = m_FindData.size;
181 :                     //  送信ファイルオープン
182 :                     UT      path[MAX_PATH];
183 :                     MAjcMakePath(path, m_drv, m_dir, m_FindData.name, NULL);
184 :                     if ((m_hFile = CreateFile(path, GENERIC_READ, FILE_SHARE_READ, NULL,
185 :                                               OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL)) != INVALID_HANDLE_VALUE) {
186 :                         //  ファイル情報設定
187 :                         pBuf->pFName = m_FindData.name;
188 :                         pBuf->size   = m_FindData.size;
189 :                         pBuf->time   = (UL)(m_FindData.time_write);
190 :                         SAjxCon::PrintF(TEXT("File transfer <%s>¥n"), path);
191 :                     }
192 :                     else {
193 :                         SAjxCon::PrintF(TEXT("File open failure <%s>¥n"), path);
194 :                     }
195 :                 }
196 :                 else {
197 :                     //  ファイル検索終了
198 :                     _findclose(m_hFind);
199 :                     m_hFind = -1;
200 :                     //  EOF 設定
201 :                     pBuf->pFName = NULL;
202 :                 }
203 :             }
204 :         }
205 :     }
206 :     //  ファイルデータ取得要求 (ファイル送信時)
207 :     VO  OnGetData(VOP pBuf, UI lBuf, UIP pBytes )   override {
208 :         ReadFile(m_hFile, pBuf, lBuf, (ULP)pBytes, NULL);
209 :         if (m_TxBytes >= *pBytes) m_TxBytes -= *pBytes;
210 :         else                      m_TxBytes  = 0;
211 :     }
212 :     //  データ送出要求 (ファイル送信時)
213 :     VO  OnSend    (C_VOP pTxD, UI lTxD          )   override {
214 :         SendBinData(pTxD, lTxD);
215 :     }
216 :     //----- SCP Override ----------------------------------------------------//
217 :     //  ポート状態通知
218 :     VO  OnNtcPortState  (C_BCP pPortName, UI Param) override {
219 :     }
220 :     //  バイナリチャンク受信通知
221 :     VO  OnNtcRxBinChunk (C_VOP pData, UI Bytes)     override {
222 :         UBP p = (UBP)pData;
223 :         while (Bytes--) {
224 :             PutRxChar(*p++);
225 :         }
226 :     }
227 :     //  送信完了通知
228 :     VO  OnNtcTxEmpty    ()                          override {
229 :         //  YMODEM-G の場合、送信完了を通知
230 :         if (m_Protocol == AJCXYP_YMODEM_G && !m_fReceiving && m_TxBytes == 0) {
231 :             TxEnd();
232 :         }
233 :     }
234 :     //----- コマンドパラメタ通知 --------------------------------------------//
235 :     VO  OnNtcStr(int argc, UT *argv[], C_UTP pTxt) override {
236 :         SAjxCon::PrintF(TEXT("Input = '%s'¥n¥n"), pTxt);
237 :         if      (argc == 1) {
238 :             if      (_tcsicmp(argv[0], TEXT("XS")) == 0) {m_Protocol = AJCXYP_XMODEM_SUM;}
239 :             else if (_tcsicmp(argv[0], TEXT("XC")) == 0) {m_Protocol = AJCXYP_XMODEM_CRC;}
240 :             else if (_tcsicmp(argv[0], TEXT("X1")) == 0) {m_Protocol = AJCXYP_XMODEM_1K; }
241 :             else if (_tcsicmp(argv[0], TEXT("YM")) == 0) {m_Protocol = AJCXYP_YMODEM_STD;}
```

```
242 :             else if (_tcsicmp(argv[0], TEXT("YG")) == 0) {m_Protocol = AJCXYP_YMODEM_G;  }
243 :             else  SAjxCon::PrintF(TEXT(" *** Invalid directory '%s'.¥n"), argv[0]);
244 :         }
245 :         else if (argc == 2) {
246 :             if (((m_Protocol &AJCXYP_XMODEM) && AjcPathIsFile     (argv[1])) ||
247 :                 ((m_Protocol &AJCXYP_YMODEM) && AjcPathIsDirectory(argv[1]))) {
248 :                 //  パス名設定
249 :                 _tcscpy_s(m_Path, AJCTSIZE(m_Path), argv[1]);
250 :                 if (m_Protocol &AJCXYP_YMODEM) {
251 :                     AjcPathCat(m_Path, TEXT(""), AJCTSIZE(m_Path));
252 :                     MAjcSplitPath(m_Path, m_drv, m_dir, NULL, NULL);
253 :                 }
254 :                 //  コマンド実行
255 :                 if      (_tcsicmp(argv[0], TEXT("TX")) == 0) {SendFile();}
256 :                 else if (_tcsicmp(argv[0], TEXT("RX")) == 0) {RecvFile();}
257 :                 else SAjxCon::PrintF(TEXT(" *** Invalid parameter '%s'.¥n"), argv[0]);
258 :             }
259 :             else SAjxCon::PrintF(TEXT(" *** Invalid directory '%s'.¥n"), argv[1]);
260 :         }
261 :         else SAjxCon::PrintF(TEXT(" *** missing number of parameters.¥n"));
262 :     }
263 : };
264 :
265 : CAjxXymEx   xym_scp;
266 :
267 : //----- m a i n ----------------------------------------------------------------------------------------------------//
268 : int AjcMain(int argc, UTP argv[])
269 : {
270 :     SAjxCon::SetStdMode();
271 :
272 :     //  ポートパラメタ設定
273 :     UT WndTxt[256];
274 :     GetConsoleTitle(WndTxt, 256);
275 :     xym_scp.DlgParamEasy(FindWindow(NULL, WndTxt));
276 :     //  ポートオープン
277 :     xym_scp.Open();
278 :
279 :     if (xym_scp.IsOpened()) {
280 :         //コマンド入力ループ
281 :         do {
282 :             SAjxCon::PrintF(TEXT("¥n"));
283 :             SAjxCon::PrintF(TEXT(" XS ------------ 通信プロトコル設定(XMODEM(SUM))¥n"));
284 :             SAjxCon::PrintF(TEXT(" XC ------------ 通信プロトコル設定( 〃    (CRC))¥n"));
285 :             SAjxCon::PrintF(TEXT(" X1 ------------ 通信プロトコル設定( 〃    (1KB))¥n"));
286 :             SAjxCon::PrintF(TEXT(" YM ------------ 通信プロトコル設定(YMODEM)¥n"     ));
287 :             SAjxCon::PrintF(TEXT(" YG ------------ 通信プロトコル設定(YMODEM-G)¥n"   ));
288 :             SAjxCon::PrintF(TEXT(" TX <FilePath> -- ファイル送信  (XMODEM 時)¥n"));
289 :             SAjxCon::PrintF(TEXT(" RX <FilePath> -- ファイル受信  (XMODEM 時)¥n"));
290 :             SAjxCon::PrintF(TEXT(" TX <DirPath> --- <DirPath>下の全ファイル送信       (YMODEM 時)¥n"));
291 :             SAjxCon::PrintF(TEXT(" RX <DirPath> --- 受信したファイルを<DirPath>に格納(YMODEM 時)¥n"));
292 :             SAjxCon::PrintF(TEXT(" ESC キー押下で終了¥n"));
293 :         } while (xym_scp.Input(TEXT("")));
294 :     }
295 :     else {
296 :         SAjxCon::PrintF(TEXT(" ポートをオープンできません。¥n"));
297 :         SAjxCon::PrintF(TEXT(" Hit Enter Key!!¥n"));
298 :         getchar();
299 :     }
300 :
301 :     return 0;
302 : }
```

2.27. 印 刷 (CAjxPrnt)

メンバ変数

#	変数／関数形式	内容
1	HAJCPRN m_hPrn;	印刷インスタンスハンドル

コンストラクタ

#	変数／関数形式	内容

1	#ifdef UNICODE
	CAjxPrn(BOOL fUnicode = TRUE); // コンストラクタ(UNICODE)
	#else
	CAjxPrn(BOOL fUnicode = FALSE); // コンストラクタ(ASCII)
	#endif
		コンストラクタ

メンバ関数

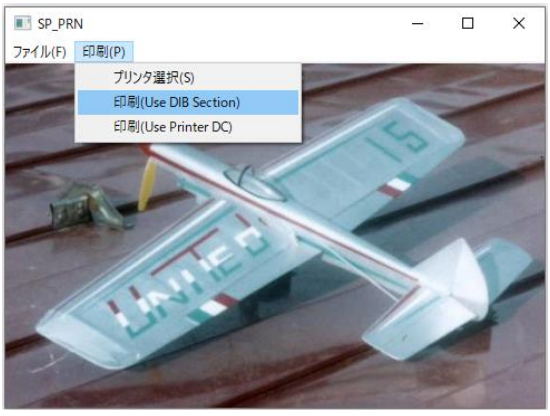
#	変数／関数形式	内容
1	BOOL SetMargin (PAJCPRN_MARGIN pmmMargin);	余白サイズ設定
2	BOOL SelectDialog(PAJCPRN_INFO pInfo, UI Flag, HWND hWndOwner);	プリンタの選択ダイアログ表示
3	BOOL PrintDialog (PAJCPRN_INFO pInfo, PAJCPRN_OPT pOpt, HWND hWndOwner);	印刷ダイアログ表示
4	int Start (BOOL fUseDib = FALSE, BOOL fOutputToFile = FALSE);	印刷開始
5	VO GetInfo (PAJCPRN_INFO pBuf, UTP pPrnName, UI lPrnName);	プリンタ設定情報取得

仮想関数

#	変数／関数形式	内容	
1	virtual BOOL OnQueryPage(PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPageInfo);	ページ印刷開始通知	cbQueryPage
2	virtual BOOL OnDrawByDibSectDC(HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo);	ＤＩＢセクション・ビットマップによる１ページ描画	cbDrawByDibSectDC
3	virtual BOOL OnDrawByPrinterDC (HDC hdc, PCAJCPRN_INFO pPrnInfo);	プリンタＤＣによる１ページ描画	cbDrawByDibPrinterDC

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例



イメージを表示し、印刷します。
「ファイル→イメージファイル選択」メニューで画像ファイルを選択します。
（画像ファイルをウインドにドロップしてもＯＫ）
「印刷」メニューで、プリンタの選択や印刷の実行を行います。
画像は、用紙の中央に原寸（画像の拡大縮小なしにそのまま）で印刷します。

```
  1 : //
  2 : //  SP_PRN.cpp
  3 : //
  4 : #include    <AjxCpp.h>
  5 : #include    <tchar.h>
  6 : #include    "..¥src¥resource.h"
  7 : using namespace AjxControl;
  8 :
  9 : static  HINSTANCE       hInst    = NULL;
 10 : static  HWND            hWndMain = NULL;
 11 : static  HMENU           hMenu    = NULL;
 12 : static  HBITMAP         hPrn     = NULL;
 13 : static  AJC_IMGINFO     ImgInf   = {0};
 14 : AJC_WNDPROC_DEF(Main);
 15 :
 16 : //----- AjxPrn の派生クラス ----------------------------------------------------------------------------//
 17 : class CAjxPrnEx : public CAjxPrn
 18 : {
 19 :     public:
 20 :         //  ページ印刷開始通知
 21 :         BOOL    OnQueryPage             (PCAJCPRN_INFO pPrnInfo, PAJCPRN_PGINFO pPageInfo)          override {
 22 :             return TRUE;
 23 :         }
 24 :         //  ＤＩＢセクション・ビットマップによる描画
 25 :         BOOL    OnDrawByDibSectDC       (HDC hdc, PCAJCPRN_INFO pPrnInfo, PCAJCDIBINFO pDibInfo)    override {
```

```
26 :             RECT    ir, pr;
27 :             int     x, y;
28 :             SetRect(&ir, 0, 0, ImgInf.width, ImgInf.height);
29 :             x = (pDibInfo->width  - ImgInf.width ) / 2;
30 :             y = (pDibInfo->height - ImgInf.height) / 2;
31 :             SetRect(&pr, x, y, x + ImgInf.width, y + ImgInf.height);
32 :             AjcImgFuncDraw(&ImgInf, &ir, hdc, &pr);
33 :             return FALSE;
34 :         }
35 :         //  プリンタＤＣによる描画
36 :         BOOL    OnDrawByPrinterDC       (HDC hdc, PCAJCPRN_INFO pPrnInfo)                           override {
37 :             RECT    ir, pr;
38 :             int     x, y;
39 :             SetRect(&ir, 0, 0, ImgInf.width, ImgInf.height);
40 :             x = pPrnInfo->x + (pPrnInfo->cx - ImgInf.width ) / 2;
41 :             y = pPrnInfo->y + (pPrnInfo->cy - ImgInf.height) / 2;
42 :             SetRect(&pr, x, y, x + ImgInf.width, y + ImgInf.height);
43 :             AjcImgFuncDraw(&ImgInf, &ir, hdc, &pr);
44 :             return FALSE;
45 :         }
46 : };
47 :
48 : CAjxPrnEx prn;
49 :
50 : //----- WinMain ---------------------------------------------------------------------------------------------//
51 : int WINAPI AjcWinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, UTP szCmdLine, int iCmdShow)
52 : {
53 :     MSG         msg = {0};
54 :     WNDCLASS    wc  = {0};
55 :
56 :     hInst = hInstance;
57 :     //  ウインド生成
58 :     wc.style         = 0;                           wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
59 :     wc.lpfnWndProc   = AJC_WNDPROC_NAME(Main);      wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
60 :     wc.hInstance     = hInstance;                   wc.lpszClassName = TEXT("SP_PRN");
61 :     RegisterClass(&wc);
62 :     hMenu = LoadMenu(hInst, MAKEINTRESOURCE(IDR_MENU1));
63 :     hWndMain = CreateWindowEx(WS_EX_ACCEPTFILES,            // extend style
64 :                 TEXT("SP_PRN"), TEXT("SP_PRN"),             // window class name, caption
65 :                 WS_OVERLAPPEDWINDOW,                        // window style
66 :                 CW_USEDEFAULT, CW_USEDEFAULT, 500, 300,     // position, size
67 :                 NULL, hMenu, hInstance, NULL);              // parent, menu, instance, param
68 :     ShowWindow(hWndMain, iCmdShow);
69 :     //  メッセージループ
70 :     while (GetMessage(&msg, NULL, 0, 0)) {
71 :         TranslateMessage(&msg);
72 :         DispatchMessage (&msg);
73 :     }
74 :     DeleteObject(hMenu);
75 :     return (int)msg.wParam ;
76 : }
77 : //----- WM_CREATE -------------------------------------------------------------------------------------------//
78 : AJC_WNDPROC(Main, WM_CREATE     )
79 : {
80 :     return 0;
81 : }
82 : //----- WM_DESTROY ------------------------------------------------------------------------------------------//
83 : AJC_WNDPROC(Main, WM_DESTROY    )
84 : {
85 :     AjcImgFuncRelease(&ImgInf);         //  イメージデータ破棄
86 :     PostQuitMessage(0);                 //  プログラム終了
87 :     return 0;
88 : }
89 : //----- WM_DROPFILES ----------------------------------------------------------------------------------------//
90 : AJC_WNDPROC(Main, WM_DROPFILES  )
91 : {
92 :     HDROP       hDrop;
93 :     UT          path[MAX_PATH];
94 :
95 :     hDrop = (HDROP)wParam;                              //  ドロップハンドル設定
96 :     if (DragQueryFile(hDrop, 0, path, MAX_PATH)) {
97 :         if (AjcPathIsFile(path)) {
98 :             AjcImgFuncRelease(&ImgInf);                 //  前イメージデータ破棄
99 :             AjcImgFuncRead(&ImgInf, path);              //  イメージデータ読み出し
100 :            InvalidateRect(hwnd, NULL, TRUE);           //  再描画
101 :        }
102 :    }
103 :    DragFinish(hDrop);                                  //  ドロップ終了
104 :    return 0;
105 : }
106 : //----- WM_SIZE ---------------------------------------------------------------------------------------------//
107 : AJC_WNDPROC(Main, WM_SIZE       )
108 : {
109 :     InvalidateRect(hwnd, NULL, TRUE);
110 :     return 0;
111 : }
112 : //----- WM_PAINT --------------------------------------------------------------------------------------------//
113 : AJC_WNDPROC(Main, WM_PAINT      )
114 : {
115 :     PAINTSTRUCT ps;
116 :     HDC         hdc;
```

```
117 :     RECT        wr, ir;
118 :     int         ww, wh;
119 :
120 :     hdc   = BeginPaint(hwnd, &ps);
121 :     if (ImgInf.fValid) {
122 :         GetClientRect(hwnd, &wr);
123 :         ww = wr.right - wr.left;  wh = wr.bottom - wr.top;
124 :         if (ImgInf.height >= ImgInf.width) {
125 :             SetRect(&wr, 0, 0, ImgInf.width * wh / ImgInf.height, wh                                );
126 :         }
127 :         else {
128 :             SetRect(&wr, 0, 0, ww                                 , ImgInf.height * ww / ImgInf.width );
129 :         }
130 :         SetRect(&ir, 0, 0, ImgInf.width, ImgInf.height);
131 :         AjcImgFuncDraw(&ImgInf, &ir, hdc, &wr);
132 :     }
133 :     EndPaint(hwnd, &ps);
134 :     return 0;
135 : }
136 : //----- WM_INITMENU ------------------------------------------------------------------------------------------//
137 : AJC_WNDPROC(Main, WM_INITMENU   )
138 : {
139 :     EnableMenuItem(hMenu, ID_PRINT_DIB, MF_BYCOMMAND | (ImgInf.fValid ? MF_ENABLED : MF_GRAYED));
140 :     EnableMenuItem(hMenu, ID_PRINT_PRN, MF_BYCOMMAND | (ImgInf.fValid ? MF_ENABLED : MF_GRAYED));
141 :     return 0;
142 : }
143 : //----- ファイル→イメージファイル選択 ------------------------------------------------------------------------//
144 : AJC_WNDPROC(Main, ID_FILE       )
145 : {
146 :     static UT   path[MAX_PATH] = {0};
147 :     if (AjcGetOpenFile(hwnd, TEXT("イメージファイルの選択"),
148 :             TEXT("All Files (*.*)/*.*/Bitmap File (*.bmp)/*.bmp/.jpg File/*.jpg/.png File/*.png/.gif File/*.gif"), TEXT("bmp"),
149 :             path, MAX_PATH)) {
150 :         AjcImgFuncRelease(&ImgInf);             //  前イメージデータ破棄
151 :         AjcImgFuncRead(&ImgInf, path);          //  イメージデータ読み出し
152 :         InvalidateRect(hwnd, NULL, TRUE);       //  再描画
153 :     }
154 :     return 0;
155 : }
156 : //----- ファイル→終了 ----------------------------------------------------------------------------------------//
157 : AJC_WNDPROC(Main, ID_EXIT       )
158 : {
159 :     DestroyWindow(hwnd);
160 :     return 0;
161 : }
162 : //----- 印刷→プリンタ選択 ------------------------------------------------------------------------------------//
163 : AJC_WNDPROC(Main, ID_SELPRN     )
164 : {
165 :     prn.SelectDialog(NULL, AJCPRNF_NONETWORKBUTTON, hwnd);
166 :     return 0;
167 : }
168 : //----- 印刷→印刷(Use DIB Section) ---------------------------------------------------------------------------//
169 : AJC_WNDPROC(Main, ID_PRINT_DIB  )
170 : {
171 :     AJCPRN_OPT opt = {AJCPRNF_DISABLECOPIES | AJCPRNF_DISABLEPAGEINFO, 1, 1, 1, 1, 1};
172 :     if (prn.PrintDialog(NULL, &opt, hwnd)) {
173 :         prn.Start(TRUE);
174 :     }
175 :     return 0;
176 : }
177 : //----- 印刷→印刷(Use Printer DC) ----------------------------------------------------------------------------//
178 : AJC_WNDPROC(Main, ID_PRINT_PRN  )
179 : {
180 :     AJCPRN_OPT opt = {AJCPRNF_DISABLECOPIES | AJCPRNF_DISABLEPAGEINFO, 1, 1, 1, 1, 1};
181 :     if (prn.PrintDialog(NULL, &opt, hwnd)) {
182 :         prn.Start(FALSE);
183 :     }
184 :     return 0;
185 : }
186 : //------------------------------------------------------------------------------------------------------------//
187 : AJC_WNDMAP_DEF(Main)
188 :     AJC_WNDMAP_MSG(Main, WM_CREATE      )
189 :     AJC_WNDMAP_MSG(Main, WM_DESTROY     )
190 :     AJC_WNDMAP_MSG(Main, WM_DROPFILES   )
191 :     AJC_WNDMAP_MSG(Main, WM_SIZE        )
192 :     AJC_WNDMAP_MSG(Main, WM_PAINT       )
193 :     AJC_WNDMAP_MSG(Main, WM_INITMENU    )
194 :     AJC_WNDMAP_CMD(Main, ID_FILE        )
195 :     AJC_WNDMAP_CMD(Main, ID_EXIT        )
196 :     AJC_WNDMAP_CMD(Main, ID_SELPRN      )
197 :     AJC_WNDMAP_CMD(Main, ID_PRINT_DIB   )
198 :     AJC_WNDMAP_CMD(Main, ID_PRINT_PRN   )
199 : AJC_WNDMAP_END
```

2.28. ヒープソート (CAjxSort)

メンバ変数

#	変数／関数形式	内容
1	HAJCPRN m_hPrn;	印刷インスタンスハンドル

コンストラクタ

#	変数／関数形式	内容
1	CAjxHso() ;	コンストラクタ

メンバ関数

#	変数／関数形式	内容
1	BOOL Sort(VOP pTbl, UI nTbl, UI size);	テーブル (配列) のソート

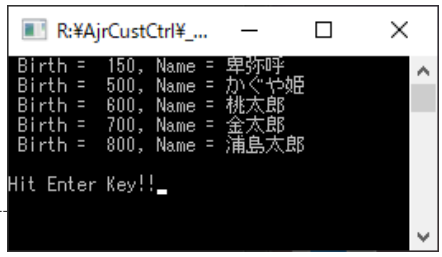
純粋仮想関数

#	変数／関数形式	内容	
1	virtual int OnComp(C_VOP pElm1, C_VOP pElm2, BOOL *pStop) = 0;	テーブル(配列)要素の比較	cbComp

※ cb*XXXXX*は、対応するコールバック関数を示します。

使用例

テーブル(配列)を birth 順にソートし、表示します。




```
 1 : //
 2 : //  SP_HSO.c
 3 : //
 4 : #include    <AjxCpp.h>
 5 : #include    <tchar.h>
 6 : using namespace AjxControl;
 7 :
 8 : //----- テーブル (配列)  ------------------------------------------------
 9 : typedef struct {
10 :     UTP     pName;      //  名前
11 :     UI      birth;      //  誕生年
12 : } TBL, *PTBL;
13 : typedef const TBL *PCTBL;
14 :
15 : static  TBL     tbl[] = {
16 :     {TEXT("金太郎"  ), 700},
17 :     {TEXT("桃太郎"  ), 600},
18 :     {TEXT("浦島太郎"), 800},
19 :     {TEXT("かぐや姫"), 500},
20 :     {TEXT("卑弥呼"  ), 150},
21 : };
22 :
23 : #define MAX_TBL ((sizeof tbl) / (sizeof tbl[0]))
24 :
25 : //----- CAjxHso の継承クラス ---------------------------------------------------------------------------//
26 : class CAjxHsoEx : public CAjxHso
27 : {
28 : public:
29 :     int OnComp(C_VOP pElm1, C_VOP pElm2, BOOL *pStop) override {
30 :         PCTBL   p1 = (PCTBL)pElm1;
31 :         PCTBL   p2 = (PCTBL)pElm2;
32 :         return (p1->birth - p2->birth);
33 :     }
34 : };
35 :
36 : CAjxHsoEx   hso;
37 :
38 : //=============================================================================================================//
39 : int  AjcMain(int argc, UTP argv[])
40 : {
41 :     SAjxCon::SetStdMode();
42 :
43 :     //  テーブルソート
44 :     hso.Sort(tbl, MAX_TBL, sizeof tbl[0]);
45 :     //  ソート済テーブル表示
46 :     for (int i = 0; i < MAX_TBL; i++) {
47 :         SAjxCon::PrintF(TEXT(" Birth = %4d, Name = %s¥n"), tbl[i].birth, tbl[i].pName);
48 :     }
49 :
50 :     SAjxCon::PrintF(TEXT("¥nHit Enter Key!!"));
51 :     getchar();
52 :
53 :     return 0;
54 : }
55 :
```

2.29. ３Ｄテストデータ生成（CAjxSpd）

コンストラクタ

#	変数／関数形式	内容
1	CAjxSpd(UI seed = 1):	コンストラクタ

メンバ関数

#	変数／関数形式			内容
1	BOOL	SetParam	(PCAJCSPD_PARAM pParam);	パラメタ設定
2	BOOL	GetParam	(PAJCSPD_PARAM pBuf);	パラメタ取得
3	BOOL	Calc	(double *x, double *y, double *z);	演算値取得
4	BOOL	Calc	(PAJC3DVEC pBuf);	演算値取得（ベクタ指定）