キャッツ株式会社
ラウンドロビンエンジニアリング

１．質問

VC で外観図を作成するには、どうすればよいのでしょうか？

２．回答

VC にて外観図を作成する場合は、VIP 通信用ライブラリ(ZVipPnl.DLL)を使用します。
VIP 通信用ライブラリを使用した外観図の作成手順例を次章より説明します。

キャッツ株式会社
ラウンドロビンエンジニアリング

３．外観図作成

３.１　プロジェクト作成

本章では、VIP 通信用ライブラリを使用する VC プロジェクトの作成手順を説明します。

【プロジェクト作成手順】

1．VC にて、プロジェクトを作成します。

※　ZIPC は外観図とメッセージにて通信します。　その為、外観図のプロジェクトは SDI プロジェクト　および　MDI プロジェクトのどちらかにして下さい。

2．VC にて、下記に示すファイルをプロジェクトに追加します。

- ZVipPnlImp.h
- VipUpdateData.h
- VipUpdateData.cpp
- VipCommunicate.h
- VipCommunicate.cpp

※　これらのファイルは、簡単に VIP 通信用ライブラリを使用できるように作成したモジュールです。　VC にて作成したプロジェクトフォルダにコピーして使用することをお奨めします。
尚、これらのファイルの詳細は、【第６章　追加ファイル説明】にて説明しています。

3．ZIPC との通信を定義するソースファイル（cpp）にて、” VipCommunicate.h ” ファイルをインクルードするように追記します。

■サンプル例

```
#include "zipcvip/VipCommunicate.h"
```

これで、VIP 通信用ライブラリを使用する環境を構築しました。

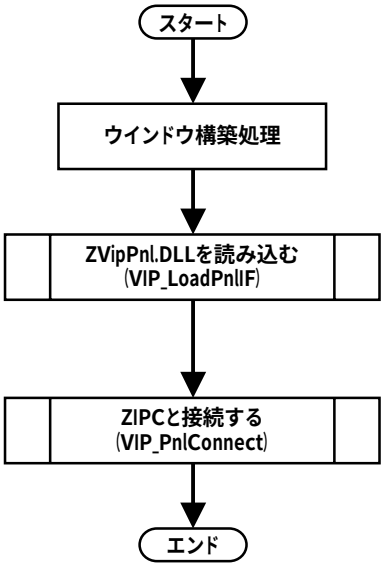
キャッツ株式会社
ラウンドロビンエンジニアリング

３.２　起動処理

本章では、外観図が起動時に ZIPC と接続を行うように、実装する方法を説明します。

外観図が起動時に ZIPC と接続するための、起動処理フローを下図に示します。

■起動処理フロー

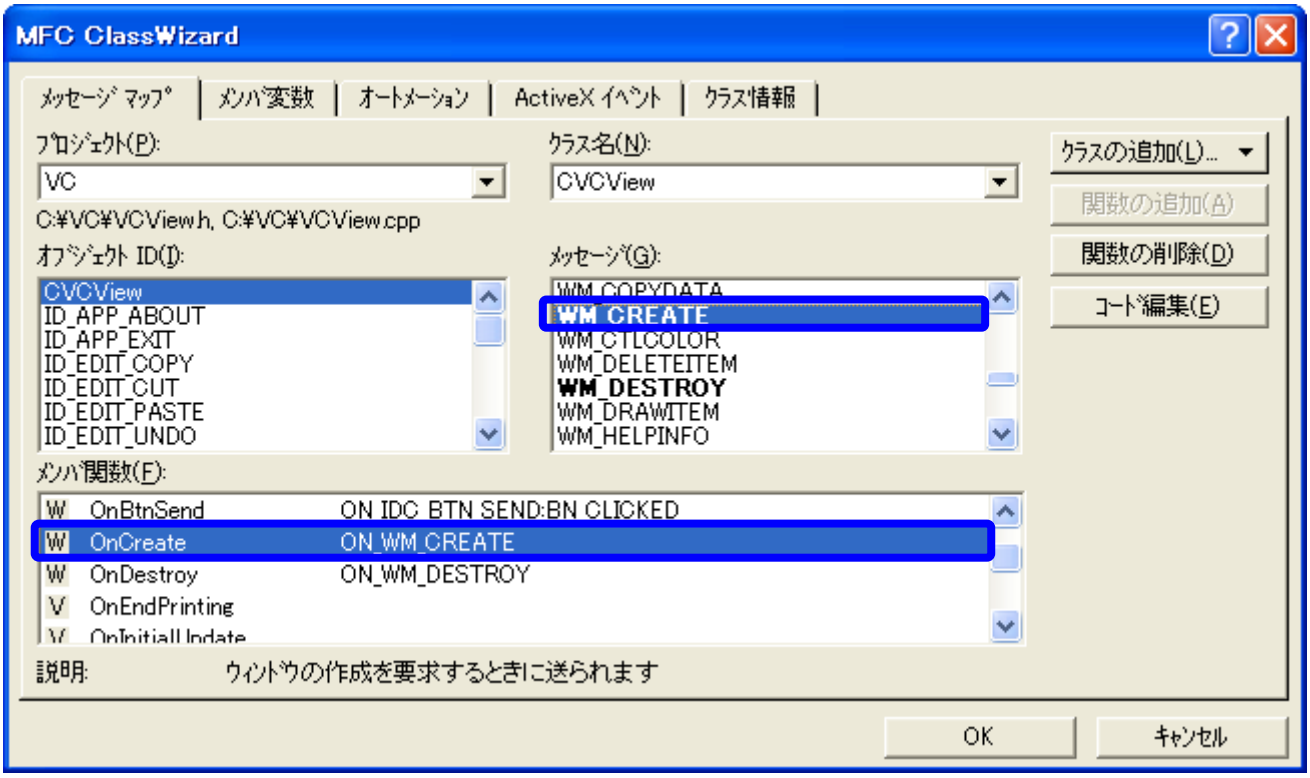


キャッツ株式会社
ラウンドロビンエンジニアリング

外観図が起動時に ZIPC と接続するように、実装する手順を説明します。

【プロジェクト作成手順】

1. ”WM_CREATE”メッセージのハンドラ（”OnCreate”関数）を定義します。



キャッツ株式会社
ラウンドロビンエンジニアリング

2. ”OnCreate”関数にて、ZIPC と接続する関数をコールします。

■サンプル例

```
int CVCView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFormView::OnCreate(lpCreateStruct) == -1)
        return -1;

    // TODO: この位置に固有の作成用コードを追加してください

    // ZIPC と接続する
    char exename[_MAX_PATH];
    char drive[_MAX_DRIVE];
    char dir[_MAX_DIR];
    GetModuleFileName( AfxGetInstanceHandle(), exename, _MAX_PATH-1 );
    _splitpath( exename, drive, dir, NULL, NULL );
    VIP_LoadPnlIF( CString( drive ) + CString( dir ) + CString("ZVipPnl.DLL") );
    VIP_PnlConnect( _T(""), GetSafeHwnd() );

    return 0;
}
```

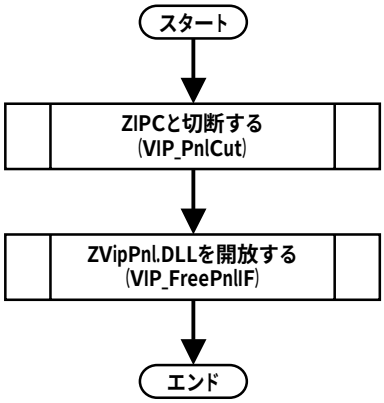
キャッツ株式会社
ラウンドロビンエンジニアリング

3.3 終了処理

本章では、外観図が終了時に ZIPC との切断を行うように、実装する方法を説明します。

外観図が終了時に ZIPC との切断するための、終了処理フローを下図に示します。

■終了処理フロー

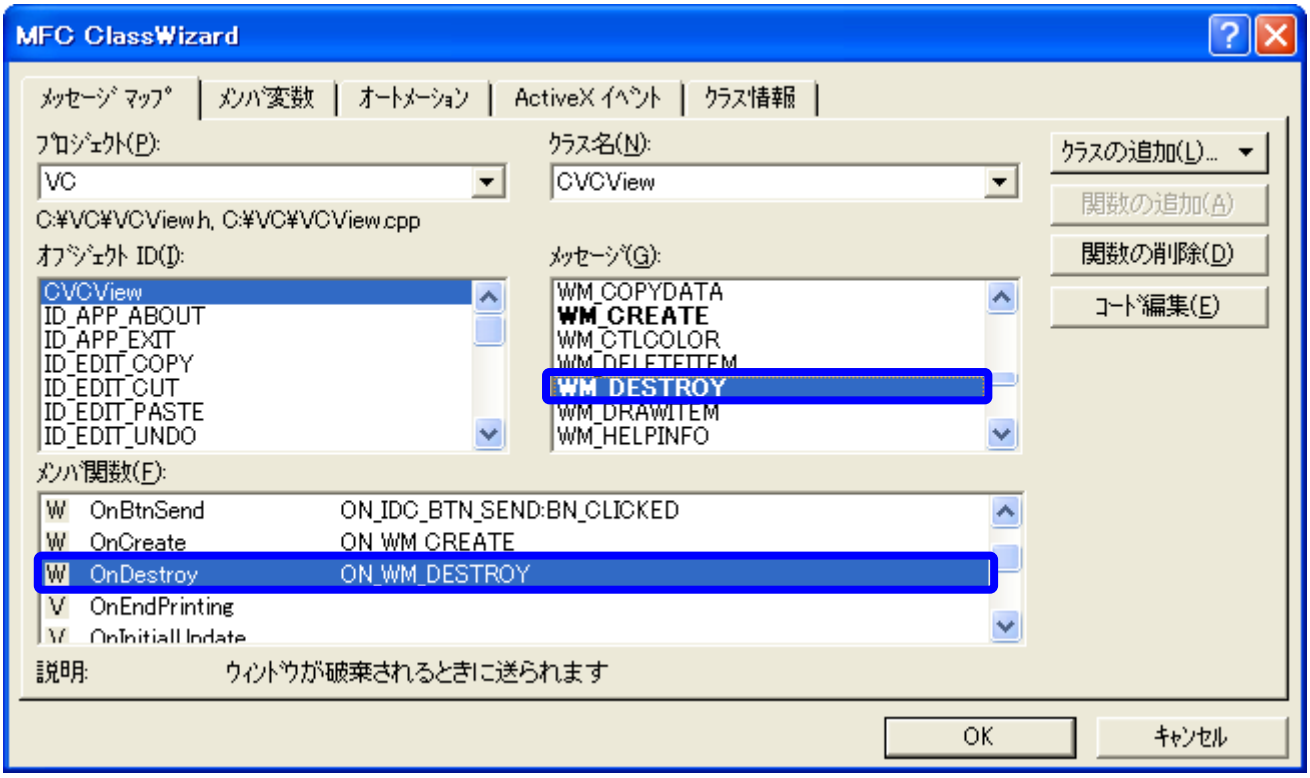


キャッツ株式会社
ラウンドロビンエンジニアリング

外観図が終了時に ZIPC と切断するように、実装する手順を説明します。

【プロジェクト作成手順】

1. ”WM_DESTROY”メッセージのハンドラ（”OnDestroy”関数）を定義します。



2. ”OnDestroy”関数にて、ZIPC と切断する関数をコールします。

■サンプル例

```
void CVCView::OnDestroy()
{
    CFormView::OnDestroy();

    // TODO: この位置にメッセージ ハンドラ用のコードを追加してください

    // ZIPC と切断する
    VIP_PnlCut( _T(""), GetSafeHwnd() );
    VIP_FreePnlIF();
}
```

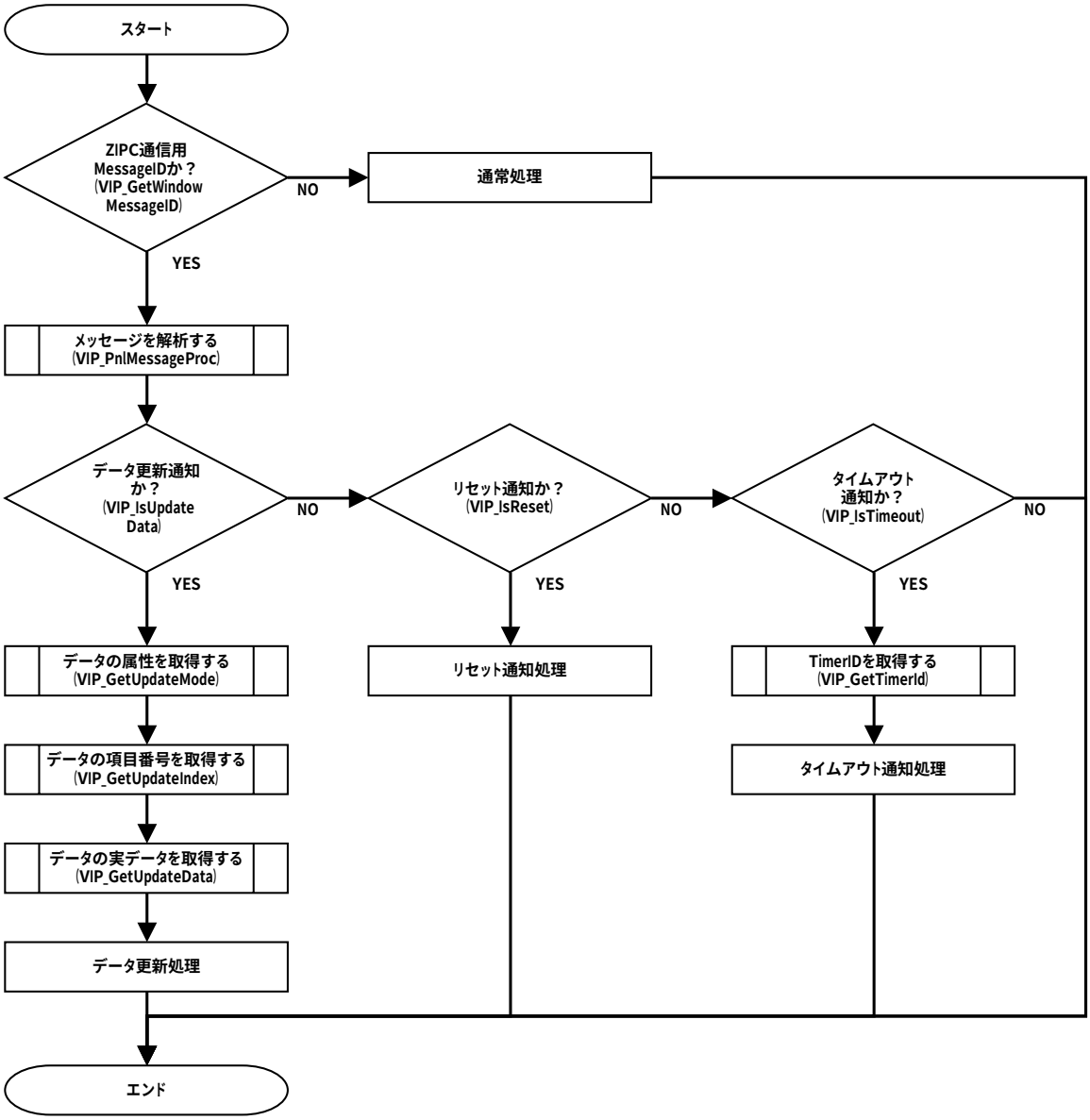
キャッツ株式会社
ラウンドロビンエンジニアリング

３.４　メッセージ受信処理

本章では、外観図がメッセージ受信時に ZIPC からのメッセージを解析するように、実装する方法を説明します。

外観図がメッセージ受信時に ZIPC からのメッセージを解析するための、メッセージ受信処理フローを下図に示します。

■メッセージ受信処理フロー

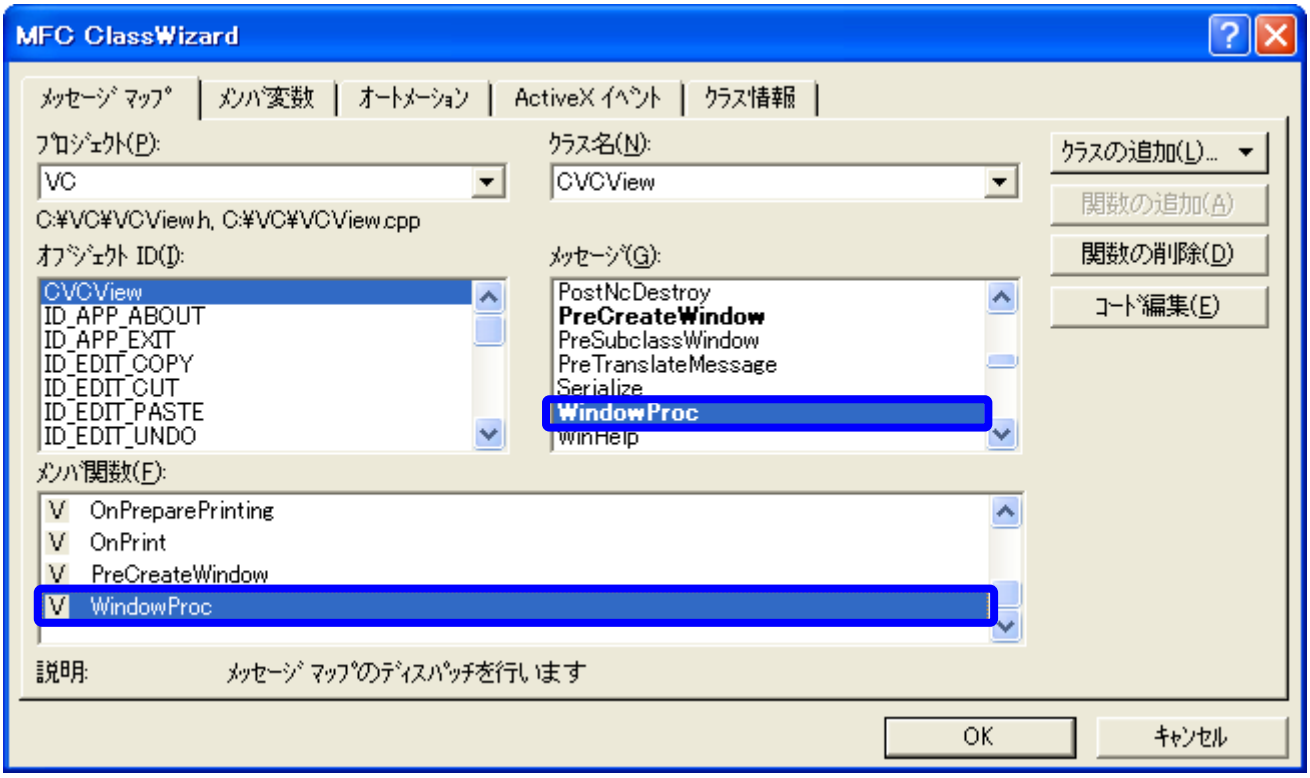


キャッツ株式会社
ラウンドロビンエンジニアリング

外観図がメッセージ受信時に ZIPC からのメッセージを解析するように、実装する手順を説明します。

【プロジェクト作成手順】

1. ”WindowProc”メッセージのハンドラ（”WindowProc”関数）を定義します。



2. ”WindowProc”関数にて、ZIPC からのメッセージを解析する関数をコールします。

■サンプル例

```
LRESULT CVCView::WindowProc(UINT message, WPARAM wParam, LPARAM lParam)
{
    LRESULT     lRet;

    // VIP 専用の通信メッセージ内容か？
    if( message == VIP_GetWindowMessageID() ) {
        // VIP 専用 Window メッセージ解析処理を行う
        lRet = VIP_PnlMessageProc( wParam, lParam );
        // 更新情報が取得されているか？
        if( VIP_IsUpdateData() != FALSE ) {
            // View の更新処理を行う
            UpdateViewData();
        }
        // リセット通知が行われた？
        else if( VIP_IsReset() != FALSE ) {
            // リセット時の処理を行う
        }
        else{
            // none
        }
    }
    else {
        lRet = CFormView::WindowProc(message, wParam, lParam);
    }

        return  lRet;
}
```

```
LRESULT CVCView::WindowProc(UINT message, WPARAM wParam, LPARAM lParam)
{
    LRESULT     lRet;

    // VIP 専用の通信メッセージ内容か？
    if( message == VIP_GetWindowMessageID() ) {
        // VIP 専用 Window メッセージ解析処理を行う
        lRet = VIP_PnlMessageProc( wParam, lParam );
        // 更新情報が取得されているか？
        if( VIP_IsUpdateData() != FALSE ) {
            // View の更新処理を行う
            UpdateViewData();
        }
        // リセット通知が行われた？
        else if( VIP_IsReset() != FALSE ) {
            // リセット時の処理を行う
        }
        else{
            // none
        }
    }
    else {
        lRet = CFormView::WindowProc(message, wParam, lParam);
    }

        return  lRet;
}
```

```
void CVCView::UpdateViewData()
{
    // ダイアログ情報取得
    UpdateData(TRUE);

    switch( VIP_GetUpdateMode() ){
        case ZIPCVIP_DATA_PIO:  // 出力データの属性がポート項目
            if( VIP_GetUpdateIndex() == 0 ){
                m_strRecv.Format( "%d", *(char*)VIP_GetUpdateData() );
            }
            else {
                // none
            }
            break;

        default:
            // none
            break;
    }

    // ダイアログ情報更新
    UpdateData(FALSE);

    return;
}
```

４．外観図にて使用する定数

本章では、外観図にて使用する定数を説明します。

４.１　データ属性（ZIPCVIP_DATAMODE）

■データ属性定数一覧（タグ名：ZIPCVIP_DATAMODE）

NO	名前	値	説明
1	ZIPCVIP_DATA_PIO	0x00	ポート属性
2	ZIPCVIP_DATA_ANG	0x01	アナログ属性
3	ZIPCVIP_DATA_STR	0x02	文字列属性
4	ZIPCVIP_DATA_UNDEF	0x03	未定義

キャッツ株式会社
ラウンドロビンエンジニアリング

５．外観図にて使用する関数

本章では、外観図にて使用する関数を説明します。

５.１ ZVipPnl.DLL をロードする（VIP_LoadPnlIF）

■構文

```
BOOL VIP_LoadPnlIF( CString strFile )
```

■戻り値

型 ：BOOL

説明：

NO	名前	説明
1	TRUE	ZVipPnl.DLL のロードに成功
2	FALSE	ZVipPnl.DLL のロードに失敗

■引数

なし。

■引数１

型 ：CString

名称：strFile

説明：”ZVipPnl.DLL”ファイル

5.2 ZVipPnl.DLL を開放する（VIP_FreePnlIF）

■構文

```
void VIP_FreePnlIF( void )
```

■戻り値

なし。

■引数

なし。

5.3 ZIPC と接続する（VIP_PnlConnect）

■構文

```
BOOL VIP_PnlConnect( LPCSTR lpszClassName, HWND hWnd )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	ZIPC との接続に成功
2	FALSE	ZIPC との接続に失敗

■引数１

型　：LPCSTR

名称：lpszClassName

説明：接続元となるウィンドウクラス名称

■引数２

型　：HWND

名称：hWnd

説明：接続元となるウィンドウハンドル値

5.4 ZIPC と切断する（VIP_PnlCut）

■構文

```
BOOL VIP_PnlCut( LPCSTR lpszClassName, HWND hWnd )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	ZIPC との切断に成功
2	FALSE	ZIPC との切断に失敗

■引数１

型　：LPCSTR

名称：lpszClassName

説明：接続元となるウィンドウクラス名称

■引数２

型　：HWND

名称：hWnd

説明：接続元となるウィンドウハンドル値

5.5 ZIPC 通信用 MessageID を取得する（VIP_GetWindowMessageID）

■構文

```
UINT VIP_GetWindowMessageID( void )
```

■戻り値

型　：UINT

説明：ZIPC 通信用 MessageID

■引数

なし。

５.６　メッセージを解析する（VIP_PnlMessageProc）

■構文

```
BOOL VIP_PnlMessageProc( WPARAM wParam, LPARAM lParam )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	メッセージ解析に成功
2	FALSE	メッセージ解析に失敗

■引数１

型　：WPARAM

名称：wParam

説明：メッセージの WPARAM 値

■引数２

型　：LPARAM

名称：lParam

説明：メッセージの LPARAM 値

5.7　リセット通知であることを確認する（VIP_IsReset）

■構文

```
BOOL VIP_IsReset( void )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	リセット通知である
2	FALSE	リセット通知でない

■引数

なし。

5.8　データ更新であることを確認する（VIP_IsUpdateData）

■構文

```
BOOL VIP_IsUpdateData( void )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	データ更新である
2	FALSE	データ更新でない

■引数

なし。

５.９　タイムアウト通知であることを確認する（VIP_IsTimeout）

■構文

```
BOOL VIP_IsTimeout( void )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	タイムアウト通知である
2	FALSE	タイムアウト通知でない

■引数

なし。

５.１０　データの属性を取得する（VIP_GetUpdateMode）

■構文

```
enum ZIPCVIP_DATAMODE VIP_GetUpdateMode( void )
```

■戻り値

型　：enum ZIPCVIP_DATAMODE

説明：データ属性（データ属性参照。）

■引数

なし。

５.１１　データの項目番号を取得する（VIP_GetUpdateIndex）

■構文

```
int VIP_GetUpdateIndex( void )
```

■戻り値

型　：int

説明：データの項目番号

■引数

なし。

５.１２　データの実データを取得する（VIP_GetUpdateData）

■構文

```
void* VIP_GetUpdateData( void )
```

■戻り値

型　：void*

説明：実データが格納されているアドレス

■引数

なし。

５.１３　TimerID を取得する（VIP_GetUpdateData）

■構文

```
long VIP_GetTimerId( void )
```

■戻り値

型　：long

説明：TimerID

■引数

なし。

５.１４　ポートにデータを設定する（VIP_SetPortData）

■構文

```
BOOL VIP_SetPortData( long nNo, LPVOID pData, long nParam )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	ポートへの値設定に成功
2	FALSE	ポートへの値設定に失敗

■引数１

型　：long

名称：nNo

説明：ポート項目番号

■引数２

型　：LPVOID

名称：pData

説明：データが格納されているアドレス

■引数３

型　：long

名称：nParam

説明：データの配列要素数

５.１５　アナログにデータを設定する（VIP_SetAngData）

■構文

```
BOOL VIP_SetAngData( long nNo, double dData )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	アナログへの値設定に成功
2	FALSE	アナログへの値設定に失敗

■引数１

型　：long

名称：nNo

説明：アナログ項目番号

■引数２

型　：double

名称：dData

説明：設定データ

キャッツ株式会社
ラウンドロビンエンジニアリング

５.１６　文字列にデータを設定する（VIP_SetStrData）

■構文

```
BOOL VIP_SetStrData( long nNo, LPCSTR pData, long nParam )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	文字列への値設定に成功
2	FALSE	文字列への値設定に失敗

■引数１

型　：long

名称：nNo

説明：文字列項目番号

■引数２

型　：LPCSTR

名称：pData

説明：文字列が格納されているアドレス

■引数３

型　：long

名称：nParam

説明：文字列長

５.１７　名称イベントを発行する（VIP_SendNameEvent）

■構文

```
BOOL VIP_SendNameEvent( CString strTask, CString strEvent )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	名称イベント発行に成功
2	FALSE	名称イベント発行に失敗

■引数１

型　：CString

名称：strTask

説明：発行先タスク名

■引数２

型　：CString

名称：strEvent

説明：発行イベント名

５.１８　シミュレーション同期タイマーを設定する（VIP_SetVipTimer）

■構文

```
BOOL VIP_SetVipTimer( long nTimerID, unsigned long ulHiTime, unsigned long ulLoTime )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	シミュレーションタイマーの設定に成功
2	FALSE	シミュレーションタイマーの設定に失敗

■引数１

型　：long

名称：nTimerID

説明：TimerID

■引数２

型　：unsigned long

名称：ulHiTime

説明：設定時間(上位 32Bit)

■引数３

型　：unsigned long

名称：ulLoTime

説明：設定時間(下位 32Bit)

５.１９　シミュレーション同期タイマーを解除する（VIP_KillVipTimer）

■構文

```
BOOL VIP_KillVipTimer( long nTimerID )
```

■戻り値

型　：BOOL

説明：

NO	名前	説明
1	TRUE	シミュレーションタイマーの解除に成功
2	FALSE	シミュレーションタイマーの解除に失敗

■引数１

型　：long

名称：nTimerID

説明：TimerID

６．追加ファイル説明

６.１　ZVipPnlImp.h

ZIPC との動的リンクにて使用するヘッダファイルです。
ZIPC にて定義している関数の型　および　定数を定義します。
※　本ファイルに定義している内容に関しては、ZIPC マニュアルを参照して下さい。

６.２　VipUpdateData.h

ZIPC VIP データ管理クラス（CVipUpdateData）のヘッダファイルです。
CVipUpdateData クラスを定義します。

６.３　VipUpdateData.cpp

ZIPC VIP データ管理クラス（CVipUpdateData）のソースファイルです。
CVipUpdateData クラスの実態を定義します。

６.４　VipCommunicate.h

外観図に公開するヘッダファイルです。
外観図にて使用する関数　および　定数を定義します。

６.５　VipCommunicate.cpp

ZIPC と外観図を通信するためのソースファイルです。
外観図にて使用する関数の実態を定義します。