‘The Blackboard’プロバイダ

黒板モデル形式データ共有

Version 1.1.0

ユーザーズ ガイド

December 18, 2013

【備考】

【改版履歴】

バージョン	日付	内容
1.0.0.0	2006-02-23	初版.
1.0.0.1	2010-02-10	エラーコード追加
1.0.0	2012-07-17	ドキュメントのバージョンルールを変更
1.1.0	2013-12-18	調停機能追加

【対応機器】

機種	バージョン	注意事項

目次

1. はじめに

本書は，ORiN2 アプリケーション間でデータ共有を簡単にする“The Blackboard”プロバイダのユーザガイドです．このプロバイダは黒板モデル[1]を使った問題解決システムを構築する場合に使われることを想定しています．ORiN2 は様々なシステムのプラットフォームとして活用できますが，このプロバイダを使うことで更に問題解決システムなどへの適用もより自然にできるようになります．

この“The Blackboard”プロバイダは内部に変数テーブルを格納し，その変数テーブルを共有することでORiN2 アプリケーション間でのデータ共有を実現します．ここで言う一つの変数テーブルが一つの「黒板」にあたります．しかも，共有されるデータは整数や実数，文字列といった単純型データだけでなく，その配列や画像などのバイナリデータを含むことができます．

また，このプロバイダを用いれば，同一マシンでのアプリケーション間データ共有だけでなく，別のマシンで稼働しているアプリケーションとのデータ共有も簡単に実現できます．CAP プロバイダなどと組み合わせると，容易にインターネットを介した問題解決システムを開発できます．

本書では，この“The Blackboard”プロバイダの機能と実装されているメソッドについて説明します．

【参考】黒板システムとは？

黒板モデルを利用した問題解決システムはしばしば下記のように比喩されます．

「一つの大きな黒板の周りにスペシャリストのグループが座って居るのを想像して下さい．そのスペシャリスト達はある問題を解決するために，作業場として黒板を用い，協力して問題解決作業を進めます．問題と初期データが黒板に書き込みされると，問題解決の作業が始まります．スペシャリスト達は黒板を見ながら，問題解決に自分の専門知識を適用する機会を待っています．あるスペシャリストが問題解決に貢献することができる重要な情報を見つけたとき，他のスペシャリストに彼らの専門知識をいかす機会を与えられることを期待して，そのスペシャリストは新しい情報を書き込みます．そして，このプロセスは問題が解決するまで続けられます．」(文献[1]より抜粋)

2. プロバイダの概要

2.1. 概要

ORiN2 の CAO(Controller Access Object)は FA 機器を抽象化したオブジェクトモデルです. 従って, その対象とするリソースには, タスクや変数, プログラムファイル, ロボットなど様々なものがありますが, “The Blackboard”プロバイダは“変数”リソースだけを持つ特定用途のプロバイダです. 内部的には, コントローラごとに変数データを管理しますので, コントローラ名が違えば, 同じ変数名でも違う変数として扱われます.

格納できるデータ型は VARIANT 型データで, その中で配列や画像データなどのバイナリデータも含めて様々な型のデータを格納することができます. このプロバイダを使うと, 複数のアプリケーション間で複雑なデータを共有する場合でも簡単に実現することができます[1].

冒頭で述べたように, このプロバイダは黒板モデルを使った問題解決システムを構築する場合に使われることを想定しています. 黒板モデルを使った問題解決システムの代表例として HEARSAY Ⅱ [2]という音声認識システムがあります. HEARSAY Ⅱ の構成を図 2-1 に示します. この中で, 赤い点線の部分が黒板とのインタラクション部分で, この部分を開発する際に本プロバイダが活用できます.

CAO のモデルから見ると, Blackboard Monitor や複数の Knowledge Source が CAO アプリケーションに対応します. それらのアプリケーションは, CAO のオブジェクトを操作することで黒板を操作します.

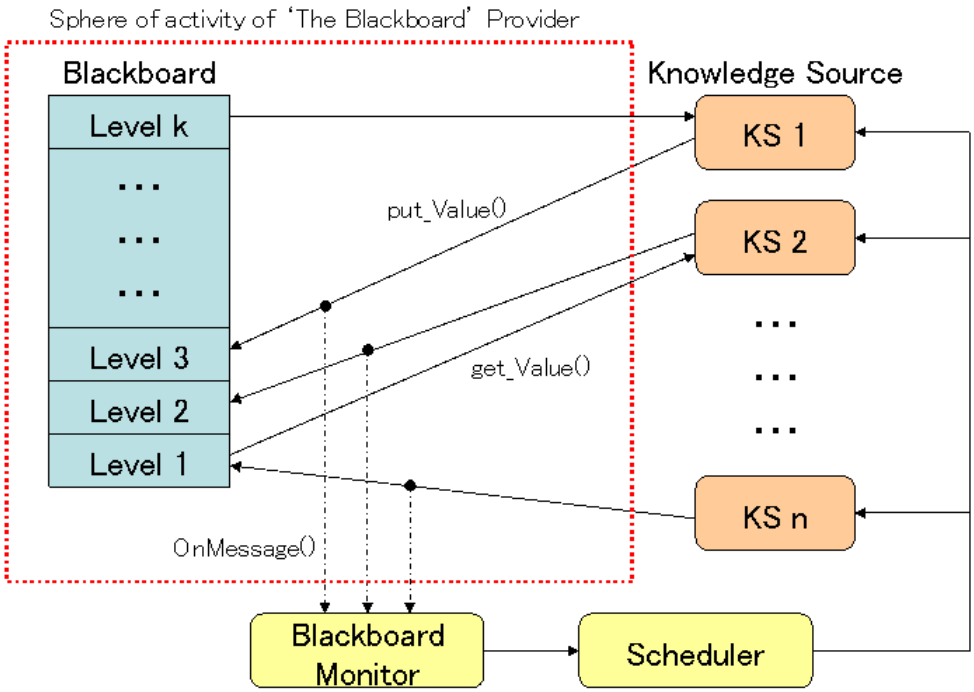

図 2-1 HEARSAY Ⅱ の構成

[1] この意味では“DataStore”プロバイダと機能的には似ていますが, “DataStore”プロバイダではイベントを使ってデータの変化を取得するようなことはできません.

表 2-1 Blackboard プロバイダ

ファイル名	CaoProvBlackboard.dll
ProgID	CaoProv.Blackboard
レジストリ登録[2]	regsvr32 CaoProvBlackboard.dll
レジストリ登録の抹消	regsvr32 /u CaoProvBlackboard.dll

【機能追加】調停機能（Ver. 1.1.0 以降）

黒板モデルには，黒板への読み書きを制御する監督者がいないので，黒板利用者（図 2-1 の KS にあたる）は自由に情報を読み書きできます．Ver. 1.1.0 以降で追加された調停機能を使うことで，調停者だけが読み書きの権利を制御できるようになります．つまり，調停者が黒板を書き込み（Write）禁止に設定すれば黒板利用者は読み込み（Read）のみ可能になります．

例えば，「温度を監視して暑くなってきたら自動的に窓を開ける」アプリと，「泥棒や雨などを監視して自動的に閉める」アプリを一つのシステムに共存させようとしたとき，この調停機能を使うとエレガントにシステムをデザインできます．この場合は当然閉めるアプリの方が優先だと思うので，そのアプリを調停者にすればよいのです．

[2] ORiN SDK でインストールした場合は手動で登録／抹消する必要はありません．

2.2. メソッド・プロパティ

2.2.1. CaoWorkspace::AddController メソッド

"The Blackboard"プロバイダでは，一つのコントローラクラス(CaoController)のオブジェクトを一人の黒板利用者に見立てています．ここでいう黒板とは変数テーブルを管理するもので，変数テーブルは黒板ごとに管理されます．新しい利用者を作成するにはワークスペースクラス(CaoWorkspace)の AddController メソッドを使います．

AddController メソッドの第一引数のコントローラ名に黒板の名前を指定します．指定した名前が既に存在する場合はその黒板(変数テーブル)が共有され，逆に存在しない場合は，新規に黒板(変数テーブル)が作成されます[3]

実行マシン名が各アプリケーションで違っているとプロバイダは違うプロセス空間にロードされますので，同じコントローラ名を指定してもそれを共有することはできません．

また，コントローラ名（黒板名）は大文字・小文字が区別されるので注意して下さい．コントローラオブジェクトが削除された時点で，もしそれを参照しているクライアントプログラムが一つも存在しない場合は，その時点でその黒板(変数テーブル)も破棄されます．一度破棄されたデータは復元できません．

以下に AddController の引数仕様を示します．

```
AddController
(
      "<コントローラ名>",                    // コントローラ名（黒板名）
      "CaoProv.Blackboard ",                 // プロバイダ名．固定．
      "<マシン名>",                          // プロバイダの実行マシン名
      "<オプション>"                         // オプション文字列
)
```

以下にオプション文字列に指定するリストを示します．

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Arbitrator=<TRUE/FALSE>	黒板参加者の参加時点での調停権を設定します． （デフォルト:FALSE） 既に別の参加者が調停権を確保している状態でこのオプションを TRUE にした場合，AddController メソッドはエラーを返します．

以下に例を示します．

```
AddController
(
    "BB1",                                  // 黒板名 ＝ BB1
    "CaoProv.Blackboard",
    "",                                     // CAO エンジンプロセスで実行
```

[3] これは，異なるワークスペースオブジェクトで同じコントローラ名を指定した場合の動作です．同じワークスペースオブジェクトに同じ名前のコントローラオブジェクトの追加は CAO の仕様でエラーになります．

```
    ""
)
```

2.2.2. CaoController::AddVariable メソッド

このCaoControllerクラスのAddVariableメソッドで，黒板（変数テーブル）に変数を登録します．追加された変数は，このメソッドを提供するCaoControllerオブジェクトの管理テーブルに登録されます．

指定した変数名が既に存在する場合はその変数が共有され，逆に存在しない場合は，新規に変数が作成されます[4]．変数名もコントローラ名と同じく大文字・小文字が区別されるので注意して下さい．また，"@"で始まる変数名はシステム変数として解釈されますので"@"で始まるユーザ変数を追加することはできません．

変数値はVARIANT型で，単純変数はもちろん配列データや画像などのバイナリデータも格納することができます．以下に，AddVariableの引数仕様を示します．

```
AddVariable
(
    "<変数名>",          // 変数テーブルに登録する変数名．
    "<オプション>"       // （未使用）
)
```

以下に例を示します．

```
AddVariable
(
    "ABC",               // ABC 変数を指定．
    ""
)
```

2.2.3. CaoController::Execute メソッド

変数テーブルに対してコマンドを実行します．

カテゴリ	コマンド	機能	
変数制御			
	PutItemAttribute	変数の属性設定	P. 9
	PutItemHelp	変数のヘルプ文字列設定	P. 9
調停機能			
	TakeArbitrator	調停権取得	P. 10
	ResignArbitrator	調停権解放	P. 10

[4] これは，異なるコントローラオブジェクトで同じ変数名を指定した場合の動作です．同じコントローラオブジェクトに同じ名前の変数オブジェクトの追加はCAOの仕様でエラーになります．

PutItemAttribute

構文 *object*.PutItemAttribute(<Item Name>, <Attribute>)

引数 <Item Name> = VT_I4: 変数名

<Attribute> = VT_I4: 属性

1	READ	読み込みのみ
2	WRITE	書み込みのみ
3	READ/WRITE	読み込み/書込み可

戻り値 None

説明 指定した変数に属性を設定します.

調停者の存在によって, 変数の動作やこのコマンドの実行権が以下のように異なります.

	変数の動作	コマンドの実行権
調停者あり	設定値	調停者のみ
調停者なし	READ/WRITE	利用者全員

このコマンドを実行したことで設定値の内容が変更された場合は"ATTR_CHANGED"(Number=4)の OnMessage イベントが発生します.

PutItemHelp

構文 *object*.PutItemHelp(<Item Name>, <Help>)

引数 <Item Name> = VT_I4: 変数名

<Help> = VT_BSTR: ヘルプ文字列

戻り値 None

説明 指定した変数にヘルプ文字列を設定します.

調停者の存在によってこのコマンドの実行権が以下のように異なります.

	コマンドの実行権
調停者あり	調停者のみ
調停者なし	利用者全員

このコマンドを実行したことで設定値の内容が変更された場合は"HELP_CHANGED"(Number=5)の OnMessage イベントが発生します.

TakeArbitrator

構文 *object*.TakeArbitrator

引数 None

戻り値 None

説明 調停権を取得して調停者になります.
既に調停者が存在する場合はこのコマンドは失敗します.
このコマンドを実行したことで調停権を取得した場合は"ARBIT_TAKEN"(Number=9)の OnMessage イベントが発生します.

ResignArbitrator

構文 *object*.ResignArbitrator

引数 None

戻り値 None

説明 調停権を解放して通常の利用者に戻ります.
このコマンドを実行したことで調停権を解放した場合は"ARBIT_RESIGNED"(Number=8)の OnMessage イベントが発生します.

2.2.4. CaoController::get_VariableNames プロパティ

get_VariableNames プロパティでは, AddVariable メソッドで追加したユーザ変数の一覧を文字列型の配列を格納した VARIANT 型で取得します. このリストにはシステム変数は含まれません.

2.2.5. CaoController::get_ID プロパティ

get_ID プロパティでは, 黒板参加者の ID を BSTR 型で取得します.

2.2.6. CaoController::OnMessage イベント

"The Blackboard"プロバイダでは, 以下の OnMessage イベントが発生します.

ADDED

発生条件 CaoController::AddVariable()実行によって変数テーブルに新規の変数が追加されたとき

Number 1

Description ADDED

Destination 変数名

Value Empty

説明 変数追加イベント
黒板(変数テーブル)に変数が追加されたことを通知します.
このイベントは最初にその名前の変数が作成されたときのみに発生します.

DELETED

発生条件 CaoVariable オブジェクトの解放によって変数テーブルから変数が削除されたとき

Number 2

Description DELETED

Destination 変数名

Value Empty

説明 変数削除イベント
黒板(変数テーブル)に変数が削除されたことを通知します.
変数テーブルの変数を参照している CaoVariable オブジェクトが全てなくなったときに発生します.

CHANGED

発生条件 CaoVariable::put_Value()によって変数の値が変更されたとき

Number 3

Description CHANGED

Destination 変数名

Value CaoVariable::put_Value()で設定した値

説明 変数変更イベント
黒板(変数テーブル)の変数の値が変更されたことを通知します.
現在の値と同じ値を設定された場合はこのイベントは発生しません.

ATTR_CHANGED

発生条件 CaoController::Execute()の PutItemAttribute コマンドによって変数の属性が変更されたとき

Number 4

Description ATTR_CHANGED

Destination 黒板名(コントローラ名)

Value CaoController::Execute()の第 2 引数(パラメータ)

説明 変数属性変更イベント
黒板(変数テーブル)の変数の属性が変更されたことを通知します.
現在の値と同じ値を設定された場合はこのイベントは発生しません.

HELP_CHANGED

発生条件 CaoController::Execute()の PutItemHelp コマンドによって変数のヘルプ文字列が変更されたとき

Number 5

Description HELP_CHANGED

Destination 黒板名(コントローラ名)

Value CaoController::Execute()の第 2 引数(パラメータ)

説明 変数ヘルプ文字列変更イベント
黒板(変数テーブル)の変数のヘルプ文字列が変更されたことを通知します.
現在の値と同じ値を設定された場合はこのイベントは発生しません.

CTRL_ADDED

発生条件 CaoWorkspace::AddController()の実行によってコントローラオブジェクト(黒板利用者)が追加されたとき

Number 6

Description CTRL_ADDED

Destination 黒板名(コントローラ名)

Value Empty

説明 コントローラ(黒板利用者)追加イベント
黒板(変数テーブル)の利用者が追加されたことを通知します.

CTRL_DELETED

発生条件 CaoController の解放によってコントローラ(黒板利用者)が削除されたとき

Number 7

Description CTRL_DELETED

Destination 黒板名(コントローラ名)

Value Empty

説明 コントローラ(利用者)削除イベント
黒板(変数テーブル)の利用者が削除されたことを通知します.

ARBIT_RESIGNED

発生条件 CaoController::Execute()の ResignArbitrator コマンドによって調停権の解放がされたとき

Number 8

Description ARBIT_RESIGNED

Destination 黒板名(コントローラ名)

Value CaoController::Execute()の第 2 引数(パラメータ)

説明 調停権解放イベント
黒板(変数テーブル)の調停権が解放されたことを通知します.

ARBIT_TAKEN

発生条件 CaoController::Execute()の TakeArbitrator コマンドによって調停権の取得がされたとき

Number 8

Description ARBIT_TAKEN

Destination 黒板名(コントローラ名)

Value CaoController::Execute()の第 2 引数(パラメータ)

説明 調停権取得イベント
黒板(変数テーブル)の調停権が取得されたことを通知します.

2.2.7. CaoVariable::get_Value プロパティ

黒板(変数テーブル)に登録されている変数の現在の値を VARIANT 型で取得します.

調停者が設定されているかつ変数に読込み属性が設定されていないとき, このプロパティはエラーを返します.

2.2.8. CaoVariable::put_Value プロパティ

黒板(変数テーブル)に値を VARIANT 型で登録します. この新しい値が前の値と違う場合はコントローラクラスの OnMessage()イベントが発生します. 同値判定は oleaut32.lib の VarCmp()関数を使用していますので, 詳細は Microsoft Corp. の API ドキュメントを参照下さい.

調停者が設定されているかつ変数に書込み属性が設定されていないとき, このプロパティはエラーを返します.

2.2.9. CaoVariable::get_Attribute プロパティ

黒板(変数テーブル)に登録されている変数の現在の属性値を取得します.

2.2.10. CaoVariable::get_Help プロパティ

黒板(変数テーブル)に登録されている変数の現在のヘルプ文字列を取得します.

2.3. 変数一覧

2.3.1. コントローラクラス

“The Blackboard”プロバイダでは，コントローラクラスの表 2-3 のシステム変数が利用可能です．それぞれの意味は下表を参照して下さい．

表 2-3 コントローラクラス システム変数一覧

変数名	データ型	説明	属性	
			get	put
@COUNT	VT_I4	ユーザ変数の数を返します．システム変数は含みません．	○	-
@CURRENT_TIME	VT_DATE	プロバイダが実行されているマシンのローカルタイム．	○	-
@VERSION	VT_BSTR	バージョン．	○	-

2.4. エラーコード

“The Blackboard”プロバイダでは，固有のエラーコードはありません．ORiN2 共通エラーについては，「ORiN2 プログラミングガイド」のエラーコードの章を参照してください．

3. プログラム設計ヒント

ここでは，本プロバイダを使って黒板システムを構築する場合の設計ヒントを紹介します．

[黒板内のデータの階層化]

問題解決システムに黒板モデルを適用する場合，そのシステムが解決しようとしている問題に関する様々なデータが黒板に記入されます．一般的にそれらのデータは階層化されている方が，各ナレッジソースは扱いやすくなります．

"The Blackboard"プロバイダは，データの階層化のための特別な仕組みは持っていませんが，変数名を工夫することで，擬似的にデータを階層化することができます．

例えば，図 3-1 の例ではピリオド". "で区切って階層を表現しています．各ノードは子ノードを含むノードも含めてすべてのノードが一つの値を格納することができます．その格納するデータのデータ型は任意であり，一つのツリーで様々なデータ型のデータを格納することができます．

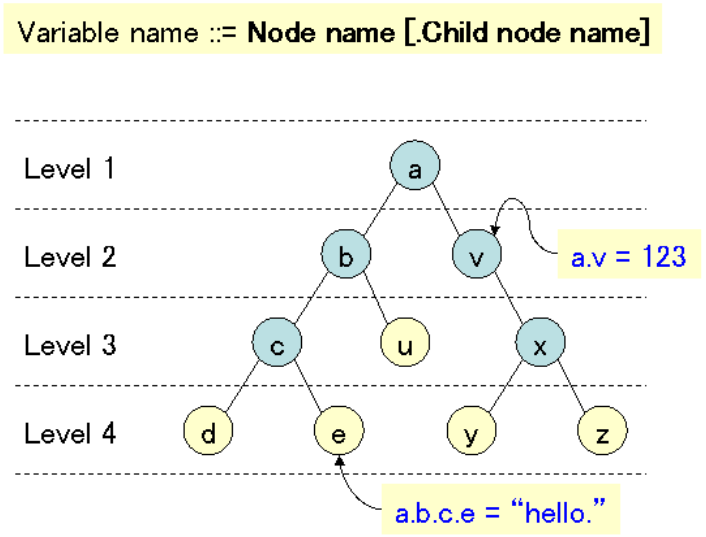

図 3-1 データの階層化例

[複数の黒板の利用]

黒板モデルを使った問題解決システムでは，下記のような理由で複数の黒板を使い分けたい場合があります.

(1) 特定のナレッジソースのみが理解できるデータ(private jargon)を分けたい場合.

(2) ある領域(region)ごとにデータを整理するために黒板を分割したい場合.

(3) 黒板を分けて検索処理を高速化したい場合.

"The Blackboard"プロバイダは，コントローラクラス(CaoController)の一つのオブジェクトに一つの黒板が対応していますので，別名のコントローラオブジェクトを作成するだけで簡単に新しい黒板を作ることができます.

[黒板内のデータの二つの取得方法]

"The Blackboard"プロバイダが保存している黒板データは次の二つの方法で取得することができます.

(1) 必要なときに get_Value()をコールして取得します.

(2) データが変化したときに発生する OnMessage()イベントで取得します.

ここで，(1)の方法は変数クラス(CaoVariable)のオブジェクトを作成して，その公開メンバー関数 get_Value()を呼び出します. この方法は任意のタイミングでデータを取得できます. (サンプルプログラムの BBSource 参照)

一方，(2)の方法はコントローラクラス(CaoController)の OnMessage()イベントで渡されるメッセージクラス(CaoMessage)のオブジェクトからデータを抽出します. この方法はデータの値の変化に同期してデータを取得できます. (サンプルプログラムの BBMonitor 参照)

どちらの方法でも構いませんが，データの値の変化が激しい場合は(2)の方法では大量のイベントが発生してしまいますので注意が必要です.

黒板モデルを使った問題解決システムの場合，一般的にはすべてのデータの値変化を監視したい黒板モニターは(2)の方法で，各ナレッジソースは(1)の方法という使い分けがよいと思われます. もちろん，一つのプログラムの中で二つの方法を使い分けても構いません.

4. サンプルプログラム

ここでは二つの簡単なサンプルプログラムを紹介します.

以下は簡単な黒板モニタープログラムです.

List 4-1 SampleBBMonitor.frm

```
'
' A Simple Blackboard Monitor
'
Dim caoEng As CaoEngine
Dim caoWs As CaoWorkspace
Dim WithEvents caoCtrl As CaoController

' Initialize
Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoWs = caoEng.Workspaces(0)

End Sub

' Create a blackboard
Private Sub cmdConnect_Click()

    Set caoCtrl = caoWs.AddController(txtBB.Text, "CaoProv.Blackboard")

End Sub

' Event sink
Private Sub caoCtrl_OnMessage(ByVal pICaoMess As CAOLib.ICaoMessage)

    On Error GoTo errSkip

    With pICaoMess
        List1.AddItem .Description & " : msgid=" & .Number & ", name=" & .Destination & ", value="
& .Value
    End With

    Exit Sub
errSkip:
    List1.AddItem Err.Description

End Sub
```

以下はナレッジソースをイメージした簡単なプログラムです.

List 4-2 SampleBBSource.frm

```
'
' A Simple Knowledge Source
'
Dim caoEng As CaoEngine
Dim caoWs As CaoWorkspace
Dim WithEvents caoCtrl As CaoController
Dim caoVar As CaoVariable

' Initialize
Private Sub Form_Load()

    Set caoEng = New CaoEngine
    Set caoWs = caoEng.Workspaces(0)

End Sub

' Create a blackboard and data
Private Sub cmdConnect_Click()

    Set caoCtrl = caoWs.AddController(txtBB.Text, "CaoProv.Blackboard")
    Set caoVar = caoCtrl.AddVariable(txtVar.Text)

End Sub

' retrieve data from the blackboard
Private Sub cmdGet_Click()

    txtVal(0).Text = caoVar.Value

End Sub

' write data on the blackboard
Private Sub cmdPut_Click()

    caoVar.Value = txtVal(1).Text

End Sub
```

[実行結果]

図 4-1 にサンプルプログラム実行時のスナップショットを示します. この中で青文字のコメントはプログラム中のオブジェクト名を表しています.

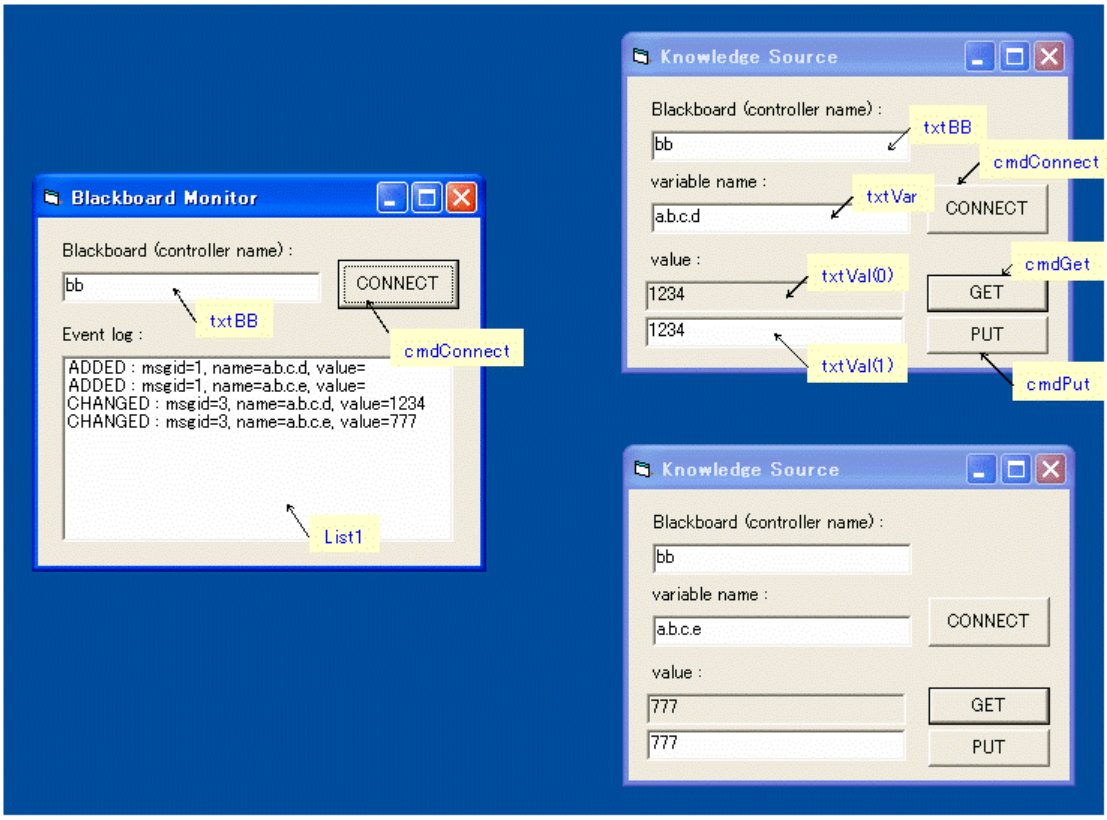

図 4-1 実行結果

5. 参考文献

[1] Daniel D. Corkill, "Blackboard Systems," AI Expert vol. 6, No. 9, pp40-47, 1991.

[2] L. D. Erman et. al., "The Hearsay-II Speech-Understanding system: Integrating Knowledge to Remote Uncertainty," ACM Computing Surveys, Vol. 12, No. 2, 1980.