CaoScript

ユーザーズ ガイド

Version 1.5.9

October 22, 2010

【備考】

【改版履歴】

日付	版数	内容
2006-03-09	1.0	初版
2006-09-02	1.1	CaoController クラスの OnMessage イベントを受信機能追加
2006-12-04	1.1.1	プロセス優先度の変更機能追加
2007-01-12	1.2.0	CaoWorkspace 共有オブジェクト('gcao')を追加
2007-03-30	1.2.1	使いやすさ向上. RunScript の機能拡張. Pnl, Mgr オブジェクトの追加
2007-04-17	1.2.2	App オブジェクトに OnErrorGoto メソッド追加
2007-06-13	1.3.0	Vars による Break Point 指定, サイクル起動, Error オブジェクトの追加
2007-08-20	1.3.1	CaoScript の RCW 追加. Manager の使いやすさ向上
2007-11-15	1.3.2	Mgr.Count, App.Filename プロパティ追加. スケジュール機能強化
2007-12-25	1.3.3	Ext と Dat オブジェクト追加し, App オブジェクトの関数の一部を移動
2008-01-08	1.4.0	Mgr オブジェクトを Automation インターフェースとして公開
2008-01-19	1.4.1	App.Path プロパティ追加. 'Gcao'から'Cao'に変更
2008-03-11	1.4.2	CaoScriptMan のタスクトレイ登録機能追加
2008-03-28	1.4.3	App.ActivateWindow, App.SendKeyCode メソッド追加
2008-04-18	1.4.4	Var.DateTime, Var.Changed, Var.Bin8/16/32 プロパティ追加
2008-04-21	1.4.5	Vars 領域にローカルメモリ領域追加. Vars.ItemValue プロパティ追加
2008-04-25	1.4.6	Csv オブジェクト, マクロ定義機能追加
2008-06-23	1.4.7	App.Priority プロパティの仕様変更
2008-07-02	1.4.8	Vars.Help の保存機能追加
2008-07-03	1.4.9	Execute("GetScriptStarted")の追加
2008-07-11	1.5.0	On Error ステートメントの有効化
2008-08-27	1.5.1	付録 VBScript リファレンス の追加
2008-11-12	1.5.2	Csv.LoadFile/.SaveFile メソッド追加. Vars サムネイル表示
2009-05-19	1.5.3	ローカル変数ウォッチ機能追加. ユーザ共有オブジェクト機能追加
2009-06-12	1.5.4	Vars イベント機能追加
2009-07-20	1.5.5	BeepEx ステートメント追加
2009-11-02	1.5.6	ユーザ共有オブジェクトの作成方法追記
2009-11-06	1.5.7	Vars.Macro 機能追加.
2010-05-13	1.5.8	Vars イベント機能の高速化(仕様変更)
2010-10-22	1.5.9	StartScript／StopScript ウィンドウ表示レスステップ送り・停止の追加

目次

1. 概要

CaoScriptツールは最もシンプルなCAOプログラム開発環境です．アプリケーションMicrosoft Visual Baiscや Visual C++などの統合開発環境がなくても，CAOスクリプト言語で簡単なCAOアプリケーションプログラムをこのツールだけで作成することができます．

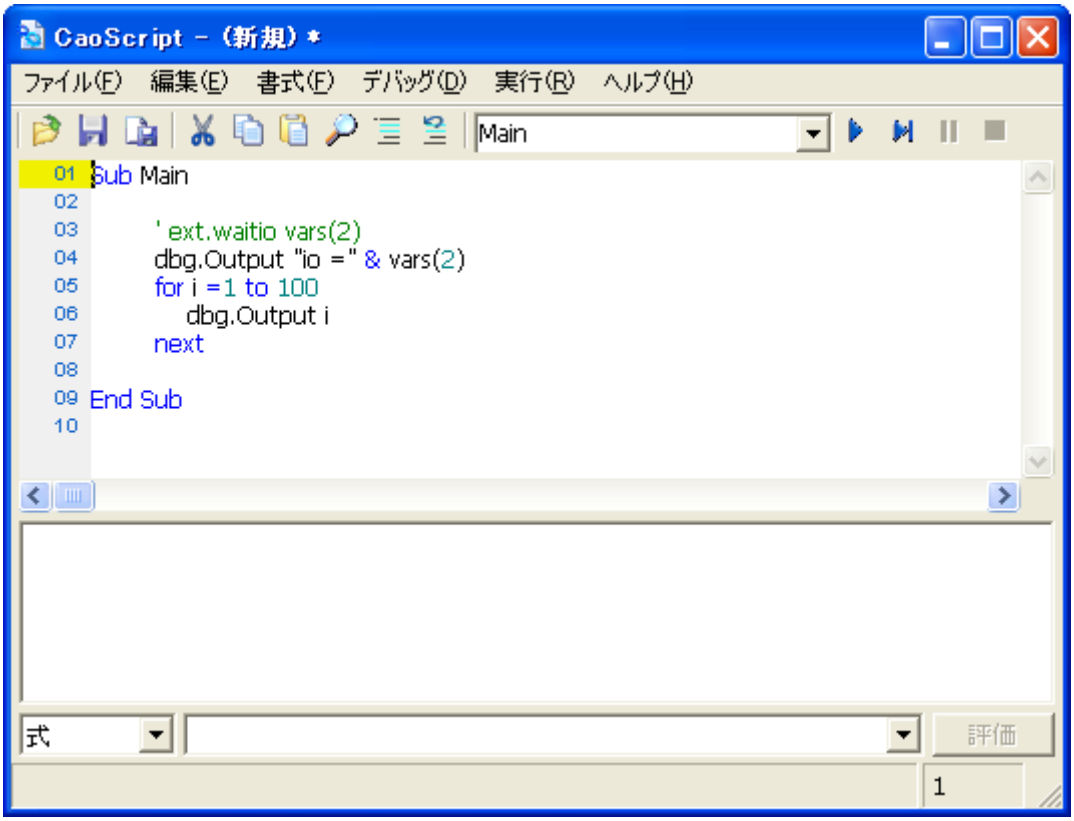


図 1-1 CaoScript 実行画面

2. CaoScript の操作

2.1. CaoScript の画面構成

CaoScript の画面は，スクリプト入力ウィンドウ，ログウィンドウ，デバッグウィンドウからなっています．

スクリプト入力ウィンドウに，CaoScript 言語を記述し実行します．実行前，実行中にエラーなどが起きた場合やプログラムがログ出力するような命令を実行した時にはログウィンドウにメッセージを表示します．

また，デバッグウィンドウではグローバル変数の式，関数の評価や，ローカル変数のモニタなどを行う場合に使用します．

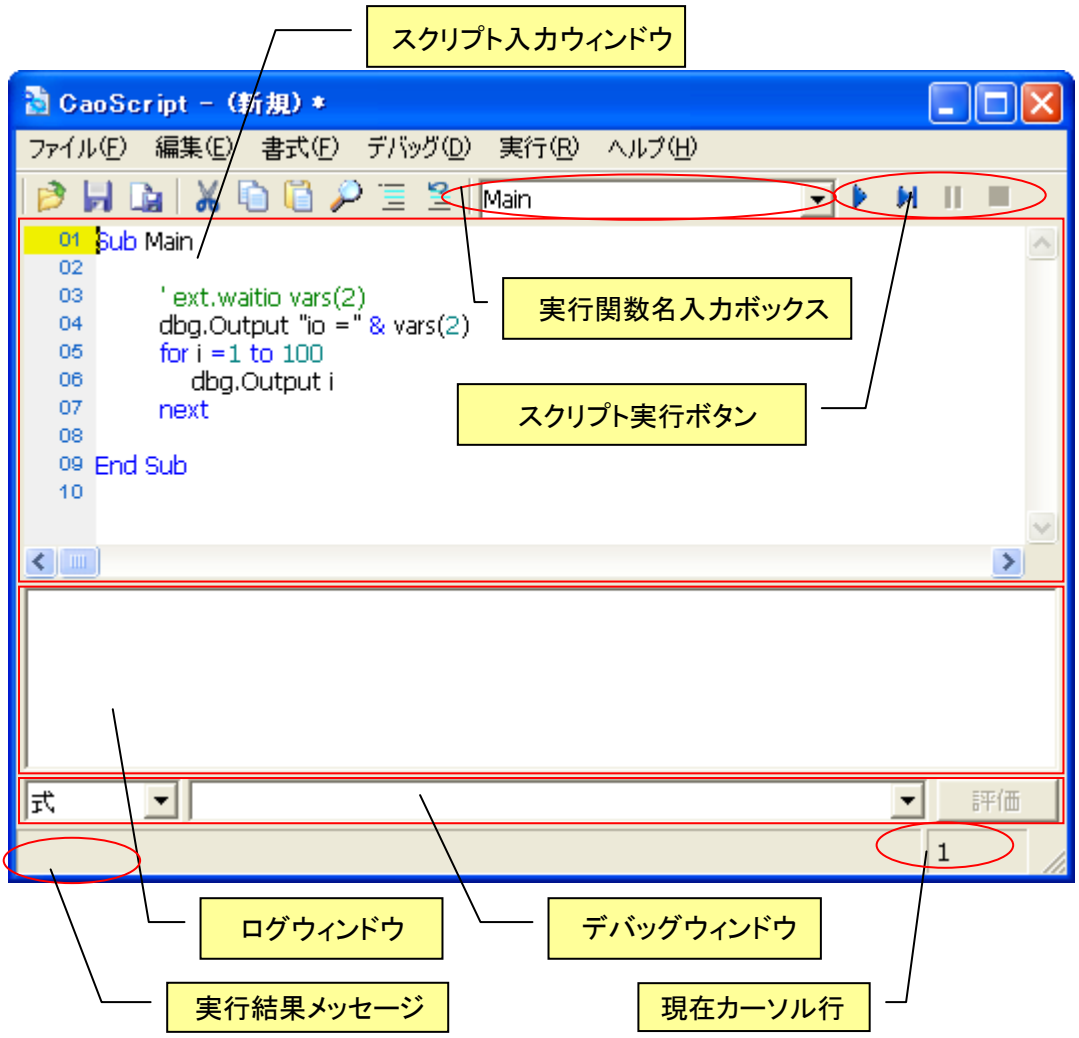


図 2-1 CaoScript 画面構成

2.2. CaoScript の機能

CaoScript の編集時には文字検索，置換，行ジャンプ，行マーカなども使用することができます．また，実行する関数をスタートアップに指定することにより実行を開始し，実行中には Breakpoint 機能を使用することで CaoScript を一時停止し，ステップ実行するなどのデバッグ実行することもできます．

2.2.1. CaoScript の実行と停止

2.2.1.1. 実行

CaoScript 言語を入力したら，次に実行します．実行するには，「実行」メニューの「開始」を選択します．

また，CaoScript 言語をステップ毎に実行したい場合は，「実行」メニューから「ステップオーバー」を選択します．「実行」，「ステップオーバー」等の実行メニューはツールバーからの実行も可能です．

このメニューからの実行による実行対象は図 2-1 の「実行関数名入力ボックス」に設定している関数名です．関数名が間違って指定されている場合は，エラーのメッセージが出ます． 何も入力をしなければ関数の外のプログラムが実行されます．CaoScript 実行中はプログラムの編集は行えません．（もちろん，スタートアップなども編集できません）

2.2.1.2. 停止

CaoScriptを停止する場合は，「実行」メニューの「停止」を選択します．一時的にプログラムを停止したい場合は，「中断」を選択します．

2.2.1.3. サイクル起動 / サイクル停止

「実行」メニューの「サイクル起動」は，プログラムを繰返し実行したい場合に，このモードで起動するとプログラムは停止した後，再度実行されます． ただし，サイクルされるのは「実行関数名入力ボックス」に設定されている関数です．

2.2.2. CaoScript のデバッグ

CaoScript のデバッグには，ブレークポイント機能の使用やデバッグウィンドウの評価機能，変数モニタ機能，dbg オブジェクトのログ出力などがあります．ここでは，ブレークポイント機能，デバッグウィンドウの評価機能，変数ウォッチ機能について説明します．

2.2.2.1. ブレークポイント機能

ブレークポイント機能は，プログラムの実行中に任意の行（ステップ）でプログラムを中断状態にすることができる機能です． CaoScript のブレークポイント機能を使用することで，プログラムのデバッグや，ある処理までは実行，その後はステップオーバーで実行といった検証する場合に非常に役立ちます．

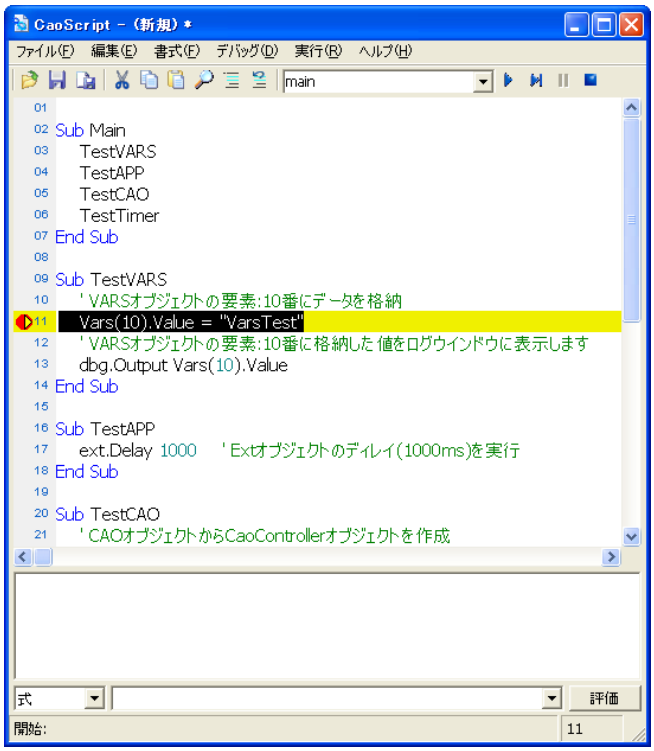


図 2-2 ブレークポイント機能でのプログラム中断の様子

ブレークポイント機能を使用するためには，まずブレークポイントを設定します．ブレークポイントの設定には二通りの設定方法があります．

一つ目は，「デバッグ」メニューの「ブレークポイント...」を選択します． ユーザがブレークポイントで中断したい行(ステップ)を登録します．この設定はブレークポイントを複数登録したい場合などに行うとよいでしょう．
複数のブレークポイントを設定した場合の動作は，最初のブレークポイントで中断したプログラムを継続実行した時に，次のブレークポイントで中断されます．

二つ目の設定は，CaoScript 入力ウィンドウ上で選択しているカレント行(ステップ)を一つずつ登録していきます．この設定は「デバッグ」メニューの「ブレークポイントの設定/解除」を選択します． このメニューは，キーボードのファンクションキー[F9]でも簡単に設定/解除が行えます．手軽に設定したい場合はこちらの方法で設定します．

ブレークポイントの解除も，設定と同様に「デバッグ」メニューの「ブレークポイント...」から登録したリストから対象を選択し「削除」するか，「すべて解除」を選択します．または，CaoScript 入力ウィンドウ上で登録されているブレークポイント行で再度，「デバッグ」メニューの「ブレークポイントの設定/解除」を選択するか，キーボードのファンクションキー[F9]を押すことでも解除することができます．

2.2.2.2. デバッグウィンドウの評価機能，変数ウォッチ機能

CaoScript のデバッグ作業の中で，もう一つ便利な機能の紹介です．図 2-1 のデバッグウィンドウでグローバル変数の式，値を返す関数評価，変数ウォッチを行います．この機能はプログラムが実行中(中断状態可)の場合に有効です．

グローバル変数の式評価は CaoScript 上のグローバルな変数に対して行います．

例えば，グローバル変数”A”を宣言している CaoScript があると仮定し，ある箇所で”123”を代入しているとします．デバッグウィンドウに「式」，「A = 123」と指定し「評価」ボタンを押すと，代入処理が終わっていればログウィンドウに結果”True”が出力されます．また「式」，「A」と指定し，変数名のみとして「評価」ボタンを押すと，「A」の現在値がログウィンドウに出力されます．

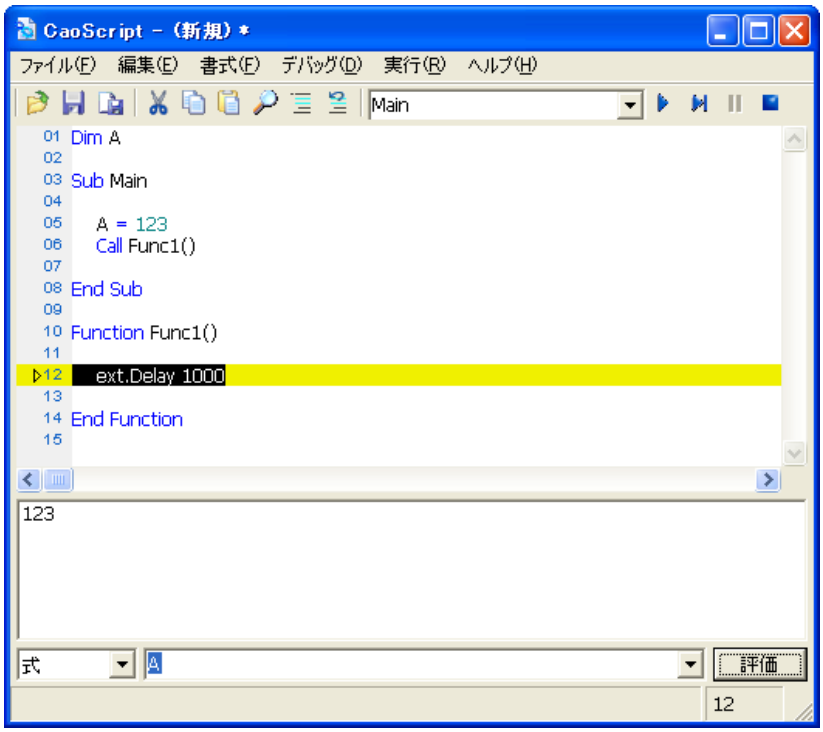


図 2-3 グローバル変数の評価

さらに，「式」の場合に限り，値を返す関数(Function)名を指定して「評価」をすることができます．この場合，「評価」ボタンを押すと，関数の返り値が設定されていればその値がログウィンドウに出力されます．また，「関数名 = 値」を指定しても評価結果を得られます．

変数ウォッチ機能は，ある関数内で宣言されたローカル変数を事前にウォッチ登録することで値をモニタすることができます．

例えば，Main 関数中にローカル変数”A”を宣言していると仮定します．ウォッチ登録するには，デバッグウィンドウに「変数名」，「A」と指定して，「ウォッチ」ボタンを押します．登録が正常に行われたらログウィンドウに”Watch on : A”と表示されます．この登録が行われていないと変数をモニタすることはできません． この変

数ウォッチ機能はグローバル変数評価機能と異なり，変数スコープが影響します．したがって，ローカル変数をモニタするには実行中の関数内のローカル変数である必要があります．

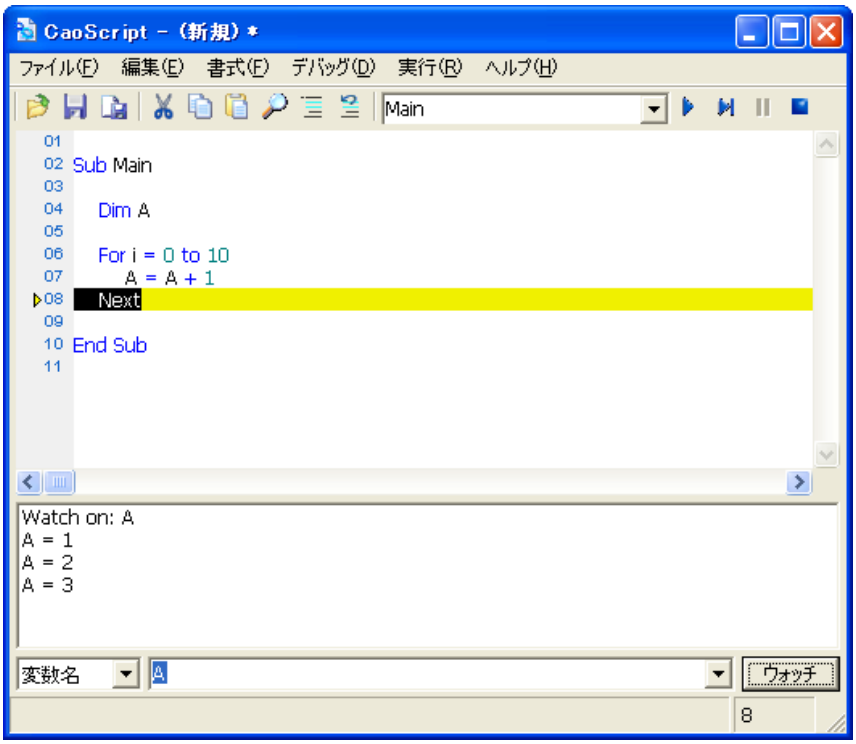


図 2-4 ローカル変数ウォッチの様子

ここで注意しておくことがあります．「式」「変数名」のどちらにおいても指定した変数名がプログラム上に存在しない場合や評価式の構文が正常でない場合は，エラーが発生します．このエラーが出ると現在実行中のプログラムは停止します．

3. CAO スクリプト言語

CAO スクリプト言語とは,

CAO スクリプト ＝ VBScript＋ 組込み 6 オブジェクト

という関係にある Microsoft Script をベースにした言語です. したがって, その文法や機能はベース言語である VBScript と全く同一ですので, CaoScript から Visual Basic への移植が簡単に行うことができます. VBScript の言語リファレンスは下記の URL 等を参考にしてください.

http://www.microsoft.com/japan/msdn/scripting/

3.1. 組込みオブジェクト

CaoScript の組込みオブジェクトには表 3-2 の4つのグローバルオブジェクトと表 3-2 の 2 つのローカルオブジェクトがあります. グローバルオブジェクトの名前はオブジェクト名と一致しており, システムによって一つだけ起動時に作成され登録されています. グローバルオブジェクトをプログラムで作成することはできません. また, ローカルオブジェクトはグローバルオブジェクトの関数実行で生成されます.

表 3-1 組込みグローバルオブジェクト一覧

オブジェクト名	説明
CAO	CaoWorkspace オブジェクト
VARS	VAR オブジェクトのコレクションオブジェクト
DBG	デバッグオブジェクト
APP	アプリケーションオブジェクト
DAT	データ変換オブジェクト
EXT	その他の機能拡張オブジェクト

表 3-2 組込みローカルオブジェクト一覧

オブジェクト	説明
CAOMESS	CaoMessage イベントシンクオブジェクト. APP オブジェクトの Attach 関数で生成されます.
VAR	バックアップ変数オブジェクト. VARS オブジェクトの Item プロパティで取得できます. (Item プロパティはコレクションオブジェクトのデフォルトプロパティですので省略することもできます.) VAR 変数に格納された値は永続的に保存されます. 配列も含めて様々なデータ型を格納できますが, 配列は 1 次元に限定されています. デフォルトの配列サイズは 100 (インデックス範囲 : 1～100) です. サイズは

	CaoConfig を使用して，DataStore プロバイダの[**Parameter**]で変更できます．例えばサイズを 500 に変更したい場合は ”ItemMax=500” と入力します．（図 3-1 参照） 【要注意】 OnMessage イベントプロシージャは任意のタイミングでメイン処理に割り込みがかかります．したがって VAR オブジェクト等のオブジェクトをメイン処理と共有する場合は意図しない動作になる可能性もあるので同時に同じオブジェクトを操作するようなプログラムは作成しないように注意してください．
ERROR	エラーオブジェクト．APP オブジェクトの Error プロパティで取得できます．OnErrorGoto でトラップされた場合は VBScript の Err オブジェクトではなく，このオブジェクトにエラー情報が格納されています．エラー情報は VBScript の Err オブジェクトのコピーで，VBScript の Err オブジェクトはクリアされています．

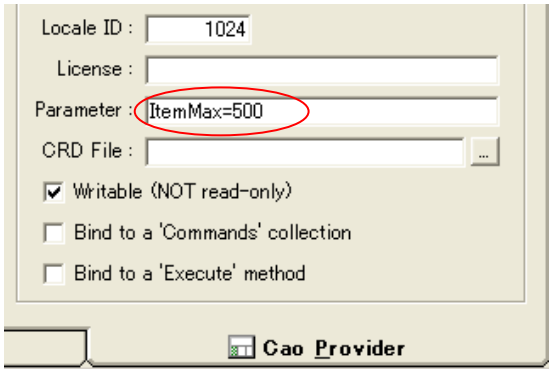


図 3-1 CaoConfig VARS 配列数設定

これらのオブジェクトの機能概要を下記にまとめます．CAO オブジェクトに関してはデフォルトの CaoWorkspace オブジェクトそのものですので，CAO の仕様書を参考にしてください．ここでは，残りの組込みオブジェクトについて解説します．

表 3-3 VARS オブジェクトメンバー覧

メンバ	機能
Property Get Item (Index As Variant) As clsVar	指定されたインデックス番号を ID に設定し，VAR オブジェクトを取得します．
Property Get ItemValue (Index As Variant) As Variant	指定されたインデックス番号の値を取得します．

Property Let ItemValue (Index As Variant, newVal As Variant)	指定されたインデックス番号の値を設定します.

表 3-4 VAR オブジェクトメンバー覧

メンバ	機能
Property Get Bin8 () As Byte	連続する 8 個分の VARS 変数をブール型に変換した後, バイト型にパックして取得します.
Property Let Bin8 (cVal As Byte)	バイト型変数を連続する 8 個分の VARS 変数にアンパックして設定します. 各 VARS 変数はブール型になります.
Property Get Bin16 () As Integer	連続する 15 個分の VARS 変数をブール型に変換した後, 整数型にパックして取得します. (注) 符号ビットは使われません.
Property Let Bin16 (iVal As Integer)	整数型変数を連続する 15 個分の VARS 変数にアンパックして設定します. 各 VARS 変数はブール型になります. (注) 符号ビットは使われません.
Property Get Bin32 () As Long	連続する 31 個分の VARS 変数をブール型に変換した後, 長整数型にパックして取得します. (注) 符号ビットは使われません.
Property Let Bin32 (lVal As Long)	長整数型変数を連続する 31 個分の VARS 変数にアンパックして設定します. 各 VARS 変数はブール型になります. (注) 符号ビットは使われません.
Property Get Updated () As Boolean	値が更新された(True)か否(False)かを取得します. (注) ローカルメモリ領域の場合は使えません.
Property Get Changed () As Boolean	値が変化した(True)か否(False)かを取得します. (注 1) ローカルメモリ領域の場合は使えません. (注 2) CaoScript バージョン 1.4 以前は Updated プロパティの動作をします.
Property Let Changed (bFlag As Boolean)	値の変更フラグを設定します. (注) ローカルメモリ領域の場合は使えません.
Property Get DateTime () As Variant	値が更新された時刻を取得します. (注 1) DateTime の分解能はミリ秒です. (注 2)ローカルメモリ領域の場合は使えません.
Property Get Help () As String	ヘルプ文字列を取得します. (注 1) ヘルプ文字列を編集するには DataStore プロバイダフォルダにある datastore.mdb の vars テーブルを直接編集して

	ください. (注 2) ローカルメモリ領域の場合は使えません.
Property Get Macro () As String	マクロ文字列を取得します. (注 1) マクロ文字列を編集するには DataStore プロバイダフォルダにある datastore.mdb の vars テーブルを直接編集してください. (注 2) ローカルメモリ領域の場合は使えません.
Property Get ID () As Variant	ID を取得します.
Property Let ID (newVal as Variant)	ID を設定します.
Property Get Value () As Variant	値を取得します.
Property Let Value (newVal As Variant)	値を設定します. VARS 変数に格納された値は永続的に保存されます.

表 3-5 DBG オブジェクトメンバー一覧

メンバ	機能
Property Get Destination () As Integer	Output のデフォルト出力先を取得します.
Property Let Destination (iDest As Integer)	Output のデフォルト出力先を設定します. Destination プロパティの初期値は 0 です. 0: CaoScript のログウインドウへ出力. 1: Win32 の OutputDebugString() API で出力. (例) Dbg. Destination = 1
Sub ClearLog ()	ログウインドウの表示をクリアします.
Sub Output (Optional vntMsg As Variant, Optional iDest as Integer)	メッセージを出力します. 出力先は下記のように指定できます. 省略した場合は, Destination プロパティで設定された値になります. (例) Dbg.Output “hello”, 1
Sub Stopoff ()	実行を一時停止状態にします. [Pause]ボタンを押すのと同じ動作です.

表 3-6 APP オブジェクトメンバー一覧

メンバ	機能
Property Get Error () As clsError	エラーオブジェクトを取得します.

Property Get ExitCode () As Variant	終了コードを取得します.
Property Let ExitCode (vntRet As Variant)	終了コードとして任意の値を格納します. 格納した値は次に上書きされるまで保存されています.
Property Get FileName () As String	ファイル名を取得します.
Property Get Path () As String	ファイルパスを取得します.
Property Get Priority() As Long	現在のスレッド優先度を取得します.
Property Let Priority(newVal As Long)	スレッド優先度を設定します. ■ 優先度(低い順) THREAD_PRIORITY_IDLE = -15 THREAD_PRIORITY_LOWEST = -2 THREAD_PRIORITY_BELOW_NORMAL = -1 THREAD_PRIORITY_NORMAL = 0 THREAD_PRIORITY_ABOVE_NORMAL = 1 THREAD_PRIORITY_HIGHEST = 2 THREAD_PRIORITY_TIME_CRITICAL = 15
Property Get ThreadId() As Long	現在のスレッド ID を取得します.
Property Get TimerInterval() As Long	タイマイベントの発生間隔をミリ秒単位で取得します. タイマイベントが無効のときは, 0 を取得します.
Property Let TimerInterval(newVal As Long)	タイマイベントの発生間隔をミリ秒単位で設定します. 0 を設定した時は, タイマイベントは無効になります.
Sub AddScript(strFilename As String, Optional vntOpt As Variant)	指定されたファイルに保存されたスクリプトを動的に追加します. (例) app.AddScript “Share.vbs”
Function Attach (caoCtrl As Variant) As clsCaoMess	CAO オブジェクトで追加した CaoController オブジェクトの OnMessage イベントを受信できるようにします. この関数実行しないと App_OnMessage イベントは発生しません.
Sub Clear()	タイマイベントの発生間隔のクリア. タイマイベントを無効に設定します.
Sub OnErrorGoto (strSub As String)	エラー発生時にジャンプするサブルーチンを設定します. 引数を渡すこともできます. 空文字("")を指定すると設定は解除されます. (例) app.OnErrorGoto "ErrHandler 123"

Function RunScript(strFilename As String, strStartup As String, Optional lStartMode As Long = 1, Optional bStopUnload As Boolean = True) As CaoScript.clsEngine	指定されたファイルに保存されたスクリプトを非同期に実行します. strStartup と lStartMode はそれぞれ Automation インターフェースの Startup プロパティ, StartScript の第 1 引数と同じ意味です. bStopUnload が True の場合は生成された CaoScript.clsEngine オブジェクトが破棄されるときにプログラムが実行中であれば停止させます. False の場合はそのままプログラムの実行が継続されます. (例) obj = app.RunScript (“Multi.vbs”, “Main”) 【備考】 Mgr オブジェクトの Script プロパティを使って StartScript を実行する方がパフォーマンスに優れています.
Function ShellExec (vntPath As Variant, Optional vntStyle As Variant) As Double	実行可能プログラムを実行し, そのプログラムのタスク ID を返します. プログラムの実行に問題が発生した場合は 0 を返します. 詳細仕様は Visual Basic 言語の Shell 関数と同じです. (例) app.ShellExec(“notepad.exe”) ’ メモ帳の起動
Sub ActivateWindow (vntTitle As Variant, Optional bWait As Boolean = False)	アプリケーションウィンドウをアクティブにします. 詳細仕様は Visual Basic 言語の AppActivate ステートメントと同じです.
Sub SendKeyCode (strKeys As String, Optional bWait As Boolean = False)	キーストロークまたはキーストロークの組み合わせを, キーボードから入力したときと同様にアクティブウィンドウに渡します. 詳細仕様は Visual Basic 言語の SendKeys ステートメントと同じです.

表 3-7 DAT オブジェクトメンバー覧

メンバ	機能
Function BstrFromVariant (vntVal as Variant) as Bstr	Variant オブジェクトを RAC 文字列に変換します. (例) strTest = dat.BstrFromVariant(vntValue)
Function ChangeType (vntSrc As Variant,vt As Integer) As Variant	任意のデータ型に変換します. 配列にも対応していますが一次元配列に限定されています. 利用できる型については表 3-11 を参照して下さい. (例) dat.ChangeType(Array(1,2,3), 4) ’ R4(4)配列に変換
Function ToVar (vntSrc As Variant)	ChangeType 関数のマクロ関数で, ChangeType(vntSrc, 12)と

As Variant	同じ意味です.
Function VariantFromBstr (strVal as String) as Variant	RAC 文字列から Variant オブジェクトを生成します. (例) vntVal = dat.VariantFromBstr(“8,TestString”)

表 3-8 EXT オブジェクトメンバー一覧

メンバ	機能
Sub Delay (dwMilliseconds As Long)	指定時間(ms)だけ実行を中断します.
Function GetValueEx (objVar As Variant, Optional defaultVal As Variant, Optional lTimeout As Long = -1) As Variant	リトライ付きの値取得. タイムアウトまたは値(objVar.Value)の読込みが成功するまでリトライします. defaultVal を指定し且つタイムアウトが発生した場合はその値を返します. lTimeout = -1 のとき無限にリトライします. (例) ext.GetValueEx(objVar, "default", 1000)
Sub PutValueEx (objVar As Variant, newVal As Variant, Optional lTimeout As Long = -1)	リトライ付きの値設定. タイムアウトまたは値(objVar.Value)の書込みが成功するまでリトライします. lTimeout = -1 のとき無限にリトライします. (例) ext.PutValueEx objVar, 123, 1000
Sub WaitIo (objVar As Object, Optional bPositive as Boolean = True, Optional lTimeout As Long = -1, Optional bHandShake As Boolean = False)	objVar.Value が 0 以外の値になるか(bPositive = True の時), または 0 になるか(bPositive = False の時), またはタイムアウトが発生するまで待機します. タイムアウトを指定しない場合は無限に待機します. ■ ハンドシェイク機能 bPositive = True & bHandShake = True の時: WaitAndReset 動作 bPositive = False & bHandShake = True の時: SetAndWait 動作
Sub BeepEx (Optional iCount As Integer = 1, Optional lInterval As Long = 0)	ビープ音を鳴らします. 回数と間隔(ms)を指定できます. 省略した場合はそれぞれ 1 回と 0ms になります. （注） 鳴動時間と周波数はハードウェアとシステムソフトウェアに依存するため, コンピュータの機種によって異なります.

表 3-9 CAOMESS オブジェクトメンバー一覧

メンバ	機能
Property Get Index() As Long	CaoController オブジェクトを識別するための Index 番号.
Sub Attach (caoCtrl As Variant)	CaoController の OnMessage イベントを取得できるように

	CaoController オブジェクトを登録します. (備考) CAOMESS オブジェクトは最初に APP オブジェクトの Attach 関数で生成されます. Detach をして明示的に同じオブジェクトを再利用したい場合に使用します.
Sub Detach()	OnMessage のイベントシンクをデタッチしてイベントを受信しないようにします. 再度有効にしたい場合は Attach 関数で再登録してください.

表 3-10 ERROR オブジェクトメンバー一覧

メンバ	機能
Property Get Description () As String	エラー内容を取得します.
Property Get Handler() As String	App.OnErrorGoto で設定された値を取得します.
Property Let Handler (newVal As String)	App.OnErrorGoto と同じ.
Property Get Line () As Long	エラー行番号を取得します.
Property Get ExtLine () As Long	App.AddScript でマージしたプログラム内でエラーが発生した場合のエラー行番号を取得します.
Property Get Number () As Long	エラー番号を取得します.

表 3-11 ChangeType 利用可能型一覧

VT 型	値	型情報
VT_BOOL	11	BOOL 型
VT_I1	16	符号付 1 バイト整数型
VT_UI1	17	符号なし 1 バイト整数型
VT_I2	2	符号付 2 バイト整数型
VT_UI2	18	符号なし 2 バイト整数型
VT_I4	3	符号付 4 バイト整数型
VT_UI4	19	符号なし 4 バイト整数型
VT_R4	4	4 バイト浮動小数点数型
VT_R8	5	8 バイト浮動小数点数型
VT_CY	6	通貨型

VT_DATE	7	日付型
VT_BSTR	8	文字列型
VT_VARIANT	12	VARIANT 型

3.2. 組込みイベント

CaoScript のイベントには表 3-12 の3つのイベントが定義されています.

表 3-12 組込みイベント一覧

メンバ	機能
Sub App_BeforeExit (lExitMode as Long)	実行終了イベント. この関数を実装すると, プログラムの実行終了時に呼び出されます. [lExitMode] 0: 正常終了 1: 停止による終了 2: 異常による終了
Sub App_OnMessage (lIndex as Long, caoMess as Variant)	CaoMessage イベント. この関数を実装すると, CaoController クラスの OnMessage イベントを受信することができます. どの CaoController オブジェクトのイベントを受信するか予め APP オブジェクトの Attach メソッドで登録する必要があります. [lIndex] CaoController オブジェクトを識別するための Index 番号を返します. Attach 実行で返された CAOMESS オブジェクトの Index プロパティと照合することでどのオブジェクトのイベントなのかを判別することができます. [objCaoMess] 受信した CaoMessage オブジェクトを返します. (例) Dim bb Dim mess Sub Main Set bb = cao.AddController(“BB”,…) Set mess = app.Attach(bb) … End Sub ‘ OnMessage イベントハンドラ

	Sub App_OnMessage(lIndex, caoMess) 　dbg.output caoMess.Value End Sub
Sub App_OnTimer()	タイマイベント. この関数を実装すると, 設定された間隔毎に呼び出されます. 呼び出し間隔は APP オブジェクトの TimerInterval プロパティで設定します. また, タイマ機能をオフにする場合は APP オブジェクトの Clear メソッドを実行します.

3.3. エラー処理

CaoScritp 言語は VBScript 言語をベースにしていますので, VBScript の On Error ステートメントは On Error Resume Next(エラーを無視する)を使うことができます. しかし, CaoScript で On Error Resume Next を使った場合, エラーが発生しても無視して停止しないだけではなくて, 停止ボタン(Automation IF の ScriptStop メソッドも同様)でも下記の場合のどちらかまで到達しないと停止しませんので注意してください.

1. Do, For, While ステートメント
2. End ステートメント

このように VBScript の On Error ステートメントでは柔軟なエラー処理を記述することができませんので, CaoScritp では独自に On Error ステートメントを拡張しています. その場合, エラーハンドリングに App オブジェクトの OnErrorGoto メソッドを使います. このメソッドでエラー発生時にジャンプするサブルーチンを設定します. ここで注意しなくてはならないのは, サブルーチンはコールされるわけではなく, サブルーチンにジャンプ(Goto)するという点です. ジャンプですので制御はそのサブルーチンに渡り, サブルーチンを抜けてもエラーが発生した行へ復帰(Resume)することはできません. また, そのサブルーチンの中で再度エラーが発生してもそのエラーはハンドリングされません. つまり, 一度ジャンプすると設定は自動的に解除されます. もし再度ハンドリングしたい場合は, そのサブルーチンの中で再度 OnErrorGoto メソッドを設定します. その場合は, 無限ループに陥らないように注意してください.

OnErrorGoto でトラップされた場合, VBScript の Err オブジェクトはすでにクリアされていますので, 代わりに App オブジェクトの Error オブジェクトを使ってエラー情報を取得することができます.

また, エラーが発生してジャンプしたあとにも変数を共有したい場合は関数の外(ファイルスコープ)で変数を宣言します. ファイルスコープの変数は関数間で共有することができますのでエラージャンプの前に値を設定してジャンプ後に参照することができます.

OnErrorGoto の設定は一つだけですので, 最後に設定した値だけが使用されます. 空文字("")を設定するとエラーハンドリングは解除されます. また, 同時に引数を渡すこともできます. 下記に使用例を示します.

```
' ------------------------------
Sub Main
        ...
        app.OnErrorGoto "ErrHandle"          ' 以降のエラーは ErrHandle へジャンプ
        ...
        app.OnErrorGoto "ErrHandleEx 123"    ' 以降のエラーは ErrHandleEx(123)へジャンプ
```

```
        ...
        app.OnErrorGoto ""                    ' 以降のエラーはハンドリングされない.
        ...
End Sub

Sub ErrHandle
        Msgbox "Error Handler"
End Sub

Sub ErrHandleEx(Arg1)
        Msgbox Arg1
End Sub
' ------------------------------
```

3.4. VARS 変数の種類

VARS 変数はバックアップしたいデータを保存したり，アプリケーション間でデータを共有したりする場合に使います．その領域によって機能やアクセス速度が違うので目的に応じて使ってください．例えば，オペレーションパネル機能を使っている場合，オペレーションパネルは別プロセスで動作していますのでローカルメモリ領域を使うことはできません．

Vars 変数	タスク間データ共有	アプリ間データ共有	バックアップ	速度	主な用途
データベース	可能	可能	可能	低速	・アプリ終了後もデータを保存したい場合に使う．例えば，設定パラメータ等．
メモリ	可能	可能	不可	中速	・他のアプリケーション(例:Operation Panel)とデータを共有(プロセス間共有)したい場合で，且つデータをバックアップする必要がない場合に使う．
ローカルメモリ	可能	不可	不可	高速	・タスク間でデータを共有すればよい場合に使う．

図 3-2 VARS 変数の種類

```
【測定条件】Pentium 4, 1.6GHz, 1GBメモリ

【測定プログラム】
  For i=1 to 10000
     Vars(1001).Value = i     ' DB=1, Mem=501, LMem=1001
     Vars(1002).Value = i     ' DB=2, Mem=502, LMem=1002
  Next
```

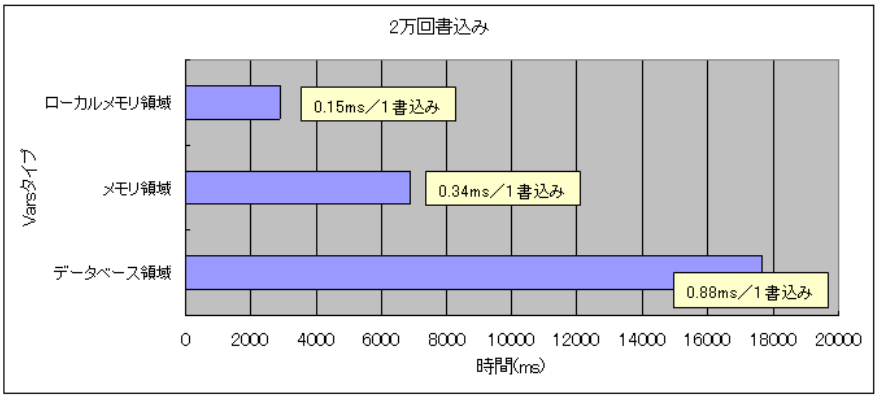


図 3-3 VARS 変数のアクセス速度比較

VARS 変数は DataStore プロバイダの@VARS 変数を使用しているので，その領域サイズは DataStore プロバイダの Parameter で定義します．DataStore プロバイダの Parameter は CaoConfig ツールで設定して下さい．

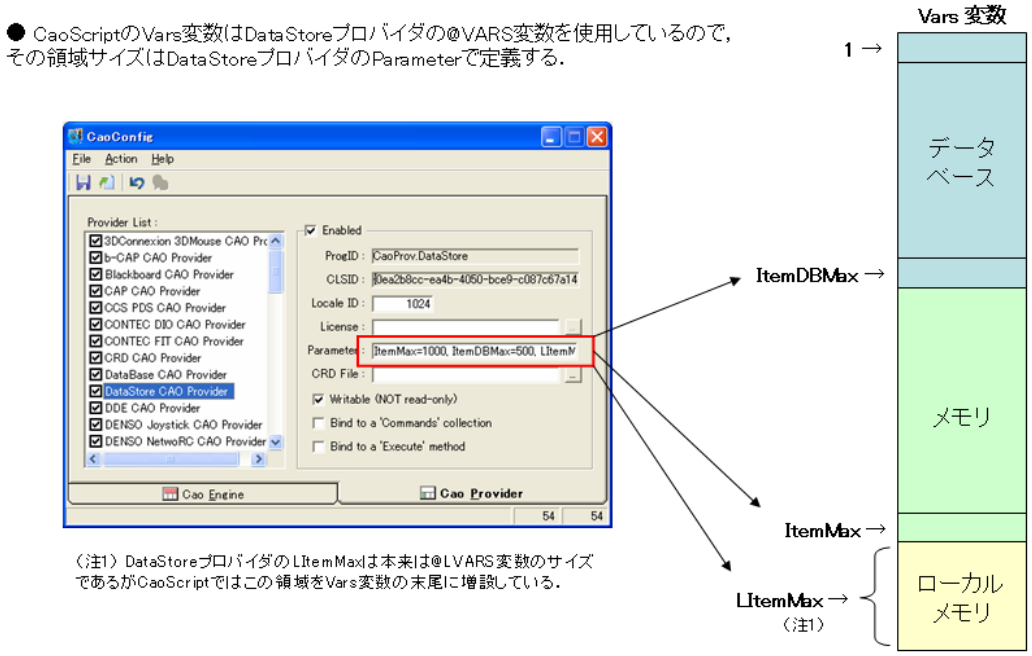


図 3-4 VARS 変数の領域サイズ

4. Automation インターフェース

CaoScript は Automation インターフェースを使用することで外部プログラムから制御することが可能です.

4.1. プログラムサンプル

Automation インターフェースを使った例を下記に示します. このサンプルプログラムは Visual Basic 6.0 で書かれおり, これを実行する前に参照設定で"CaoScript"を有効にする必要があります. また, さらに詳細なサンプルについては下記のフォルダにある Tester のソースを参考にしてください.

<ORiN2>/CAO/Tools/CaoScript/Src.Tester

(例) VB6 の場合

```
Set scr = New CaoScript.clsEngine          ' CaoScript エンジン生成
With scr
  .ScriptText = "Msgbox ""Hello!"""        ' スクリプトを設定
  .Startup = ""                            ' エントリ関数を設定
  .StartScript 1                           ' スクリプトを実行
End With
```

4.2. .NET クライアント

.NET 言語(例:VB2005)を使って CaoScript を同様に制御する場合は, 参照設定で COM として設定するのではなく, 下記の RCW (Runtime Callable Wrapper)を参照設定してください[1]. こうすることでクライアント毎に CaoScript の RCW が作成されるのを防げます. 公開される機能は表 4-1 を参照してください.

<ORiN2>/CAO/Tools/CaoScript/Bin/CaoScrRCW.dll

同様に CaoScript Manager の RCW は下記のファイルです. 公開される機能は共有オブジェクト Mgr(表 6-4)を参照してください.

<ORiN2>/CAO/Tools/CaoScript/Bin/CaoScrManRCW.dll

4.3. Automation インターフェース仕様

Automation インターフェースのメンバは以下の通りです.

表 4-1 Automation インターフェース メンバー覧

メンバ名	機能
Property Get ExitCode () As Variant	App.ExitCode で設定された終了コードを取得します.
Property Get FileName () As String	現在開いているファイル名を取得します.
Property Get FileNameChanged () As Boolean	ファイル名の変更フラグを取得します.
Property Let FileNameChanged (bChanged As	ファイル名の変更フラグを設定します.

[1] このモジュールは ORiN2 SDK 2.0.7 以降で使えます.

Boolean)	
Property Let GUIEnabled (strObjPath As String, bEnabled As Boolean)	指定した GUI オブジェクトの Enable を設定します.
Property Get GUIEnabled (strObjPath As String) As Boolean	指定した GUI オブジェクトの Enable を取得します.
Property Get hWnd () As Long	ウィンドウハンドルを取得します.
Property Get LastLog () as String	最後にログウインドウに表示されたログを取得します.
Property Get LineNumber () As Long	現在実行停止行を取得します.
Property Get LogText () as String	ログウインドウ全体を取得します.
Property Get Modified () As Boolean	スクリプトテキストの編集状態を取得します.
Property Let Modified (bModified As Boolean)	スクリプトテキストの編集状態を設定します.
Property Get ReadOnly () As Boolean	スクリプトテキストの編集可否状態を取得します.
Property Let ReadOnly (bReadOnly As Boolean)	スクリプトテキストの編集可否状態を設定します.
Property Get ScriptState () As Long	CaoScript の状態を取得します. 取得する数値は以下の通りです. -1:初期化中 0:スクリプト非実行状態 1:スクリプト実行中 2:スクリプトエラー発生
Property Get ScriptText () As String	スクリプトテキストを取得します.
Property Let ScriptText (strText As String)	スクリプトテキストを設定します.
Property Get Startup () As String	実行関数名を取得します.
Property Let Startup (strText As String)	実行関数名を設定します.
Property Get Tag () As Variant	タグ情報を取得します.
Property Let Tag (varTag As Variant)	タグ情報を設定します.
Property Get Visible () As Boolean	CaoScript の可視/不可視状態を取得します.
Property Let Visible (bVisible As Boolean)	CaoScript の可視/不可視状態を設定します.
Sub AddObject (strName As String, objAddin As Object)	スクリプトにユーザ定義のオブジェクトを登録します. これにより CaoScript の言語機能を自由に拡張することができます. strName の大文字・小文字は区別されません.
Function CallScript (strText As String) As Variant	関数の同期呼び出し. ExitCode の値が結果として返されます. (例1) Ret = scr.CallScript("Sub1(100) ") ' 引数 整数 100

	(例2) Ret = scr.CallScript("Sub2(""" & text & """)") ’ 引数に文字列指定
Sub ClearLog ()	ログウインドウをクリアします.
Sub Execute (strCmd As String, Optional vntParam As Variant)	拡張機能の実行. 表 4-3 参照.
Sub Hide (Optional vntOpt As Variant)	CaoScript を非表示にします.
Sub Initialize (Optional vntOpt As Variant)	初期化処理を実行します. (備考)現在使われておりません.
Sub OpenFile (strFilename As String, Optional vntOpt As Variant)	指定したパスのファイルを開きます.
Sub RemoveObject (strName As String)	AddObject で追加したオブジェクトを開放します. strName の大文字・小文字は区別されません. (備考)明示的に開放しなくても CaoScript エンジンの開放で同時に開放されます.
Sub SaveFile (strFilename As String, Optional vntOpt As Variant)	現在の内容を指定ファイルパスに保存します.
Sub Show (Optional vntOpt As Variant)	CaoScript を表示します.
Sub StartScript (lMode As Long, Optional vntOpt As Variant)	スクリプトを非同期実行します. [2] モードに指定する数値は以下の通りです. 1: 1 サイクル実行 2: 連続実行 3: 1 ステップ送り 4: 1 ステップ戻し 5: 高速実行 6: 1 ステップ送り(ウィンドウ表示レス)
Sub StopScript (lMode As Long, Optional vntOpt As Variant)	スクリプトを非同期停止します. [3] モードに指定する数値は以下の通りです. 0: デフォルト停止(1 と同じ) 1: 瞬時停止 2: ステップ停止 3: サイクル停止 4: 初期化停止(1 と同じ) 5: ステップ停止(ウィンドウ表示レス)

[2] 4(1 ステップ戻し)は未実装です.
[3] スクリプト停止の指示だけを行い, この関数自体は非同期で終了します. もし, 確実に停止したことを確認したい場合は State プロパティで状態を確認して下さい.

Sub Terminate (Optional vntOpt As Variant)	終了前処理を実行します.

※ GUIEnabled プロパティの GUI オブジェクト名は以下のフォーマットで指定します.

<親オブジェクト名>[.<子オブジェクト名>[.<孫オブジェクト名>]]

親オブジェクト名にはメニュー, もしくはツールバーを指定できます. メニューを指定する場合は"Menu", ツールバーを指定する場合は"Tlbar"と指定します. 子オブジェクト名, 孫オブジェクト名については表 4-2 を参照して下さい.

表 4-2 GUIEnabled で利用可能なオブジェクト名一覧

親オブジェクト名	子オブジェクト名	孫オブジェクト名
menu[4]	file	new
		open
		save
		save_as
		save_log_as
		exit
	edit	undo
		cut
		copy
		paste
		delete
		select_all
		find
		find_next
		replace
		goto
		break_point
		line_trace
		clear_log
	form	font
	run	start
		step_over
		pause

[4] トップメニューの Enabled 設定/取得はできません.

		stop
		fast_start
		cycle_start
		cycle_stop
	help	version
tlbar	open	
	save	
	save_log	
	cut	
	copy	
	paste	
	find	
	comment	
	del_comment	
	startup	
	start	
	step_over	
	pause	
	stop	

例えば, メニューの[File]の[Save As…]を指定する場合, GUI オブジェクト名は“menu.file.save_as”となります. ツールバーの[Cut]を指定する場合, GUI オブジェクト名は“tlbar.cut”となります.

表 4-3 Execute で利用可能なコマンド一覧

コマンド名	引数	説明
GetScriptStarted	なし	StartScript で非同期実行した場合, 実際にスクリプトが開始していれば True を返します.

5. コマンドラインオプション

CaoScript.exe を起動する際に下記のオプションを指定することができます.

CaoScript.exe <ファイル名> [<関数名> [start]]

<ファイル名>: Open するファイル名を指定します.

<関数名>: スタートアップ関数に設定する関数名を指定します.

<start>: 自動的に実行します.

(例) > CaoScript.exe d:¥temp¥test.vbs PRO1 start

6. CaoScriptManager

6.1. 概要

CaoScript Manager は複数の CaoScript を一括管理できるツールです.

1 つの CaoScript を 1 行としてグリッドに表示し, 設定の変更, スクリプトの実行を行うことができます.

CaoScript プロジェクトファイル(拡張子:csp)に画面の情報を保存することができ, 管理環境を保存/再現することができます.

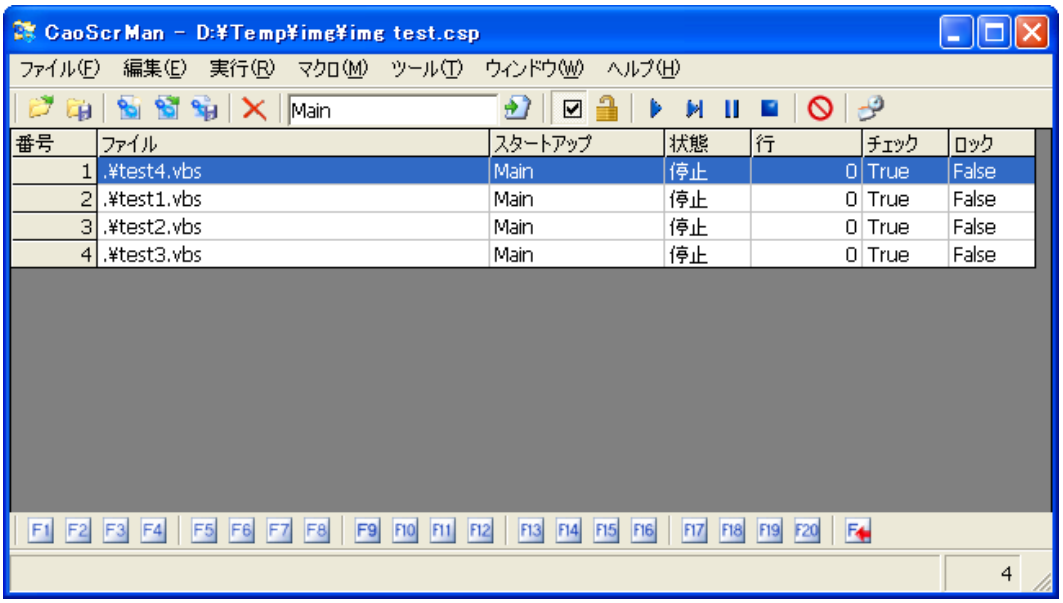


図 6-1 CaoScriptManager 起動画面

6.2. 画面説明

6.2.1. CaoScript 表示グリッド

番号	ファイル	スタートアップ	状態	行	チェック	ロック
1	.¥test4.vbs	Main	停止	0	True	False
2	.¥test1.vbs	Main	停止	0	True	False

図 6-2 CaoScript 表示グリッド

各列については表 6-1 を参照して下さい.

表 6-1 CaoScript 表示グリッド 各列説明

列名	意味	対応 CaoScript プロパティ名
番号	行番号. 複数行選択実行の場合, Index 番号の若い行から順に実行されます.	(なし)
ファイル	CaoScript にて開いているファイル名.	FileName
スタートアップ	スタートアップ関数名.	Startup
状態	CaoScript のステータス値.	ScriptState
行	CaoScript の現在実行行.	LineNumber
チェック	詳しくは実行メニュー (6.2.4), ウインドウメニュー (6.2.7)の Target を参照して下さい.	(なし)
ロック	スクリプト入力ウインドウの編集可/不可設定.	ReadOnly

6.2.2. ファイルメニュー

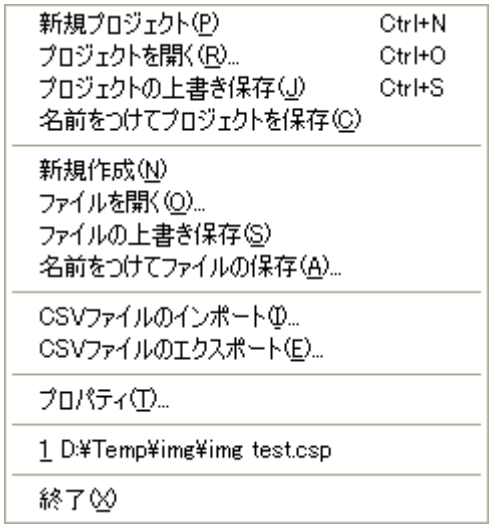


図 6-3 ファイルメニュー

[新規プロジェクト]

プロジェクトを新規に作成します.

[プロジェクトを開く…]

CaoScript プロジェクトファイルを開きます.

[プロジェクトの上書き保存]

現在開いているプロジェクトを上書きします.

[名前をつけてプロジェクトを保存…]

現在開いているプロジェクトに名前をつけて保存します.

[新規作成]

ファイル名なしの CaoScript を追加します.

[ファイルを開く…]

指定したファイルを開いた CaoScript を追加します.

[ファイルの上書き保存]

現在選択行の CaoScript を保存します.

[名前をつけてファイルの保存…]

現在選択行の CaoScript に名前をつけて保存します.

[CSV ファイルのインポート…]

CSV ファイルの内容を DataSheet に読み込みます.

[CSV ファイルのエクスポート…]

DataSheet の内容を CSV ファイルに保存します.

[プロパティ…]

プロジェクトプロパティウインドウ(6.2.9)を開きます.

6.2.3. エディットメニュー

エディットメニューの動作は，選択されている行全てに適用されます．

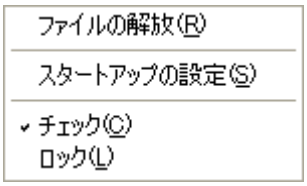


図 6-4 エディットメニュー

[ファイルの解放]

現在選択行を削除します．

[スタートアップの設定]

現在選択行の CaoScript にツールバーのテキストボックスに設定されたスタートアップ関数名を設定します．

図 6-5 スタートアップ関数名設定ボックス

[チェック]（初期値”True”）

複数のファイルを対象としたい場合，チェックを”True”にします．図はチェック状態で，もう一度選択することで”False”となります． 例えば，「実行メニュー」の「実行ターゲット」→「チェック」を選択した場合には，このチェックが”True”の設定のファイルを全て実行します．

[ロック]

ファイル操作を禁止したい場合などに使用します．このロックが”True”の場合，編集，実行などが行えなくなります．

6.2.4. 実行メニュー

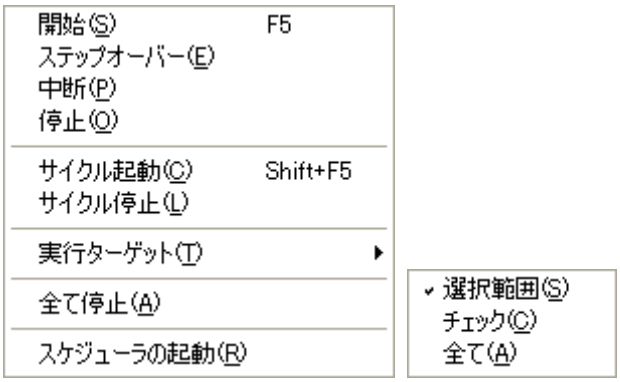


図 6-6 実行メニュー

[開始]

実行ターゲットで対象の CaoScript の Start メソッドを実行します.

[ステップオーバー]

実行ターゲットで対象の CaoScript の Step Over メソッドを実行します.

[中断]

実行ターゲットで対象の CaoScript の Pause メソッドを実行します.

[停止]

実行ターゲットで対象の CaoScript の Stop メソッドを実行します.

[サイクル起動]

実行ターゲットで対象の CaoScript の Cycle Start メソッドを実行します.

[サイクル停止]

実行ターゲットで対象の CaoScript の Cycle Stop メソッドを実行します.

[実行ターゲット]

実行対象行条件を選択します.

[選択]:現在選択行

[チェック]:Checked 列が True の行

[全て]:全ての行

[全て停止]

全ての Scritpt を停止します.

[スケジューラの起動]

スケジューラタスクを実行します.

6.2.5. マクロメニュー

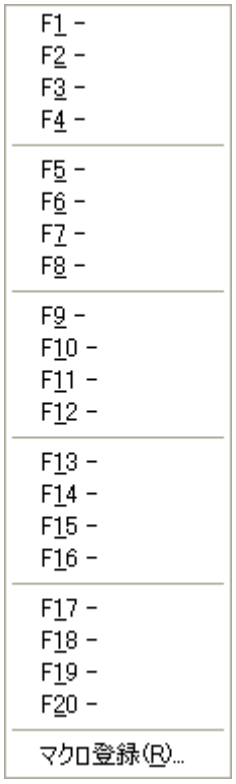


図 6-7 マクロメニュー

[F1-20]

登録されているマクロを実行します.

[マクロ登録…]

マクロウインドウ(6.2.10)を開きます.

6.2.6. ツールメニュー

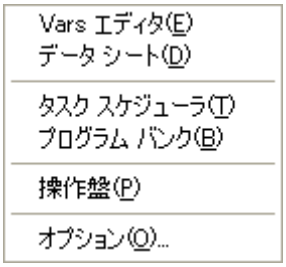


図 6-8 ツールメニュー

[Vars エディタ]

Vars エディタウインドウ(6.2.11)を開きます.

[データ シート]

データシートウインドウ(6.2.12)を開きます.

[タスク スケジューラ]

タスク スケジューラウインドウ(6.2.14)を開きます.

[プログラム バンク]

プログラムバンクウィンドウを開きます.

[操作盤]

Vars Panel ウィンドウを開きます.

[オプション…]

オプションウインドウ(6.2.16)を開きます.

6.2.7. ウインドウメニュー

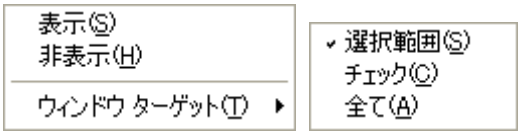


図 6-9 ウインドウメニュー

[表示]

ウィンドウターゲットで対象の CaoScript の Show メソッドを実行します.

[非表示]

ウィンドウターゲットで対象の CaoScript の Hide メソッドを実行します.

[ウィンドウ ターゲット]

実行対象行条件を選択します.

[選択]:現在選択行

[チェック]:Checked 列が True の行

[全て]:全ての行

6.2.8. ヘルプメニュー

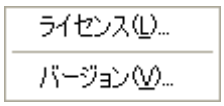


図 6-10 ヘルプメニュー

[ライセンス…]

ライセンス設定画面を表示します.

[バージョン…]

バージョン情報を表示します.

6.2.9. プロジェクトプロパティウィンドウ

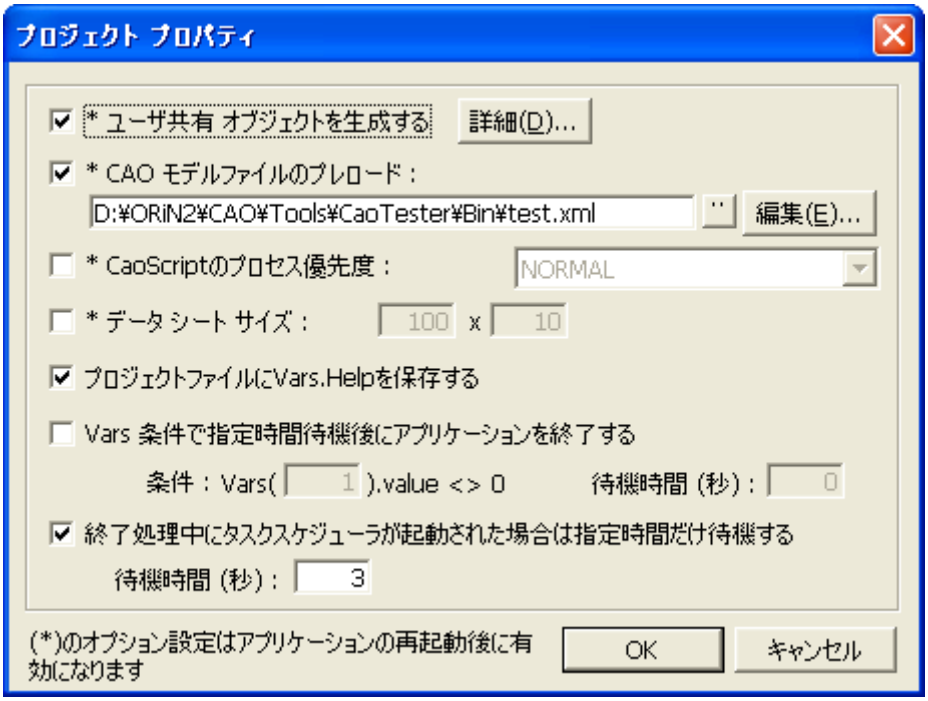


図 6-11 プロジェクトプロパティ画面

[ユーザ共有オブジェクトを生成する]

ユーザ共有オブジェクト生成の有無を設定します. チェックボックスを ON にした場合, CaoScript にユーザがあらかじめ設定している共有オブジェクトが追加定義されます. 設定方法は 6.2.9.1 を参照します.

[CAO モデルファイルのプレロード]

共有オブジェクト“Cao”が起動時に生成するオブジェクト構造を CRD ファイルで指定します. ま

た，”編集”ボタンを押すことでCRDファイルが関連付けられているEditorで編集することも可能です．編集中はプロパティ設定を終えることができません．

[CaoScript のプロセス優先度]

CaoScript プロセスの優先度を IDLE，BELLOW NORMAL，NORMAL，ABOVE NORMAL，REAL TIME 設定します．

[データシートサイズ]

データシートの横×縦のサイズを設定します．

[プロジェクトファイルに Vars.Help を保存する]

Vars オブジェクトの Help プロパティをプロジェクトファイルに保存するかどうかを設定します．

[Vars 条件で指定時間待機後にアプリケーションを終了する]

CaoScriptMan の終了条件を設定します．チェックボックスを ON にした場合，指定した Vars 変数の値が 0 以外の値が指定時間（単位：秒）続いた場合に CaoScrMan が終了します．

（注）新規にこの設定を行ったときは，指定した Vars 変数の値は 0 に初期化されます．

[終了処理中にタスクスケジューラが起動された場合は制定時間だけ待機する]

シャットダウン時にタスクスケジューラによりスクリプトが起動した場合に指定時間だけ待機します．

(*)の記号が付いている設定を有効にするためには，プロジェクトファイルを保存した後に CaoScriptManager を再起動しなければなりません．

6.2.9.1. ユーザ共有オブジェクト

CaoScript にユーザが作成した共有オブジェクトを追加することができます．この機能を使用することで，CaoScript でオブジェクトを生成する際に，CreateObject をせずにオブジェクトを使用することができます．

また，生成したオブジェクトは複数のスクリプトファイル間で共有することも可能です．以下に設定方法を説明します．

まず，ユーザ共有オブジェクトの設定を行うには，図 6-11 のプロパティウィンドウから「ユーザ共有オブジェクトを生成する」をチェックします．次に項目の横にある「詳細」ボタンを押します．

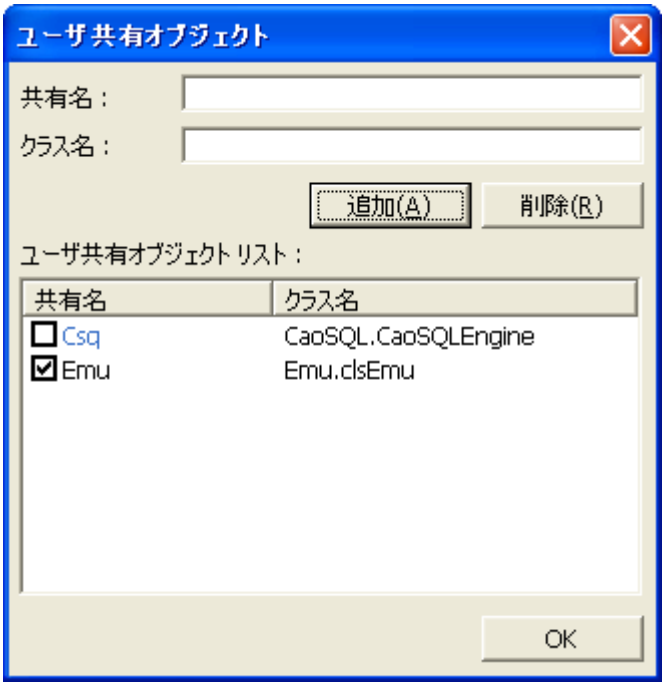


図 6-12 ユーザ共有オブジェクト

図のような設定ダイアログが表示されます．それぞれの項目を設定し，「追加」ボタンを押します．
追加するとオブジェクト名がリストの最後に追加されます．必要なオブジェクトのみチェックをします．
逆に削除したい場合は，リストから対象のオブジェクト名を選択して，「削除」ボタンを押します．

図 6-12 のように青文字で追加されている共有名(現在は"Csq"のみ)は，CaoScrMan デフォルトの設定なので削除できません． また，同じ名前の共有名を追加しようとすると警告が出ます．

以下，設定ダイアログの項目を説明します．

[共有名]

共有名を設定します．この設定した名前がそのまま CaoScript 上での共有オブジェクト名として使用されます．

[クラス名]

共有オブジェクトを生成する時に必要となるクラス名を設定します（CreateObject の引数として指定する名前です）．

ユーザ共有オブジェクトを作成するには，xxx を参照してください．

6.2.10. マクロウィンドウ

マクロの登録を行っておくと，CaoScript のある関数を直接呼び出すことができます．

例えば，外部アプリケーションを起動する CaoScript に記述し，その関数を呼び出すことによって，Add-In のような機能が実現することができます．この詳細は 6.5 マクロ登録による外部アプリケーションのを参照して

ください.

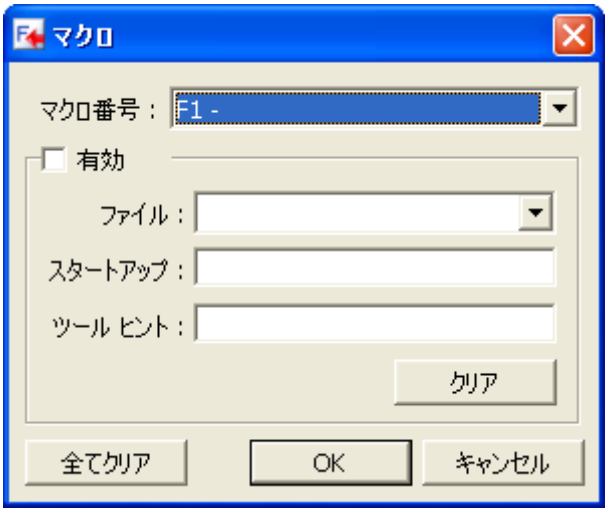


図 6-13 マクロウィンドウ

[マクロ番号]

マクロウィンドウに表示しているマクロ番号

[有効]

指定しているマクロ番号の有効にします.

[ファイル]

マクロが実行するスクリプトファイルを指定します.

[スタートアップ]

マクロが実行するスタートアップ関数を指定します.

[ツールヒント]

マクロのヒントを指定します.

[クリア]

指定しているマクロ番号の設定内容をクリアします.

[全てクリア]

全てのマクロ設定をクリアします.

6.2.11. Vars エディタウインドウ

DataStore プロバイダの@VARS システム変数の値を管理します.

@VARS システム変数については DataStore プロバイダガイドを参照して下さい.

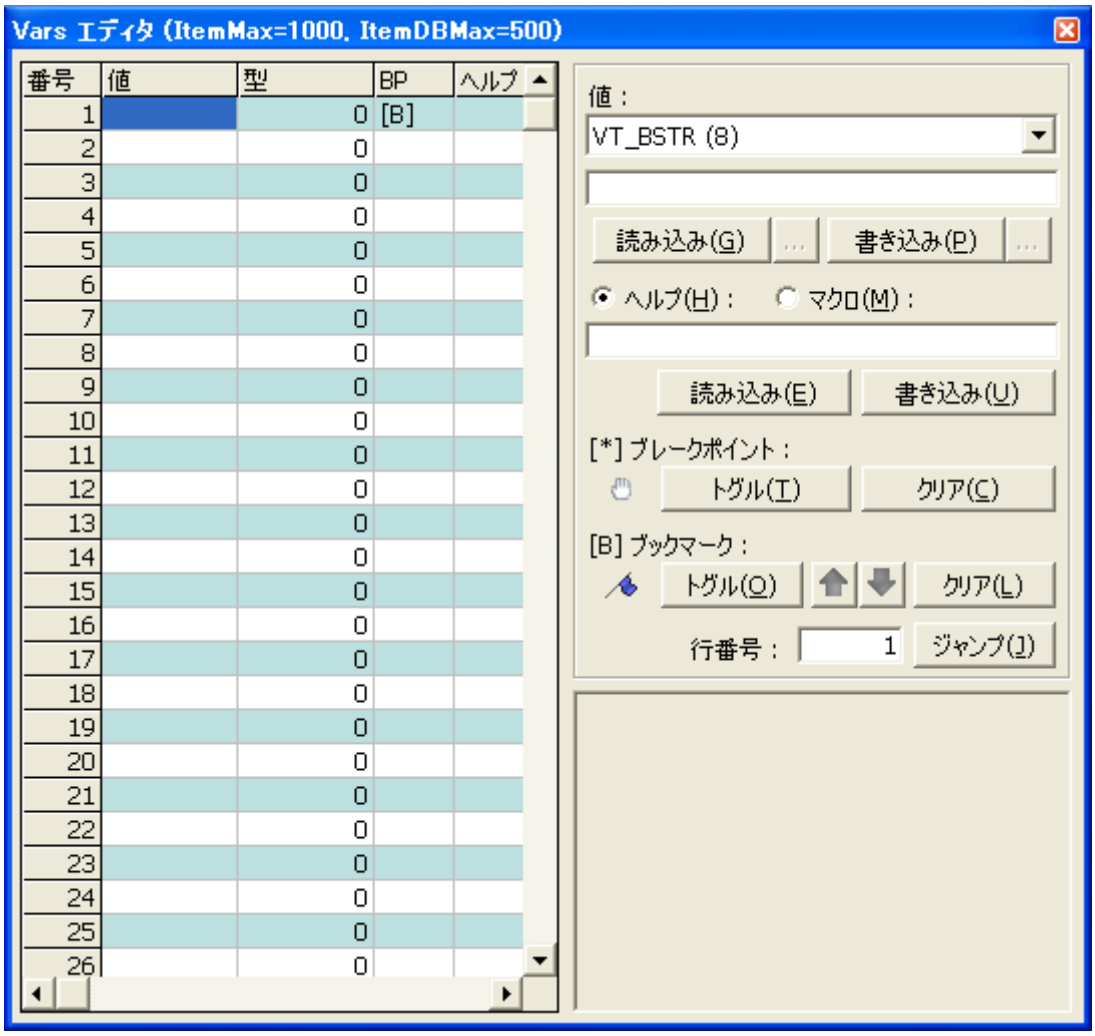


図 6-14 VARS エディタ 画面

[値]-[読込み]

選択されている行の値を取得します. 複数行選択時は初めに選択された行のみ取得します.

[値]- [書き込み]

選択されている行に値を設定します. 複数行選択時は選択されている行全てに設定します.

[ヘルプ]-[読込み]

選択されている行のヘルプを取得します. 複数行選択時は初めに選択された行のみ取得します.

[ヘルプ]-[書き込み]

選択されている行にヘルプを設定します. 複数行選択時は選択されている行全てに設定します.

[マクロ]-[読込み]

選択されている行のマクロを取得します. 複数行選択時は初めに選択された行のみ取得します.

[マクロ]-[書き込み]

選択されている行にマクロを設定します. 複数行選択時は初めに選択された行のみ設定します.

[ブレークポイント]-[トグル]

選択されている行にブレークポイントの設定・解除を行います．複数行選択時は初めに選択された行のみ設定します．

[ブレークポイント]-[クリア]

全てのブレークポイントを解除します．

[ブックマーク]-[トグル]

選択されている行にブックマークの設定・解除を行います．複数行選択時は初めに選択された行のみ設定します．

[ブックマーク]-[クリア]

全てのブックマークを解除します．

[ジャンプ]

指定した行にカーソルを移動します．

6.2.12. Vars イベントハンドラ ウィンドウ

Vars イベントハンドラの設定を行います．

Vars イベントハンドラは，登録した Vars の値が変化したときに，指定したハンドラを実行します。

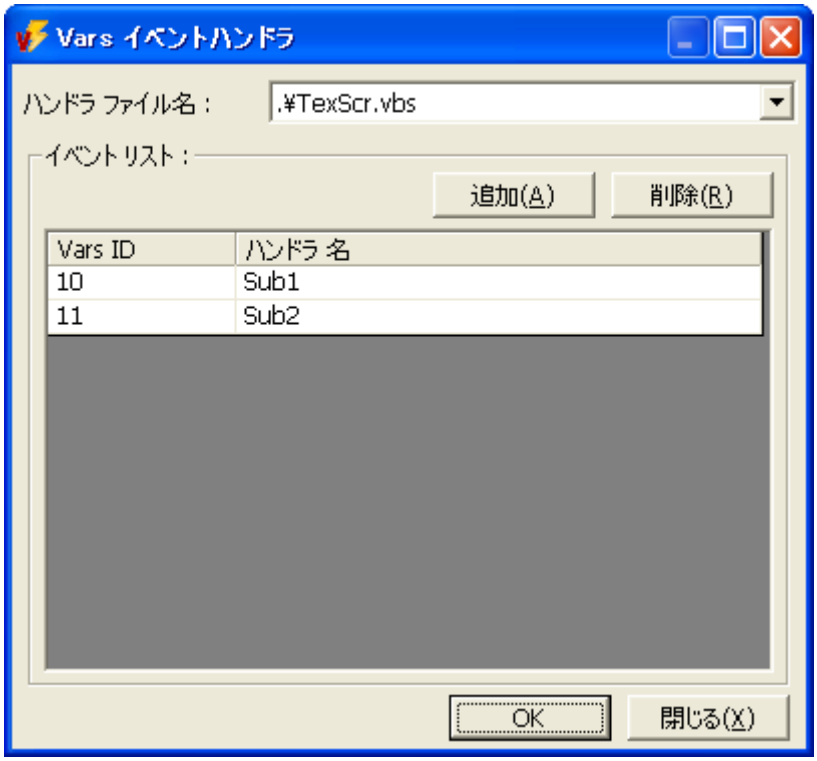


図 6-15 Vars イベントハンドラ ウィンドウ

[ハンドラ ファイル名]

イベントハンドラを記述しているスクリプトファイルを CaoScrMan のスクリプト一覧から選択します．

[追加]

空の Vars イベントを行の一番後ろに追加します.

[削除]

選択されている Vars イベントを削除します.

[イベントリスト]

Vars ID, ハンドラ名を入力します. Vars ID には数値のみを設定します. ハンドラ名は指定したスクリプト内に存在するハンドラを指定してください. 存在しないハンドラが指定されている場合, 登録した Vars が値変化しても, イベントハンドラを呼ぶことができません.

6.2.13. データシートウィンドウ

Csv オブジェクトに格納されているデータの表示及び編集を行います.

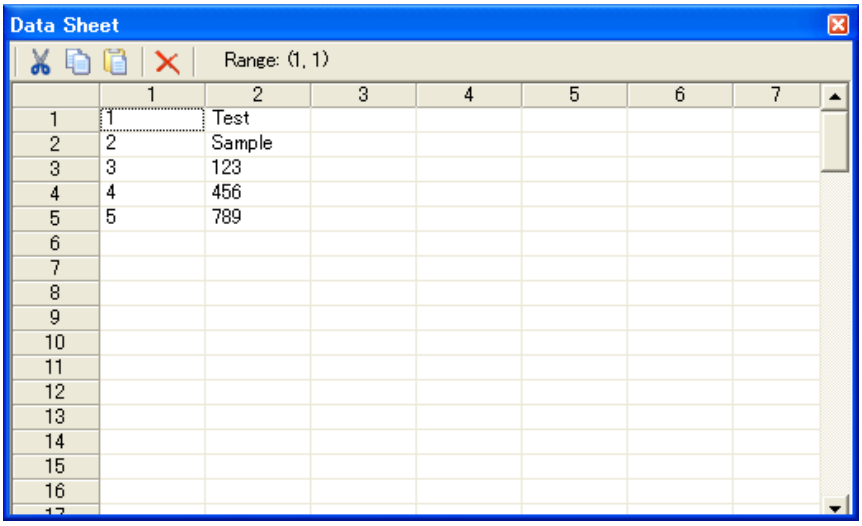


図 6-16 データシート ウィンドウ

6.2.14. タスクスケジューラウインドウ

タスクスケジュールの一覧を表示します. チェックボックスにより, タスクスケジュールの ON/OFF の切り替えを行うことができます.

図 6-17 タスクスケジューラ 画面

[追加]

タスクスケジュールを追加します.

[編集]

選択されているタスクスケジュールを編集します.

[コピー]

選択されているタスクスケジュールをコピーします.

[貼り付け]

コピーされたタスクスケジュールを貼り付けします.

[削除]

選択されているタスクスケジュールを削除します.

タスクスケジュールの追加及び編集時には, 以下の画面で詳細な内容を決定します.

図 6-18 Task Scheduler 編集画面（スケジュール設定）

[ファイル]

タスクスケジュールを実行するファイル名

[スタートアップ]

タスクスケジュールを実行するスタートアップ関数

[スケジュールタスク]

タスクスケジュールの実行種類

“1 回のみ”：指定時間に実行

“アプリケーション起動時”：アプリケーション開始時に実行

“日単位”：1 日単位で実行

“アプリケーション終了時”：アプリケーション終了時に実行

[開始時刻]

タスクスケジュールの開始時刻

[開始日]

タスクスケジュールの開始日付

[終了日]

タスクスケジュールの終了日付

[間隔]

“Daily”指定時の実行日数間隔

[繰返し]

タスクの繰り返し実行指定

[間隔]

タスクの繰り返し実行時の実行間隔

[終了時間]

タスクの繰り返し実行終了時間

[時刻]:時刻指定. 開始時刻よりも前の時刻を指定したときは, 翌日の指定時刻を終了時間とします.

[継続時間]:開始時刻からの経過時間指定

6.2.15. プログラムバンクウィンドウ

プログラムバンクは関数集として使用します. 使用頻度が高い関数をユーザ自身が追加しておけば, いつでもこのプログラムバンクからコピーして, そのまま vbs ファイルへ追加することが可能です.

下記の図では,「Test」という名前のカテゴリ,「Pro1」という名前のアイテムを追加した様子です.

また, 検索ボックスにキーワードを入力して検索を行うことも可能です.

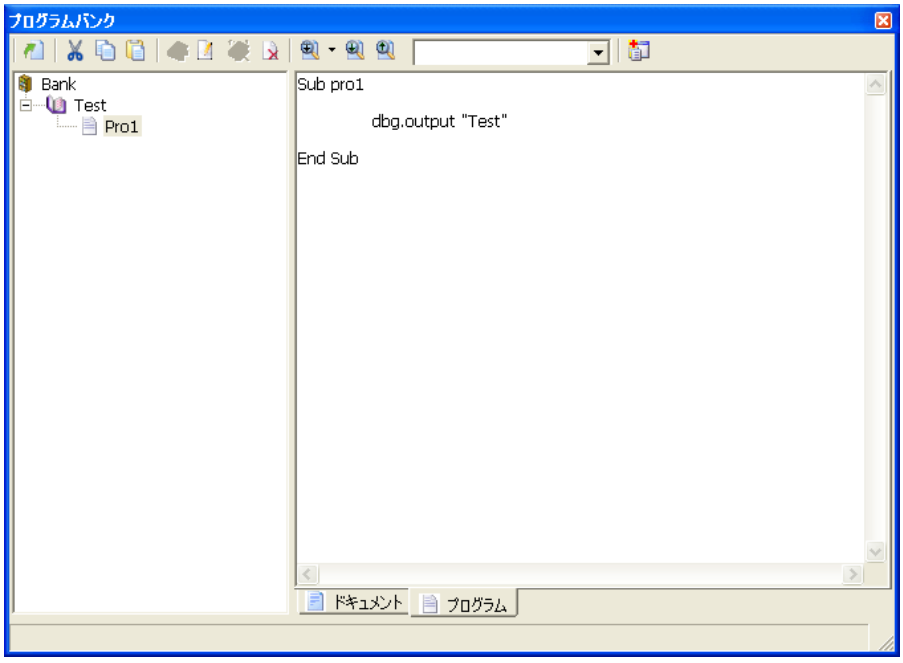


図 6-19 プログラムバンク ウィンドウ

6.2.16. オプションウインドウ

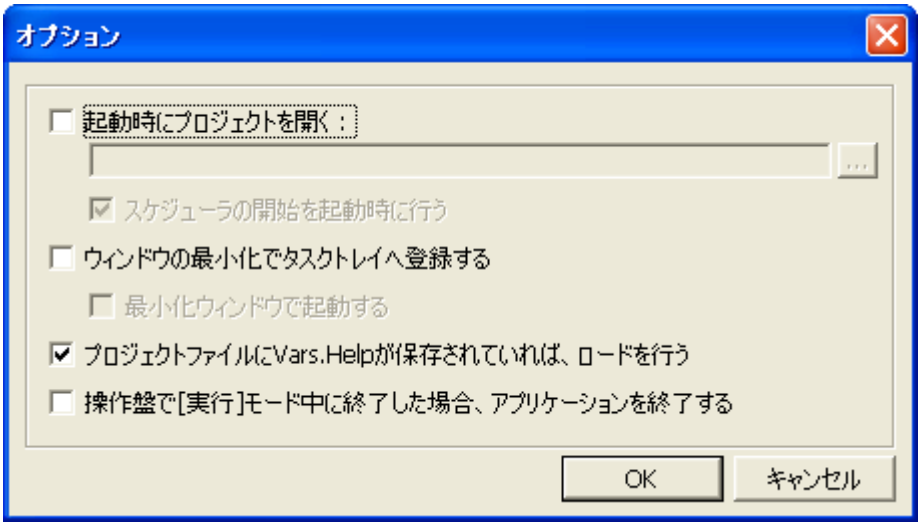


図 6-20 オプション画面

[起動時にプロジェクトを開く]

CaoScriptManager の起動時に指定したプロジェクトファイルを開きます.

[スケジューラの開始を起動時に行う]:CaoScriptManager 起動時にタスクスケジューラを実行します.

[ウィンドウの最小化でタスクトレイへ登録する]

CaoScriptManeger を最小化したときにタスクトレイに追加します.

[最小化ウィンドウで起動する]:CaoScriptManager の起動時に最小化します.

[プロジェクトファイルに Vars.Help が保存されていれば, ロードを行う]

ローカルに保存されている Vars オブジェクトの Help プロパティをロードします.

[操作盤で[実行]モード中に終了した場合, アプリケーションを終了する]

Operation Panel ウィンドウを実行中に閉じたときは, CaoScriptManager を終了します.

6.3. Vars イベント

Vars イベントハンドラ機能はハンドラを記述しているファイルを指定し, 監視する Vars を設定することで任意のハンドラを呼び出すことが可能です.

Vars イベントハンドラに設定した CaoScript は Startup が自動的に「Dispatcher__」となります. イベントハンドラに設定している間は, Startup を変更することはできません.

設定が終わったら, イベントハンドラに設定した CaoScript を実行します.

以下に, 例として Vars の 10 番が"123"になった時, "TextScr.vbs"に記述した「Sub1」というハンドラを呼び出す例の設定手順を示します.

最初に, CaoScrMan の「ツール」メニューから「Vars イベントハンドラ」を呼び出し, 設定を行います.

Vars イベントハンドラの設定では, まずイベントハンドラとなるファイルを選択します. 次にイベントリストにイ

ベント項目を追加します. [追加]ボタンを押すと, 空の条件が追加されるので Vars ID とハンドラ名を入力していきます.

まず, イベントとして登録する Vars の ID を指定します. ハンドラ名は, イベントハンドラとして指定したファイル内に存在するハンドラ名を記述します. ここで存在しないハンドラ名を指定すると, Vars の値が変化してもハンドラが正しく呼び出されません.

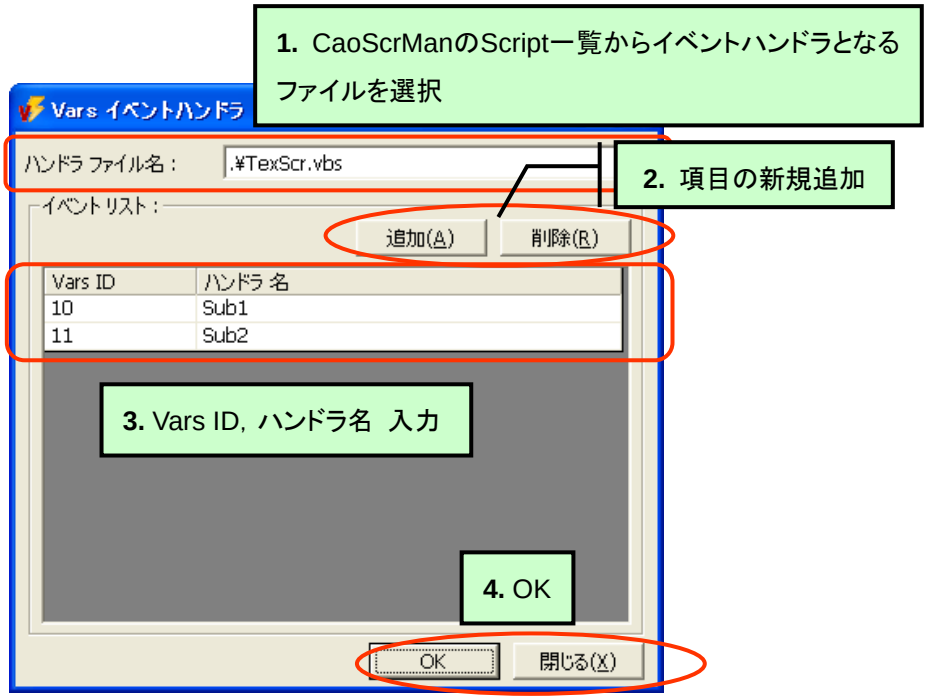


図 6-21 Vars イベントハンドラの設定

次に, CaoScrMan のメインウィンドウからイベントハンドラに設定した CaoScript を実行します.
さきほど設定した Vars 条件が成立するように, Vars エディタで Vars の 10 番を「123」に書込みします.
条件が成立すれば, "TextScr.vbs"の"Sub1"が実行され, メッセージボックスが表示されるのを確認します.

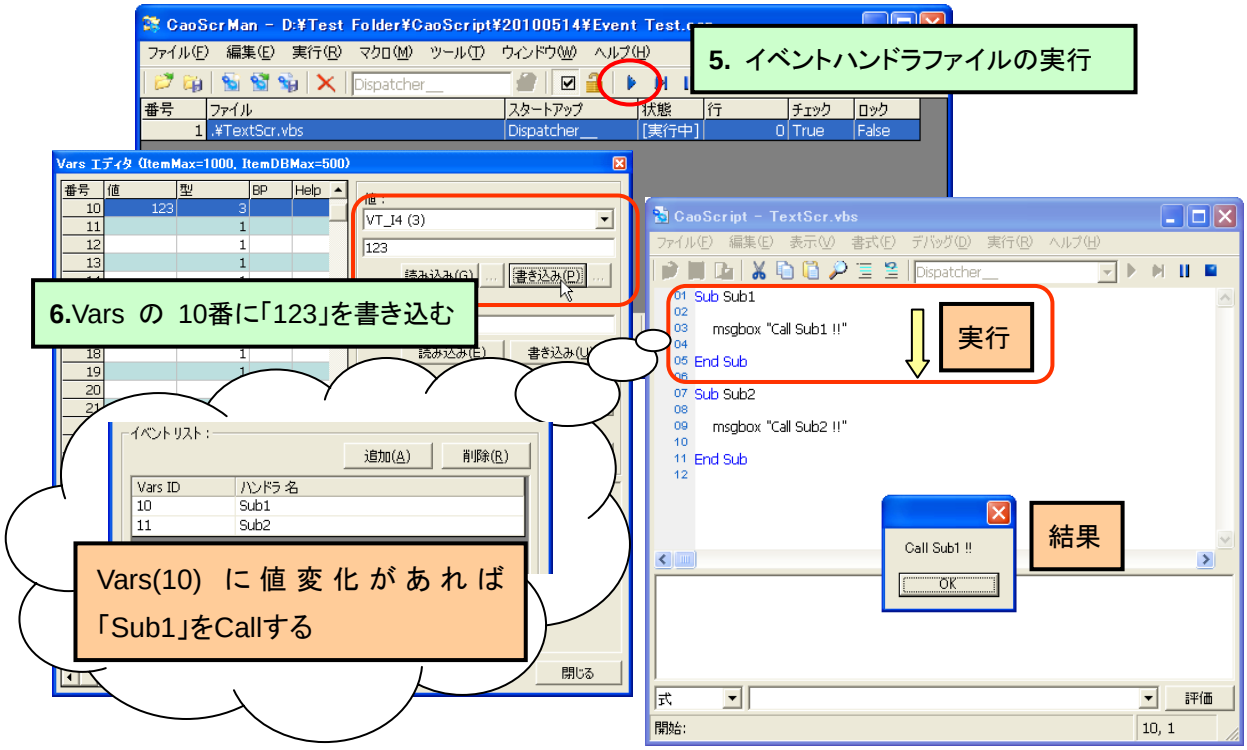


図 6-22 Vars イベント 条件の成立結果

ただし，Vars イベントは値の変化でハンドラを呼び出すので特定の値に変化した時に処理を実行したい場合，下記のようにハンドラの中に条件を記述することで実現できます．

例．Vars の値が 123 になったら処理を実行したいとした場合

```
Sub Sub1

    If Vars(10).Value = 123 Then
            Msgbox “Call Sub1 !!”
    End If

End Sub
```

【要注意】

Vars イベントは値の変化を Vars オブジェクトの Changed プロパティを使用しています．
イベントを設定すると， Dispatcher__にイベント用のコードが追加されます．

```
If Vars(x).Changed Then ハンドラ :Vars(x).Changed = False
```

Vars の値の変化を Changed プロパティで確認し，Changed プロパティをクリアしています．
したがって，ハンドラの中で Changed プロパティを扱う場合などは注意が必要です．

また，イベントハンドラを記述する Script ファイル(Dispatcher)が停止している間に，Vars イベントに登録している Vars を変更した場合，Changed プロパティは真となります．したがって，次にイベント Script (Dispatcher)を起動すると，Changed プロパティは値変化が発生しているので，イベントが発生します．これは Vars イベントの仕様の動きです．

6.4. ユーザ共有オブジェクトの作成方法（Microsoft Visual Basic 6.0 の場合）

6.2.9.1 では，ユーザ共有オブジェクトの設定方法を紹介しましたが，ここでは作成方法を紹介します．作成には Visual Basic 6.0 (以下 VB6) を用いて解説しますが，Visual C++等でも同様に作成可能です．

作成するユーザ共有オブジェクトは，CaoScript から Show メソッド，Hide メソッドを呼び出すことで，フォームを表示，非表示するものです．

6.4.1. VB6 の起動

最初に VB を起動すると，図 6-23 のようなダイアログが表示されます．ユーザ共有オブジェクトは，外部からの呼び出しで動作しますので，ここでは，「ActiveX EXE」を選択します．

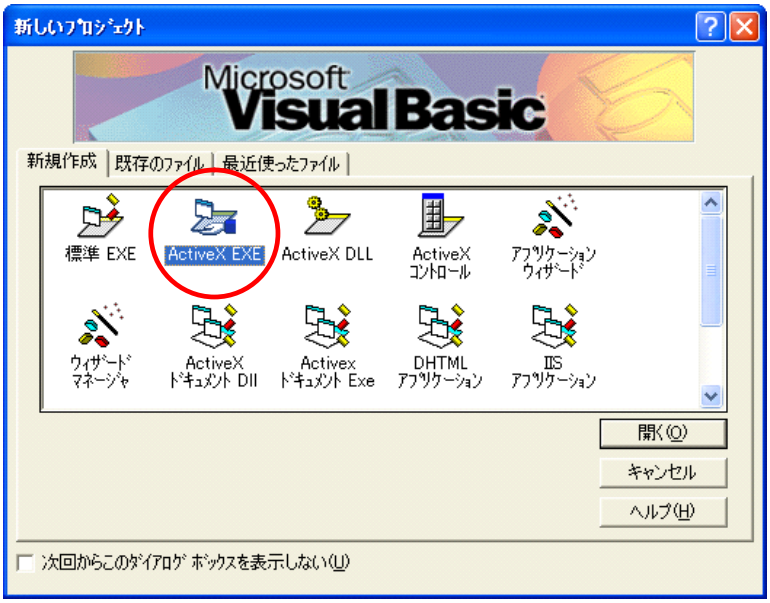


図 6-23 VB の新規プロジェクト画面

メイン画面を開きましたら，まずプロジェクトに Form を追加します．今回，紹介するユーザ共有オブジェクトは Form を使用します．GUI を持たない ActiveX EXE，ActiveX DLL なら Form の追加は必要ありません．Form の追加は，[プロジェクト(P)]の[フォームモジュールの追加(F)]を選択し，保存先を指定します．

次にプロジェクト全体の設定を行います．メニューの[プロジェクト(P)]を選択すると，図 6-24 のようなダイアログが表示されます．今回，作成するプロジェクト名を入力します．ここでは，”Sample”とします．

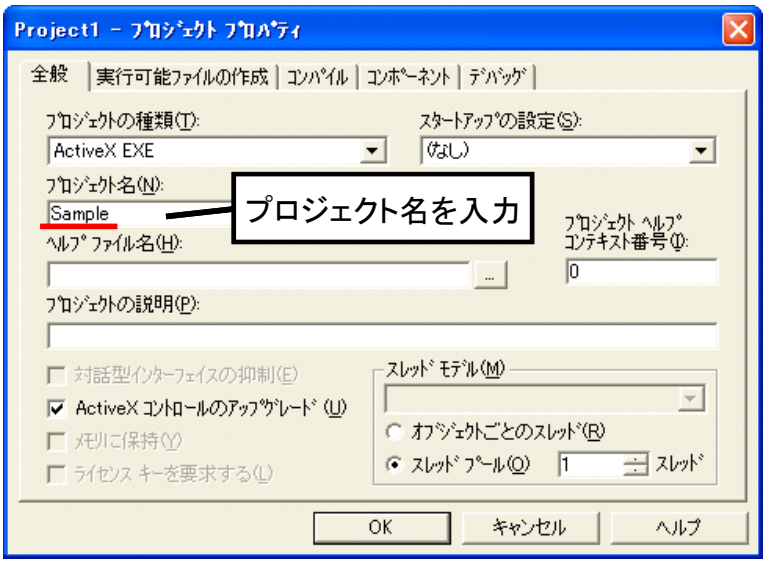


図 6-24 プロジェクトの設定［全般］タブ

つづいて，プロジェクトの[実行可能ファイルの作成]の設定を行います．ここでは，フォームのタイトル名を入力します．設定が完了したら[OK]してダイアログを閉じます．

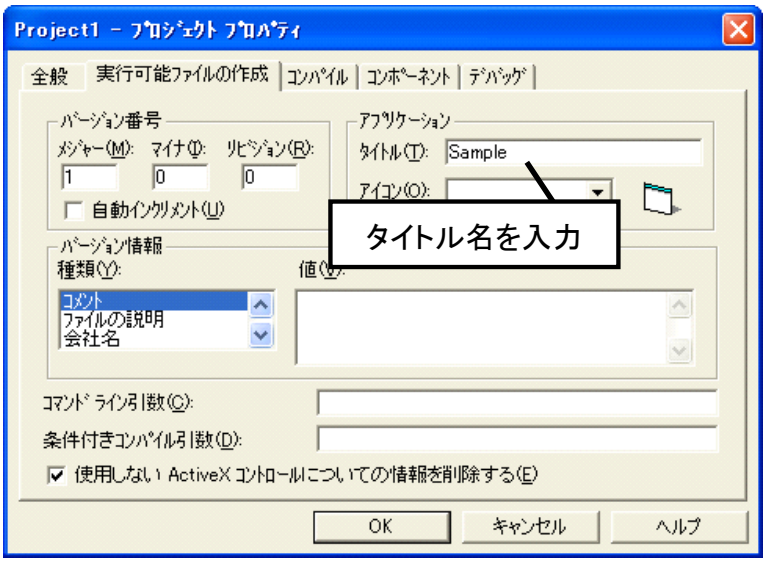


図 6-25 プロジェクトの設定［実行可能ファイルの作成］タブ

6.4.2. プログラムの入力（VB6）

プログラムの入力を行う前に，モジュールのオブジェクト名を設定します．今回はクラスモジュールの名前は”clsSample”に設定します．(図 6-26)

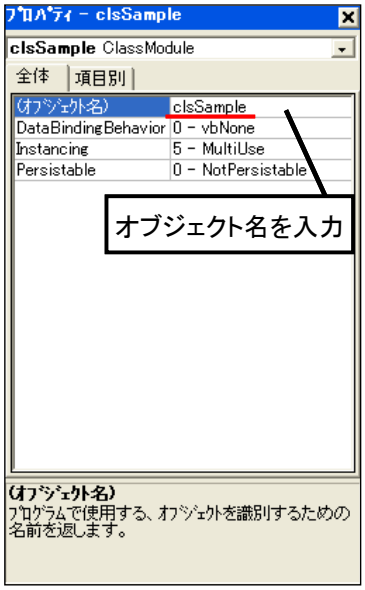
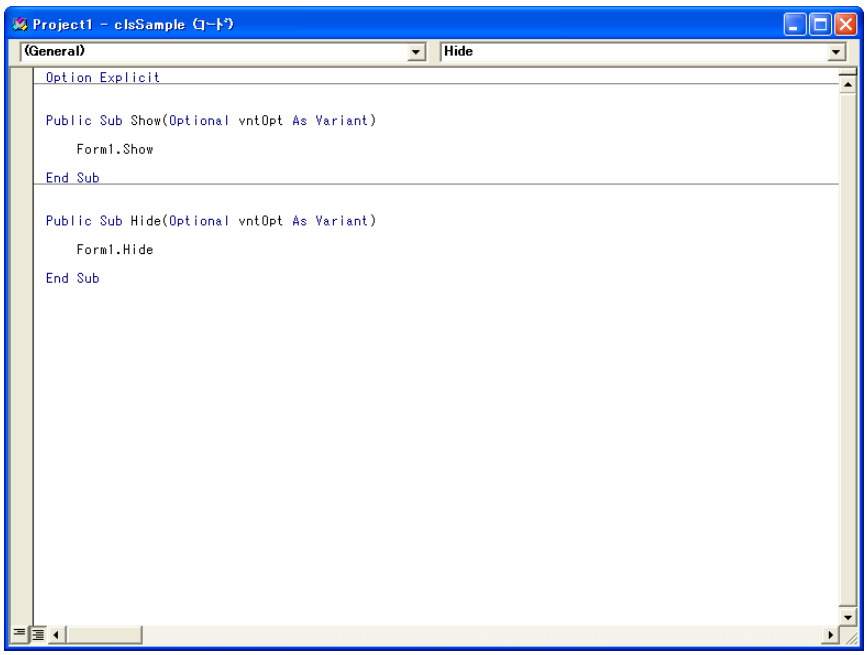


図 6-26 クラスモジュールのプロパティ

今回は CaoScript から呼び出される Show メソッド，Hide メソッドを実装します．
関数仕様は以下の通りです．

- Public Sub Show(Optional vntOpt As Variant)[5]
- Public Sub Hide(Optional vntOpt As Variant)



[5] 引数 vntOpt は拡張のために記述します．今回の Show, Hide のメソッドではこの引数は使用しません．

図 6-27 プログラム入力画面

6.4.3. 実行ファイルの作成

プログラムの入力が完了しましたら，次に実行ファイルを作成します．実行ファイルを作成するには，[File(F)]メニューの「Sample.exe」の作成を選択します．この名前は，プロジェクトの設定により依存します．

図 6-28 実行ファイルの作成

6.4.4. 作成したユーザ共有オブジェクトを追加

前節で作成したユーザ共有オブジェクトを追加します．追加の手順は 6.2.9.1 ユーザ共有オブジェクトを参照してください．ここでは，共有名を”Smp”とし，クラス名にはユーザ共有オブジェクトの作成の際につけた名前で，<プロジェクト名>.<クラスモジュール名>を設定します．

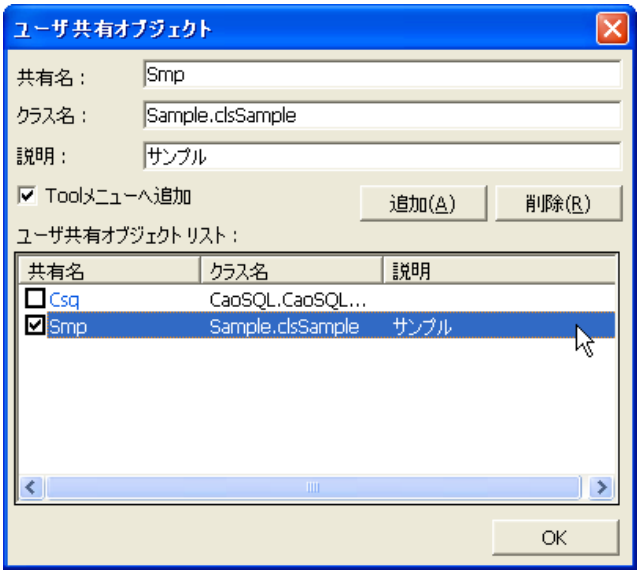


図 6-29 ユーザ共有オブジェクト設定画面

6.4.5. プログラム入力（CaoScript）

次に CaoScript から作成したユーザ共有オブジェクトを呼び出すプログラムを入力します.

呼び出す時に，先ほどのユーザ共有オブジェクトの登録でつけた共有名に公開されているメソッド名で呼び出すことが出来ます.

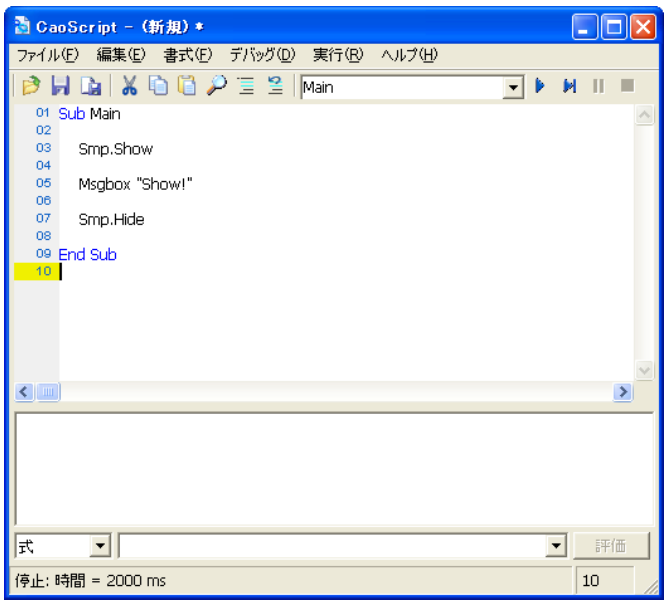


図 6-30 CaoScript 入力画面

6.4.6. 実行結果

上記の設定を行い，実行した結果です. Form が表示され，メッセージボックスを[OK]すると，Form は Hide

します.

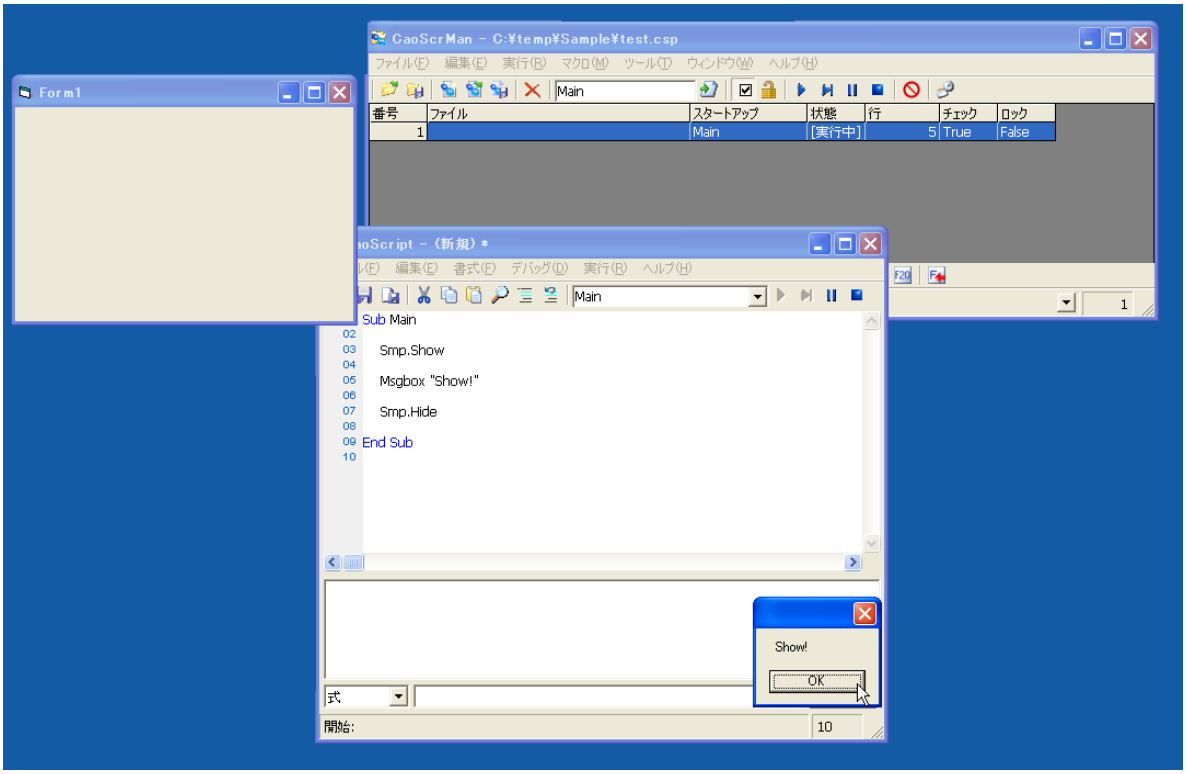


図 6-31 ユーザ共有オブジェクト実行結果

ユーザ共有オブジェクトと CaoScript のデータの連携は Vars を使うと容易に行うことができます.

6.5. マクロ登録による外部アプリケーションの起動方法

CaoScriptManager のマクロ登録機能()を使用して, 外部アプリケーションを起動する方法を紹介します. CaoScript で外部アプリケーションを起動するには, App オブジェクトの ShellExec を使用します. ()

起動させたいアプリケーションの Exe パスを指定して, 関数を記述します. ここでは, Windows に標準でインストールされている「メモ帳」を起動するようにします.

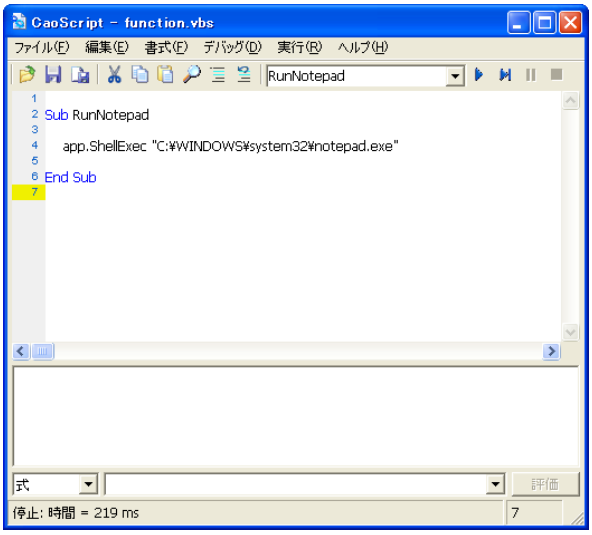


図 6-32 CaoScript で Notepad を起動

次に，記述した CaoScript を(例えば，)function.vbs のような名前で保存し，マクロに登録します．

図 6-33 Notepad のマクロ登録

6.6. 共有オブジェクト

CaoScript Manager は各 CaoScript でオブジェクトを共有するために下記のような組込みオブジェクトを提供します．このオブジェクトは実体が1つであり，各 CaoScript でそれを共有します．

表 6-2 共有オブジェクト一覧

名前	意味	備考
Csq	CaoSQLEngine オブジェクト	CaoSQL へのアクセス機能を提供する．[Create a CaoSQL object.]オプションで有効・無効を切り替える．

Cao	CaoWorkspace オブジェクト	常に有効. CaoWorkspace へのアクセス機能を提供する.
Pnl	VarsPanel オブジェクト	Operation Panel へのアクセス機能を提供する. ライセンスがインストールされている場合に有効.
Mgr	CaoScriptManager オブジェクト	常に有効. CaoScript Manager に登録されているスクリプトへのアクセス機能を提供する.
Csv	DataSheet オブジェクト	常に有効. Data Sheet のデータアクセス機能を提供する.

表 6-3 Pnl オブジェクトメンバー覧

メンバ	機能
Property Get Page () As Long	現在のページ番号を取得します.
Property Let Page (lPage As Long)	現在のページ番号を設定します.
Property Get TimeInterval () As Long	ページ更新間隔を取得します.
Property Let TimeInterval (lInterval As Long)	ページ更新間隔を設定します.
Property Get Visible () As Boolean	ウィンドウが開いている(True)か否(False)を取得します.
Property Let Visible (bVisible As Boolean)	ウィンドウが開いている(True)か否(False)を設定します.
Property Get WindowState () As Long	ウィンドウ状態を取得します. (2 以外: ノーマル, 2: 最大化)
Property Let WindowState (lState As Long)	ウィンドウ状態を設定します.
Sub Execute (strCmd As String, Optional vntParam As Variant)	将来の拡張用.
Sub Hide (Optional vntOpt As Variant)	パネルウィンドウを閉じます.
Sub PanelStart (Optional vntOpt As Variant)	Run モードに切り替えます.
Sub PanelStop (Optional vntOpt As Variant)	Design モードに切り替えます.
Sub Show (Optional vntOpt As	パネルウィンドウを開きます.

Variant)	

表 6-4 Mgr オブジェクトメンバー一覧

メンバ	機能
Property Get Count() As Integer	CaoScript オブジェクトの数を返します.
Property Get FileName() As String	CaoScrMan のプロジェクトファイル名を返します.
Property Get Path() As String	CaoScrMan のプロジェクトフォルダのパスを返します.
Property Get Script (vntID As Variant) As CaoScript.clsEngine	CaoScript オブジェクトを取得します. オブジェクトが公開するメンバは「Automation インターフェース」の章を参考にしてください. vntID は Index 番号またはファイル名を指定します. Index 指定する場合の有効範囲は 1～Count です.
Sub Hide (Optional vntOpt As Variant)	CaoScript Manager を非表示にします.
Sub Show (Optional vntOpt As Variant)	CaoScript Manager を表示します.

表 6-5 Csv オブジェクトメンバー一覧

メンバ	機能
Property Get TextMatrix (ByVal lRow1 As Long, ByVal lCol1 As Long, Optional ByVal lRow2 As Long = -1, Optional ByVal lCol2 As Long = -1, Optional strRowDelimiter As String, Optional strColDelimiter As String) As String	指定されたセルの値を返します. (例) x = csv.TextMatrix(1,1) y = csv.TextMatrix(1,1,1,2,””,”,”) z = csv.TextMatrix(1,1,2,2,”:”,”,”)
Property Let TextMatrix (ByVal lRow1 As Long, ByVal lCol1 As Long, Optional ByVal lRow2 As Long = -1, Optional ByVal lCol2 As Long = -1, Optional strRowDelimiter As String, Optional strColDelimiter As String, strText As String)	指定されたセルの値を設定します. (例) csv.TextMatrix(1,1) = “a” csv.TextMatrix(1,1,1,2,””,”,”) = “a,b” csv.TextMatrix(1,1,2,2,”:”,”,”) = “a1,b1:a2,b2:a3,b3”
Property Get ValueMatrix (ByVal	指定されたセルの値を返します.

lRow1 As Long, ByVal lCol1 As Long, Optional ByVal lRow2 As Long = -1, Optional ByVal lCol2 As Long = -1) As Variant	(例) x = csv.ValueMatrix(1,1) y = csv.ValueMatrix(1,1,1,2) z = csv.ValueMatrix(1,1,2,2)
Property Let ValueMatrix (ByVal lRow1 As Long, ByVal lCol1 As Long, Optional ByVal lRow2 As Long = -1, Optional ByVal lCol2 As Long = -1, vntValue As Variant)	指定されたセルの値を設定します. (例) Dim x csv.ValueMatrix(1,1) = x Dim y(2,1) csv.ValueMatrix(1,1,1,2) = y Dim z(2,2) csv.ValueMatrix(1,1,2,2) = z
Sub LoadFile (strFilename As String)	データをファイルからロードします. 現在のデータは上書きされます.
Sub SaveFile (strFilename As String)	データをファイルへ保存します.

6.6.1. CSV オブジェクト

CSV オブジェクトは CaoScript Manager が管理するデータシートへのアクセス機能を提供するオブジェクトです. データシートはプロジェクト保存時に CSV ファイルに保存されます. ファイル名はプロジェクトファイル名の拡張子を".csv"に置換した名称になります. この CSV オブジェクトの主な用途は"設定パラメータの保存"です. データシートは表形式になっています.

下記にサンプルを示します. このプログラムはロボットから現在位置を 10 回取得してそれをデータシートに格納しています.

```
' ロボットの現在位置を 10 回取得してデータシートへ保存
Sub SaveCurrentAngle

	cao.Controllers.Clear
	set rc = cao.AddController("","CaoProv.DENSO.NetwoRC","","Conn=eth:192.168.0.1")
	set rob = rc.AddRobot("")
	set ang = rob.AddVariable("@CURRENT_ANGLE")

	for k=1 to 10
		x = dat.ToVar(ang.Value)
		for i=1 to 6
			csv.TextMatrix(k,i) = x(i-1)
		next
	next

End Sub
```

この CSV オブジェクトを用いて VARS オブジェクト同様にタスク間通信することもできますが, 下記の理由でお勧めできません.

・ 文字列に限定される
・ 処理速度が遅い

したがって，タスク間のデータ共有や同期は VARS オブジェクトを使ってください．

6.7. コマンドラインオプション

CaoScrMan.exe を起動する際に下記のオプションを指定することができます．また，複数のオプションを指定することができます．コマンドラインオプションを使用する際は，スペースで区切られた文字列が引数として認識されますのでプロジェクトファイル名のファイルパスにスペースが含まれている場合などは注意が必要です．また，オプションを指定する場合にもオプション間はスペースを入れて指定してください．

6.7.1. Start オプション

このオプションを指定すると起動時に指定したプロジェクトファイル名を開き，タスクスケジューラを有効にした状態にします．書式を次に示します．

CaoScrMan.exe [<プロジェクトファイル名> [start]]

<プロジェクトファイル名>：Open するプロジェクトファイル名を指定します．
<Start>：起動時にタスクスケジューラを起動します．

(例 1) >CaoScrMan.exe “d:¥temp¥test.csp”
(例 2) >CaoScrMan.exe “d:¥temp¥test.csp” start

6.7.2. Icon オプション

このオプションを指定すると，CaoScriptMan を Icon 化した(タスクトレイに入った)状態で起動することができます．

CaoScrMan.exe [<プロジェクトファイル名> [icon]]

<プロジェクトファイル名>：Open するプロジェクトファイル名を指定します．
<Icon>：起動時に Icon 化(タスクトレイに入った状態に)します．

(例 1) >CaoScrMan.exe “d:¥temp¥test.csp” icon
(例 2) >CaoScrMan.exe “d:¥temp¥test.csp” start icon

7. サンプルプログラム

以下に CAO オブジェクト, Dbg オブジェクト, VARS オブジェクト, App オブジェクトの各メンバを使用するサンプルを示します.

List 7-1 Sample.vbs

```
Sub Main
  TestVARS
  TestAPP
  TestCAO
  TestTimer
End Sub

Sub TestVARS
  ' VARS オブジェクトの要素:10 番にデータを格納
  Vars(10).Value = "VarsTest"
  ' VARS オブジェクトの要素:10 番に格納した値をログウインドウに表示します
  dbg.Output Vars(10).Value
End Sub

Sub TestAPP
  ext.Delay 1000          ' Ext オブジェクトのディレイ(1000ms)を実行
End Sub

Sub TestCAO
  ' CAO オブジェクトから CaoController オブジェクトを作成
  Set Ctrl = cao.AddController("", "CaoProv.DataStore")

   'CaoVariable オブジェクトを作成
  Set Var1 = Ctrl.AddVariable("Var1","")

  vntAry = Array("test1", "test2", 1974)
  Var1.Value = vntAry
   ' CaoVariable に格納された配列の全要素をスペース区切りでログウインドウに表示
  dbg.Output Var1.Name & " : " & Join(Var1.Value)
End Sub

Sub TestTimer
  app.TimerInterval = 100  '100ms にイベント間隔を設定
  ext.Delay 100            'イベント発生時間まで待つ
End Sub

'タイマイベント
Sub App_OnTimer
  dbg.Output "Event Test"
End Sub
```

付録A.

付録A.1. VBScript リファレンス

◆ イベント

イベント	機能
Initialize	クラスのインスタンスが生成された時に発生します
Terminate	クラスをインスタンスが終了された時に発生します

◆ ステートメント

ステートメント	機能
Call	**Sub** プロシージャおよび **Function** プロシージャに制御を渡すフロー制御ステートメントです
Class	クラスの名前を宣言します
Const	リテラル値の代わりに使用する定数を宣言します
Dim	変数を宣言してメモリ領域を割り当てます
Do...Loop	指定された条件が真 **(True)** である間，または条件が真 **(True)** になるまで，一連のステートメントを繰り返し実行するフロー制御ステートメントです
Erase	静的配列の要素を再初期化したり，動的配列に割り当てたメモリを解放します
Execute	指定された 1 つ以上のステートメントを実行します
ExecuteGlobal	スクリプトのグローバル名前空間で指定された 1 つ以上のステートメントを実行します
Exit	**Do...Loop**，**For...Next**，**Function** プロシージャまたは **Sub** プロシージャから抜け出すためのフロー制御ステートメントです
For...Next	指定された回数だけ，一連のステートメントを繰り返すフロー制御ステートメントです
For Each...Next	配列やコレクションの各要素に対して，一連のステートメントを繰り返し実行するフロー制御ステートメントです
Function	**Function** プロシージャの名前，引数を宣言し，**Function** プロシージャの始まりを示します
If...Then...Else	式の値に基づいて，条件付きの実行を行うフロー制

	御ステートメントです
On Error	エラー処理を有効にします
Option Explicit	スクリプト内のすべての変数に対して，明示的な宣言を強制します
Private	プライベート変数を宣言して，格納領域を割り当てます
Property Get	プロパティの値を取得する **Property** プロシージャを構成する名前，引数，およびコードを宣言します
Property Let	プロパティの値を代入する **Property** プロシージャを構成する名前，引数，およびコードを宣言します
Property Set	オブジェクトへの参照を設定する **Property** プロシージャを構成する名前，引数，コードを宣言します
Public	パブリック変数を宣言して，格納領域を割り当てます
Randomize	乱数ジェネレータを初期化 (乱数系列を再設定) します
ReDim	動的配列変数を宣言し，そのメモリ領域の割り当てや再割り当てを行います
Rem	プログラム内にコメントを記述するときに指定します
Select Case	条件式の値に従って，複数のステートメント ブロックのいずれかを実行させるフロー制御ステートメントです
Set	オブジェクトへの参照を変数またはプロパティに代入します
Sub	**Sub** プロシージャの名前，引数を宣言し，**Sub** プロシージャの始まりを示します
While...Wend	指定された条件が真 (**True**) である間，一連のステートメントの実行を繰り返すフロー制御ステートメントです
With	1 つのオブジェクトに対して一連のステートメントを実行します

◆ **メソッド**

メソッド	機能
Clear	**Err** オブジェクトのすべてのプロパティの設定値をクリアします
Execute	指定された文字列を正規表現で検索します

Raise	実行時エラーを生成します
Replace	正規表現による検索で見つかったテキストを置換します
Test	指定された文字列を正規表現で検索します

◆ **プロパティ**

プロパティ	機能
Description	エラーと関連付ける，エラーを説明する文字列を設定します
FirstIndex	検索対象の文字列内で一致が見つかった場所を返します
Global	ブール値を設定するか返します
HelpContext	ヘルプ ファイルのトピックを表すコンテキスト番号を設定します．値の取得も可能です
HelpFile	ヘルプ ファイルへのパスを設定します．値の取得も可能です
IgnoreCase	パターン検索で大文字と小文字を区別するかどうかを示すブール **(Boolean)** 値を設定します
Length	検索対象の文字列内で一致した文字列の長さを返します
Number	エラーを指定する数値を設定します．値の取得も可能です
Pattern	検索される正規表現のパターンを設定します．値の取得も可能です
Source	最初にエラーを発生させたオブジェクトまたはアプリケーションの名前を設定します．値の取得も可能です
Value	検索対象の文字列内で一致した値またはテキストを返します

◆ **関数**

プロパティ	機能
Abs	指定された数値の絶対値を返す数値演算関数です

Array	配列が格納されたバリアント型 (**Variant**) の値を返します
Asc	指定された文字列内の, 最初の文字の ANSI コードまたはシフト JIS コードを返す変換関数です
Atn	指定された角度のアークタンジェントを倍精度浮動小数点数型 (**Double**) の値で返す数値演算関数です
CBool	指定された式をバリアント型 (内部処理形式がブール型 (**Boolean**) の **Variant**) にする変換関数です
CByte	指定された式をバリアント型 (内部処理形式がバイト型 (**Byte**) の **Variant**) にする変換関数です
CCur	指定された式をバリアント型 (内部処理形式が通貨型 (**Currency**) の **Variant**) にする変換関数です
CDate	指定された式をバリアント型 (内部処理形式が日付型 (**Date**) の **Variant**) にする変換関数です
CDbl	指定された式をバリアント型 (内部処理形式が倍精度浮動小数点数型 (**Double**) の **Variant**) にする変換関数です
Chr	指定された ANSI コードまたはシフト JIS コードに対応する文字を返します
CInt	指定された式をバリアント型 (内部処理形式が整数型 (**Integer**) の **Variant**) にする変換関数です
CLng	指定された式をバリアント型 (内部処理形式が長整数型 (Long) の **Variant**) にする変換関数です
Cos	指定された角度のコサインを倍精度浮動小数点数型 (**Double**) の値で返す数値演算関数です
CreateObject	オートメーション オブジェクトを作成します
CSng	指定された式をバリアント型 (内部処理形式が単精度浮動小数点型 (Single) の **Variant**) にする変換関数です
CStr	指定された式をバリアント型 (内部処理形式が文字列型 (**String**) の **Variant**) にする変換関数です
Date	現在のシステムの日付を返します
DateAdd	指定された時間間隔を加算した日付を返します
DateDiff	指定された 2 つの日付の時間間隔を返します
DatePart	指定された日付の一部分を返します

DateSerial	引数に指定された年, 月, 日に対応する日付をバリアント型 (内部処理形式が日付型 (**Date**) の **Variant**) の値で返します
DateValue	指定された日付をバリアント型 (内部処理形式が日付型 (**Date**) の **Variant**) の値で返します
Day	月の何日かを表す 1～31 の範囲の整数を返します
Eval	式を評価し, 結果を返します
Exp	指数関数 (*e* を底とする数式のべき乗) を計算する数値演算関数です
Filter	指定されたフィルタ条件に基づいた文字列配列のサブセットを含むゼロ ベースの配列を返します
Fix	引数に指定された数式の小数部を切り捨て, 整数部分だけを返す数値演算関数です
FormatCurrency	システムの [コントロール パネル] で定義されている通貨記号を使って通貨形式の文字列に書式設定して返す文字列処理関数です
FormatDateTime	日付形式または時刻形式の文字列に書式設定して返す文字列処理関数です
FormatNumber	数値形式の文字列に書式設定して返す文字列処理関数です
FormatPercent	パーセント記号 (%) が付加されたパーセント形式 (100 で乗算した) の文字列に書式設定して返す文字列処理関数です
GetLocale	現在のロケール ID の値を返します
GetObject	ファイルから取得したオートメーション オブジェクトへの参照を返します
GetRef	イベントとバインドできるプロシージャに対する参照を返します
Hex	指定された値を 16 進数で表した文字列で返します
Hour	1 日の時刻を表す 0～23 の範囲の整数を返します
InputBox	ダイアログボックスにメッセージとテキストボックスを表示し, テキストが入力されるか、またはボタンがクリックされると, テキスト ボックスの内容を返します.
InStr	ある文字列 (*string1*) の中から指定された文字列 (*string2*) を検索し, 最初に見つかった文字位置 (先頭からその位置までの文字数) を返す文字列

	処理関数です
InStrRev	ある文字列 (*string1*) の中から指定された文字列 (*string2*) を最後の文字位置から検索を開始し，最初に見つかった文字位置 (先頭からその位置までの文字数) を返す文字列処理関数です
Int	引数に指定された数式の小数部を切り捨て，整数部分だけを返す数値演算関数です
IsArray	変数が配列であるかどうかを調べ，結果をブール値で返します
IsDate	式を日付に変換できるかどうかを調べ，結果をブール値で返します
IsEmpty	変数が初期化されたかどうかを調べ， 結果をブール値で返します
IsNull	式に **Null** 値が含まれているかどうかを調べ，結果をブール値で返します
IsNumeric	式が数値として評価できるかどうかを調べ，結果をブール値で返します
IsObject	式がオートメーション オブジェクトを参照しているかどうかを調べ，結果をブール値で返します
Join	配列に含まれる各要素の内部文字列を結合して作成される文字列を返します
LBound	配列の指定された次元で使用できるインデックス番号の最小値を返します
LCase	アルファベットの大文字を小文字に変換する文字列処理関数です
Left	文字列の左端から指定された文字数分の文字列を返します
Len	指定された文字列の文字数を返します
LoadPicture	ピクチャ オブジェクトを返します．32 ビット版プラットフォームでのみ使用できます
Log	数値の自然対数を返す数値演算関数です
LTrim	先頭にスペースのない文字列のコピーを返します
Mid	文字列から指定された文字数分の文字列を返します
Minute	時刻の分を表す 0～59 の範囲の整数を返します
Month	1 年の何月かを表す 0～12 の範囲の整数を返しま

	す
MonthName	指定された月を表す文字列を返します
MsgBox	ダイアログ ボックスにメッセージを表示し，ボタンがクリックされるのを待って，どのボタンがクリックされたかを示す値を返します
Now	コンピュータのシステムの日付と時刻の設定に基づいて，現在の日付と時刻を返します
Oct	指定された値を 8 進数で返します
Replace	指定された文字列の一部を，別の文字列で指定された回数分で置換した文字列を返します
RGB	RGB カラー値を表す値を返します
Right	文字列の右端から指定された文字数分の文字列を返します
Rnd 関数	単精度浮動小数点数型 (**Single**) の乱数を返します
Round	指定された小数点位置で丸めた数値を返します
RTrim	末尾にスペースのない文字列のコピーを返します
ScriptEngine	使用中のスクリプト言語を表す文字列を返します
ScriptEngineBuildVersion	使用中のスクリプト エンジンのビルド バージョン番号を返します
ScriptEngineMajorVersion	使用中のスクリプト エンジンのメジャー バージョン番号を返します
ScriptEngineMinorVersion	使用中のスクリプト エンジンのマイナ バージョン番号を返します
Second	時間の秒を表す 0～59 の整数を返します
Sgn	引数に指定された数式の符号を返します
Sin	指定された角度のサインを返します
Space	指定された数のスペースから成る文字列を返します
Split	各要素に区切られた文字列からゼロ ベースの 1 次元配列を作成し，返します
Sqr	数式の平方根を返します
StrComp	文字列比較の結果を表す値を返します
String	指定された文字コード (ASCII またはシフト JIS コード) の示す文字，または文字列の先頭文字を，指定された文字数だけ並べた文字列を返します
StrReverse	指定された文字列の文字の並び順を逆にした文字

	列を返します
Tan	指定された角度のタンジェントを返します
Time	現在のシステムの時刻を返します
Timer	午前 0:00 以降に経過した秒数を返します
TimeSerial	引数に指定された時，分，秒に対応する時刻を返します
TimeValue	時刻を返します
Trim	先頭または末尾にスペースのない文字列のコピーを返します
TypeName	指定された変数に関する情報を提供する文字列を返します
UBound	配列で指定した次元で使用できるインデックス番号の最大値を返します
UCase	指定されたアルファベットの小文字を大文字に変換します
VarType	変数の内部処理形式を表す値を返します
Weekday	何曜日であるかを表す 1 (日曜)～7 (土曜)の範囲の値を返します
WeekdayName	指定された曜日を表す文字列を返します
Year	年を表す整数を返します

◆ オブジェクトとコレクション

メソッド	機能
Class	クラスのイベントにアクセスする手段を提供します
Err	**Err** オブジェクトは，実行時エラーに関する情報を保有しています
Match	正規表現で一致した文字列の読み取り専用プロパティにアクセスする手段を提供します
Matches	正規表現の **Match** オブジェクトのコレクションです
RegExp	正規表現の機能を提供します
SubMatches	正規表現のサブマッチ文字列のコレクションです