'The Blackboard' Provider

Blackboard model form data sharing

Version 1.1.0

User's guide

December 18, 2013

[Remarks]

[Revision history]

Version	Date	Content
1.0.0.0	2006-02-23	First edition.
1.0.0.1	2010-02-10	Error code was added
1.0.0	2012-07-17	Version rule of the document was changed.
1.1.0	2013-12-18	Arbitration function was added

[Hardware]

Model	Version	Notes

Contents

1. Introduction

This book is a user's guide of "The Blackboard" provider that makes easier to share the data between ORiN2 applications. This provider is intended to construct the problem solving system with "The Blackboard system". Although ORiN2 can be used as a platform of various systems, this provider enables more smooth application to the problem solving system etc.

This "The Blackboard" provider achieves the data sharing between the ORiN2 applications by storing the variable table internally, and sharing the variable table. One variable table mentioned here means one "Blackboard". Moreover, the shared data can contain not only the simple type data like the integer, the real number, and the character string, etc. but also the binary data of the array and the image, etc.

Furthermore, if this provider is used, not only the data sharing between the applications in the same machine but also the data sharing with the application that is the operation with another machine can be easily achieved. And the combination with CAP provider makes it easy to develop the problem solving system through the internet.

This book explains the function of this "The Blackboard" provider and the mounting method.

[Reference] What is "The blackboard" System?

Blackboard-based problem solving is often presented using the following metaphor:

"Imagine a group of human specialists seated next to a large blackboard. The specialists are working cooperatively to solve a problem, using the blackboard as the workplace for developing the solution. Problem solving begins when the problem and initial data are written onto the blackboard. The specialists watch the blackboard, looking for an opportunity to apply their expertise to the developing solution. When a specialist finds sufficient information to make a contribution, she records the contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved." (Excerpted from bibliography[1])

2. Outline of provider

2.1. Outline

CAO (Controller Access Object) of ORiN2 is an object model where the FA equipment is abstracted. Therefore resource objects can be various types, such as tasks, variables, program files, and robots, though, "The Blackboard" provider is a particular application provider with only "Variable" resources. Internally, if the controller name is different, the same variable name is treated as a different variable because each controller manages the variable data.

The storable data type is VARIANT type that enable to store various data types including binary data of the array and the image data, etc. Even when complex data is shared between two or more applications, it is possible to achieve it easily by using this provider.[1]

As mentioned earlier, this provider is intended to construct the problem solving system with the blackboard system. There is HEARSAYII [2] speech-recognition system as an example of the representative of the problem solving system that handles the blackboard model. HEARSAY II composition is shown in Figure 2-1. The enclosed part with the red dotted line is the interaction area with the blackboard. This provider is utilized for development of this area.

The blackboard monitor and two or more knowledge source correspond to the CAO application when seeing from the model of CAO. Those applications operate the blackboard by operating the object of CAO.

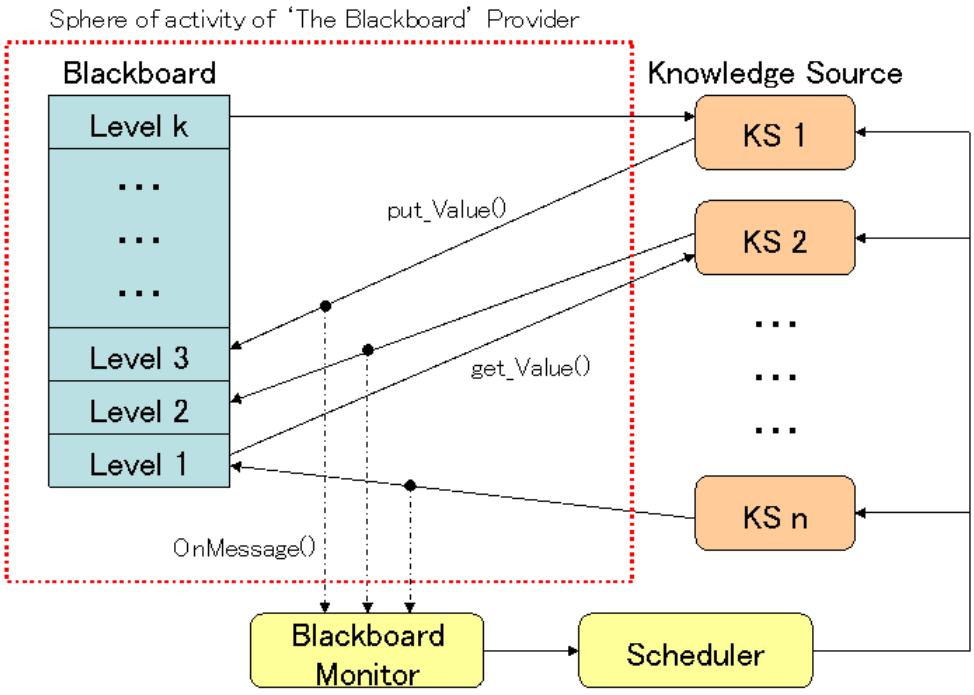

Figure 2-1 HEARSAY II composition

[1] This function is similar to the "DataStore" provider in this point. However, the "DataStore" provider cannot acquire the change of data by using event procedure.

Table 2-1 Blackboard provider

File name	CaoProvBlackboard.dll
ProgID[2]	CaoProv.Blackboard
Registry registration	regsvr32 CaoProvBlackboard.dll
Remove registry registration	regsvr32 /u CaoProvBlackboard.dll

[Function added] Arbitration function (Ver.1.1.0 or higher)

In the Blackboard, user can read from/write in the blackboard freely. This is because the Blackboard model has no supervisor to control the writing to or reading from the Blackboard. With the arbitration function, which is available for the version 1.1.0 or higher, only the arbitrator can control the reading and writing authorities. It means once the arbitrator prohibits the writing, the users will be "read-only" status.

For example, assume that there are two applications: one is to open the window automatically when the temperature rises under the thermo surveillance, and the other is to close the window to prevent from entering robbers or rain by using the surveillance. To coexist with these two applications, using this arbitration function is a practical way to design the system smartly. In this case, needless to say, the application that closes the window is prioritize. So, set this application as an arbitrator.

[2] When installed by ORiN SDK, manual register/deregister is not required.

2.2. Method and Property

2.2.1. CaoWorkspace::AddController method

The Blackboard provider regards one object in the Controller class (CaoController) as a blackboard user (described as "KS" in Figure 2-1). The blackboard mentioned here is a system to control the variable tables, and these variable tables are controlled by each blackboard. To create a new user, use AddController method of the Workspace class (CaoWorkspace).

Specify the name of the blackboard for a controller name of the first argument of the AddController method. The blackboard (variable table) is shared when the specified name already exists, and the blackboard (variable table) is created newly when not existing[3].

Even if the same controller name is specified, when the execution machine name is different by each application, it cannot be shared because the provider is loaded into a different process space.

Moreover, note that the capital letter and the small letter are distinguished as for the controller name (blackboard name). The variable table is deleted at that time when the controller object is deleted if the controller object is not referred by any client programs. The once deleted data is not restorable.

The argument specification of AddController is shown as follows.

```
AddController
(
    "< controller name >"          // Controller name (blackboard name)
    "CaoProv.Blackboard ",  // Provider name. Fixed
    "< machine name >"             // Execution machine name of provider
    "< option >"                   // (unused)
)
```

The following shows the list to specify in the option character string.

Table 2-2 Option character string of CaoWorkspace::AddController

Option	Meaning
Arbitrator=<TRUE/FALSE>	Set the arbitration authority at the participation of the blackboard participants. (Default: FALSE) If this option is set to TRUE while different user has already held the arbitration authority, AddController method will return the error.

The following coding shows an example.

```
AddController
(
    "BB1",                                  // blackboard name = B1
    "CaoProv.Blackboard",
    "",                             // execute by CAO engine process.
    ""
)
```

[3] This description is applied to the case that the same controller name is specified in the different workspace object. To add the same controller object name to the same workspace object results in a CAO specification error.

2.2.2. CaoController::AddVariable method

Register the variable on the Blackboard (variable table) by the AddVariable method of this CaoController class. The added variable is registered in the management table of the CaoController object that offers this method.

The variable is shared when the specified variable name already exists, and the variable is created newly when not existing[4]. Note that the capital letter and the small letter are distinguished as well as the controller name as for the variable name. Moreover, because the variable name that starts by "@" is recognized as a system variable, the user variable that starts by "@" cannot be added.

The value of a variant is VARIANT type which is capable of storing not only simple variable but also the binary data of the array and image data, etc. The argument specification of AddVariable is shown as follows.

```
AddVariable
(
    "< variable name >"     // Variable name registered in variable table.
    "< option >"                    // (unused)
)
```

The following coding shows an example.

```
AddVariable
(
    "ABC",                  // ABC variable is specified.
    ""
)
```

2.2.3. CaoController::Execute method

Execute command against the variable table.

Category	Command name	Function	
Variable control			
	PutItemAttribute	Set an attribute to the specified variable	P.9
	PutItemHelp	Set the help character string to the variable	P.9
Arbitration function			
	TakeArbitrator	Get the arbitration authority	P.10
	ResignArbitrator	Release the arbitration authority	P.10

[4] This description is applied to the case that same variable name is specified by the different controller object. To add the same variable object to the same controller object results in a CAO specification error.

PutItemAttribute

Format

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

Parameters

<Item Name> = VT_I4: Variable name

<Attribute> = VT_I4: Attribute

1	READ	Read only
2	WRITE	Write only
3	READ/WRITE	Readable /Writeable

Return value

None

Explanation

Set an attribute to the specified variable

The behavior of variables and the execution authority of this command differ depending on the existence of the arbitrator.

	Variable behavior	Command execution authority
With arbitrator	Setting value	Arbitrator only
Without arbitrator	READ/WRITE	All users

When the content of setting value has been changed because of the execution of this command, OnMessage event of "STTR_CHANGED" (Number=4) will occur.

PutItemHelp

Format

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

Parameters

<Item Name> = VT_I4: Variable name

<Help> = VT_BSTR: Help character string

Return value

None

Explanation

Set the help character string to the variable specified.

The execution authority of this command differs depending on the existence of the arbitrator, as shown below.

	Command execution authority
With arbitrator	Arbitrator only
Without arbitrator	All users

When the content of setting value has been changed because of the execution of this

command, OnMessage event of "HELP_CHANGED" (Number=5) will occur.

TakeArbitrator

Format ***object*.TakeArbitrator**

Parameters None

Return value None

Explanation Get an arbitration authority.

This command will fail if an arbitrator already exists.

When the arbitration authority is obtained by the execution of this command, OnMessage event of "ARBIT_TAKEN" (Number=9) will occur.

ResignArbitrator

Format ***object*.ResignArbitrator**

Parameters None

Return value None

Explanation Release the arbitration authority to be an standard user.

When the arbitration authority is released by the execution of this command, OnMessage event of "ARBIT_RESIGNED" (Number=8) will occur.

2.2.4. CaoController::get_VariableNames property

In get_VariableNames property, acquire the list of the user variable added by the AddVariable method in the VARIANT type that stores the array of the character string type. The system variable is not included in this list.

2.2.5. CaoController::get_ID property

In this get_ID property, IDs of the blackboard participants are obtained by BSTR type.

2.2.6. CaoController::OnMessage event

In "The Blackboard"provider, the following OnMessage events are issued.

Number	Description	Explanation	
1	ADDED	Variable-added event	P.11
2	DELETED	Variable-deleted event	P.11
3	CHANGED	Variable-changed event	P.12
4	ATTR_CHANGED	Variable's attribute-changed event	P.12
5	HELP_CHANGED	Variable's help character string changed event	P.13
6	CTRL_ADDED	Controller(user)-added event	P.13
7	CTRL_DELETED	Controller(user)-deleted event	P.13
8	ARBIT_RESIGNED	Arbitration authority-released event	P.14
9	ARBIT_TAKEN	Arbitration authority-obtained event	P.14

ADDED

Condition When a new variable has been added to the variable table by the execution of "CaoController::AddVariable"

Number 1

Description ADDED

Destination Variable name

Value Empty

Explanation Variable-added event
Inform the blackboard (variable table) that the variable has been added.
This event occurs only if the added variable name has not been created before.

DELETED

Condition When the variable has been deleted from the variable table by releasing CaoVariable object

Number 2

Description DELETED

Destination Variable name

Value Empty

Explanation Variable-deleted event

Inform the blackboard (variable table) that variable has been deleted.
This event occurs when all CaoVariable objects that refer to the variables in the variable table has been deleted.

CHANGED

Condition When the variable value has been changed by CaoVariable::put_Value()

Number 3

Description CHANGED

Destination Variable name

Value Value specified by CaoVariable::put_Value()

Explanation Variable-changed event
Inform that the variable value of the blackboard (variable table) has been changed.
This event does not occur if the same value as the current value is specified.

ATTR_CHANGED

Condition When the attribute of variable has been changed by PutItemAttribut command of CaoController::Execute()

Number 4

Description ATTR_CHANGED

Destination The blackboard name (Controller name)

Value The second argument of CaoController::Execute() (Parameter)

Explanation Variable's attribute-change event
Inform that the attribute of variable in the blackboard (variable table) has been changed
This event does not occur if the same value as the current value is specified.

HELP_CHANGED

Condition When the help character string of variable has been changed by PutItemHelp command of CaoController::Execute().

Number 5

Description HELP_CHANGED

Destination The blackboard name (Controller name)

Value The second argument of CaoController::Execute() (Parameter)

Explanation Variable's help character string-changed event
Inform that the help character string of variable in the blackboard (variable table) has been changed.
This event does not occur if the same value as the current value is specified.

CTRL_ADDED

Condition When the controller object (the blackboard user) has been added by CaoWorkspace::AddController() execution.

Number 6

Description CTRL_ADDED

Destination The blackboard name (Controller name)

Value Empty

Explanation Controller(user)-added event
Inform that the user of the blackboard (variable table) has been added.

CTRL_DELETED

Condition When the controller (the blackboard user) has been deleted by releasing CaoController.

Number 7

Description CTRL_DELETED

Destination The blackboard name (Controller name)

Value Empty

Explanation Controller (user)-deleted event.
Inform that the user of the blackboard (variable table) has been deleted.

ARBIT_RESIGNED

Condition When the arbitration authority has been released by ResignArbitrator command of CaoController::Execute().

Number 8

Description ARBIT_RESIGNED

Destination The blackboard name (Controller name)

Value The second argument of CaoController::Execute() (Parameter)

Explanation Arbitration authority-released event
Inform that the arbitration authority of the blackboard (variable table) has been released

ARBIT_TAKEN

Condition When the arbitration authority has been obtained by TakeArbitrator command of CaoController::Execute().

Number 9

Description ARBIT_TAKEN

Destination The blackboard name (Controller name)

Value The second argument of CaoController::Execute() (Parameter)

Explanation Arbitration authority-obtained event
Inform that the arbitration authority of the blackboard (variable table) has been obtained.

2.2.7. CaoVariable::get_Value property

Acquire a current value of the variable registered on the blackboard (variable table) in the VARIANT type.

This property returns an error under the condition that the arbitrator has been set and the attribute for reading has not been set to the variable.

2.2.8. CaoVariable::put_Value property

Register the value on the Blackboard (variable table) by the VARIANT type. When this new value is different from the previous value, OnMessage() event of the controller class is issued. An equivalence decision procedure uses VarCmp() function of oleaut32.lib, and refer to API document of Microsoft Corp. for details.

This property returns an error under the condition that the arbitrator has been set and the attribute for writing has not been set to the variable.

2.2.9. CaoVariable::get_Attribute property

Obtain the current attribute value of the variable registered in the Blackboard (variable table).

2.2.10. CaoVariable::get_Help property

Obtain the current help character string of the variable registered in the Blackboard (variable table).

2.3. Variable list

2.3.1. Controller class

In "The Blackboard" provider, system variables on Table2-33 Controller class system variable list are available. Refer to the table below for each meaning.

Table2-3 Controller class system variable list

Variable name	Data type	Explanation	Attribute	
			get	put
@COUNT	VT_I4	Return the number of user variables. The system variable is not included.	√	-
@CURRENT_TIME	VT_DATE	Local time of machine from which provider is executed.	√	-
@VERSION	VT_BSTR	Version.	√	-

2.4. Error code

In "The Blackboard" provider, there is no peculiar error code. Please refer to the chapter of the error code of "ORiN2 Programming guide" for the ORiN2 commonness error.

3. Program design hint

This section introduces the design hint when the blackboard system is constructed by using this provider.

[Layering of data in blackboard]

When the blackboard model is applied to the problem solving system, various data concerning the problem that the system tries to solve is filled in on the blackboard. Generally, being layered comes to treat each knowledge source easily as for those data.

"The Blackboard" Data can be pseudo-layered by devising the variable name though the provider doesn't have a special mechanism for layering data.

For instance, the example of Figure 3-1 delimits by period "." to express the layer. All nodes including the node including the child node can store one value in each node. The data type of the stored data is arbitrary, and the data of a variety of data types can be stored by one tree.

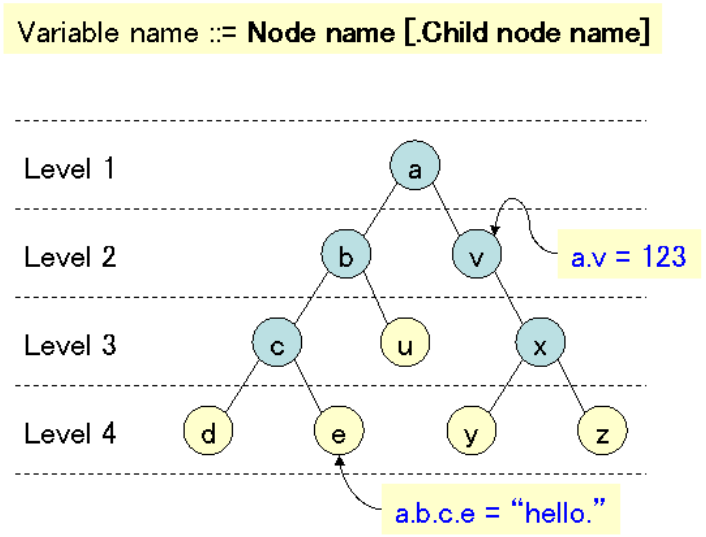

Figure 3-1 Example of layering data

[Use of two or more blackboards]

The problem solving system using the blackboard model is required to use two or more blackboards properly for the following reasons occasionally.

(1) When you would like to divide data (private jargon) to which only a specific knowledge source can be understood.

(2) When you would like to divide the blackboard to arrange data at each certain area (region).

(3) When you would like to speed up the retrieval processing dividing the blackboard.

In this "The Blackboard" provider, an object of the controller class (CaoController) is correspond to each blackboard that makes it easy to create a new blackboard by creating another name's controller object.

[Two methods of acquiring data in blackboard]

The blackboard data which is saved in "The Blackboard" provider can be acquired by following two methods.

(1) Call and acquire get_Value () if it is necessary.

(2) By means of OnMessage () event which issued when data changes.

In the method of (1), create the object of variable class (CaoVariable), and calls the public member function get_Value(). This method enables to acquire data in arbitrary timing. (Refer to BBSource of the sample program.)

On the other hand, the method of (2) extracts data from the object of message class (CaoMessage) passed in the OnMessage() event of controller class (CaoController). This method can acquire data synchronizing with the change of the value of data. (Refer to BBMonitor of the sample program.)

Either method is available though please keep in mind that a large amount of event is issued in the method of (2) when the change of the value of data is intense.

For the problem solving system using the blackboard model, it is recommended to select methods depends on the purpose. Generally, a method of (2) is recommended when the blackboard monitor is required to monitor all the data change. On the other hand, a method of (1) is suitable for monitoring each knowledge source. It is also no problem to use two methods properly in one program.

4. Sample program

There are two simple sample programs here.

The code below is an easy blackboard monitor program.

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
```

The code below is an easy program that images the knowledge source.

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
```

[Execution result]

Figure 4-1 shows a snapshot of the sample program execution. The comment on a blue character shows the object name under the program.

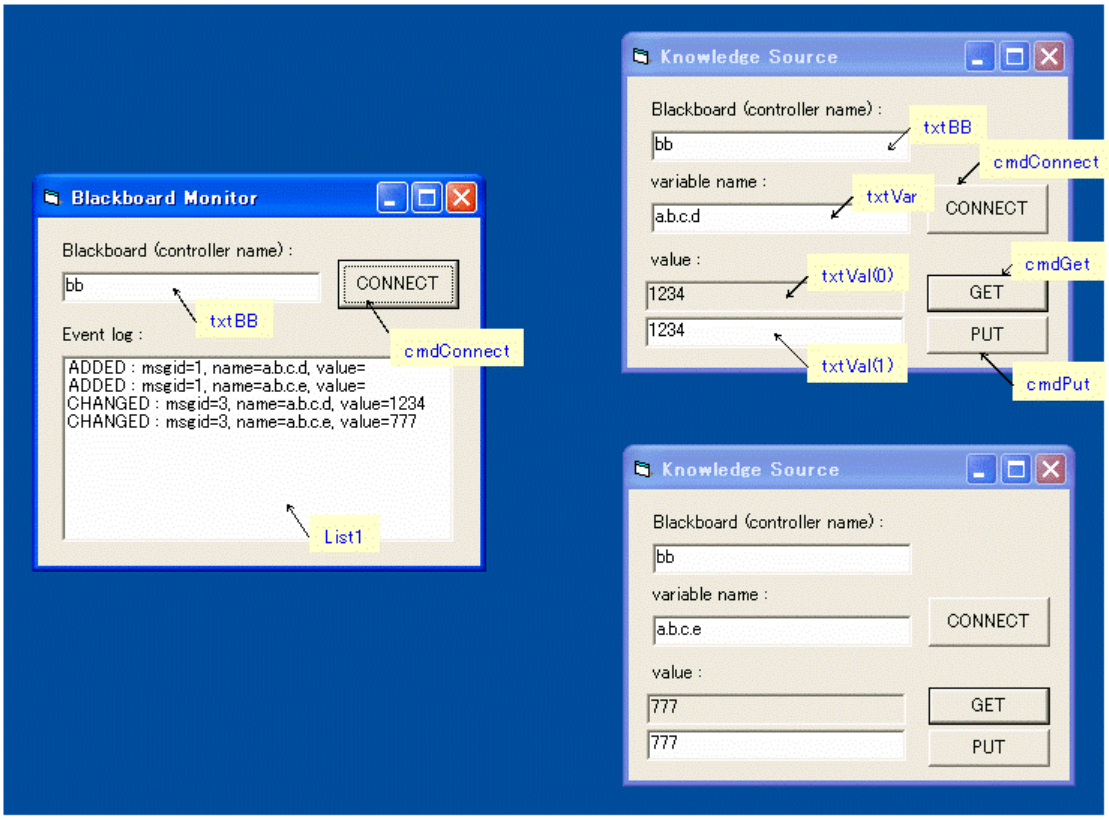

Figure 4-1 Execution result

5. Bibliography

[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.