VE026A
Specifications of b-CAP Communication

User's guide

Version 1.0.0

December 07, 2012

【Remarks】

【Revision history】

Date	E/c level	Content
2012-12-07	1.0.0	First release

Contents

1. Introduction

This specification provides communication protocol of b-CAP.

b-CAP is a protocol which is created following the concept of CAP to improve communication speed. Therefore, b-CAP has the same feature as CAP series, as follows.

(For more detail information about CAP series, Please refer to "CAP provider User's guide" (CAP_ProvGuide_en.doc) included in ORiN2 SDK.)

- It has the same service structure as the object model of CAO provider.
- It calls function by specifying objects by the object ID.
- It provides events of the server by polling.

Command to run the VE026A is the only "slave mode". Slave mode is described in the next chapter.

2. Slave mode

2.1. Slave mode

The Slave mode is a new function to control robot directly by sending a destination position In short interval time. Two command are implemented on VE026A.

2.2. Functions of the Slave mode

2.2.1. slvChangeMode

This function choose the position data type and response style of slvMove.

If “Sync” is chosen, then timeout detector (in each 8msec) is enabled. If “Async” is chosen, then you can switch the timeout detector by the teaching pendant.

If zero is chosen, then the slave mode is stopped.Setup a robot controller with a teaching pendant according to the following procedure.

The following shows the parameters of the "slvChageMode".

Table 2-1 Parameter values of slvChangeMode

Parameter Value	Position data type	Sync/Async
0x002	J	Sync
0x102	J	Async

*When changing to the Slave mode with this command, the change mode operation waits until a robot stops completely. However, if this waiting time exceeds 500 msec, an error 600B [Robot is running] will occur. To avoid this error and the waiting time, it is recommended to choose an @E option when using robot motion commands (e.g. Move/Approach/Drive) before changing to slave-mode.

2.2.2. slvMove

This function send the trajectory data, and the data type is specified by the function “slvChangeMode”.

3. Structure of the packet

3.1. Structure of the packet

The following is the structure of b-CAP message.

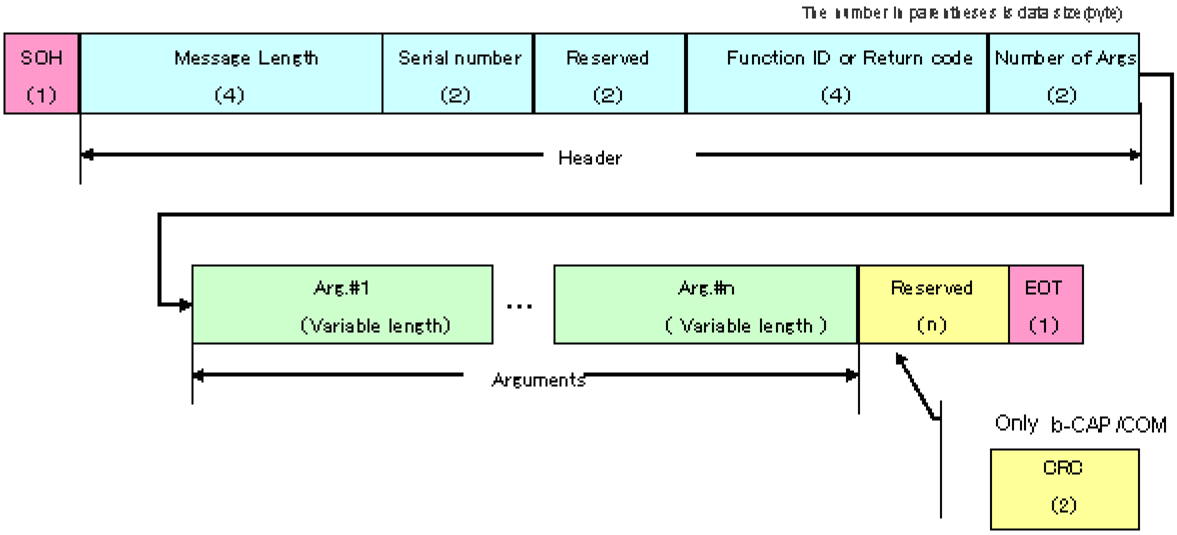


Figure 3-1 b-CAP packet structure

Descriptions of packet elements are listed as follows. The image of each data is stored in Intel format (little endian).

・ Header Code places at the head of packet. SOH (0x01) is used.

・ Message length Data length of the whole message. Unsigned long integer (DWORD) is stored. The length from header to terminator is stored.

・ Serial number Serial number of the message. Unsigned short integer (WORD) is stored.
Range of Serial number is 1 to WORD_MAX(0x0001~0xFFFF).
The values have to be the same for request and respond messages.

・ Reserved area Reserved area in the packet. "0" is always stored in this area.

・ Function ID ID for visited function. Unsigned long integer (DWORD) is stored.
Only used for request messages. (See 3.3)

・ Return Code Code for performance result of visited function.
Unsigned long integer (DWORD) is stored.
Only used for response messages. (See 3.4)

・ The number of arguments The number of arguments for visited function, or number of output variable for visited function. Unsigned short integer (WORD) is stored.

・ Argument #n n-th argument (see 3.2)

- Reservation area　It is an area that has been reserved with the system. This area is variable length area.
- CRC　**It adds only for b-CAP/COM.**

  CRC from header information part CRC to the argument part is stored.

  The calculation condition of CRC is shown below.

CRC type	:	CRC-CCITT
Initial value	:	0xFFFF
Output XOR	:	0x0000
Direction of bit sending	:	Left
Input bit reversing	:	None
Output bit reversing	:	None

- Terminator　End code at the end of packet. EOT (0x04) is used.

3.2. Structure of argument part

Argument part varies in length depending on the data type. It is created to describe several data type.

The following is the structure of argument part.

The number in parentheses is data size(byte)

Argument data length	Data Type	The number of arrays	Data
(4)	(2)	(4)	(Data size)

Figure 3-2 Augument data structure

Argument part includes data type and the number of arrays. The structure of data depends on data type and number of arrays.

The following is description of elements of argument part.

- Argument data length　Total bytes of “Data type”, “The number of arrays” and “Data”. But “Argument data length” is NOT included. Unsigned long integer (DWORD) is stored. See Table 3-1 Data type for data size of each type.
- Data type　Data type of argument. Unsigned short integer (WORD) is stored. See Table 3-1 for enabled data type.
- The number of arrays　The number of arrays in an argument.

  Unsigned Long integer (DWORD) is stored.

  The value is always “1” when VT_ARRAY is not used for data type.
- Data　Data of argument. Varies in size depending on the type.

  See for Table 3-1 each data size. See Figure 3-3 for the structure of information stored in data.

Table 3-1 Data type

Data Type	Value	Size (Byte)	Description
VT_EMPTY	0	0	Empty data
VT_NULL	1	0	NULL value
VT_ERROR	10	2	Error code
VT_UI1	17	1	Binary
VT_I2	2	2	Short integer
VT_UI2	18	2	Unsigned short integer
VT_I4	3	4	Long integer
VT_UI4	19	4	Unsigned long integer
VT_R4	4	4	Single-precision floating point
VT_R8	5	8	Double-precision floating point
VT_CY	6	8	Currency type
VT_DATE	7	8	Date type
VT_BOOL	11	2	Boolean type
VT_BSTR	8	(Number of characters) ×2 + 4	String type String type consists of “String length” and “String data”. ”String data” is stored after “string length” in Unicode, where one character is stored with 2 bytes. “String length” describe the number of byte in “String data”
VT_VARIANT	12	-	Variant type Structure is the same as argument part. Used only with VT_ARRAY.
VT_ARRAY	0x2000	-	Array Data type is determined by logical OR. Data of the specified type is stored in a row.

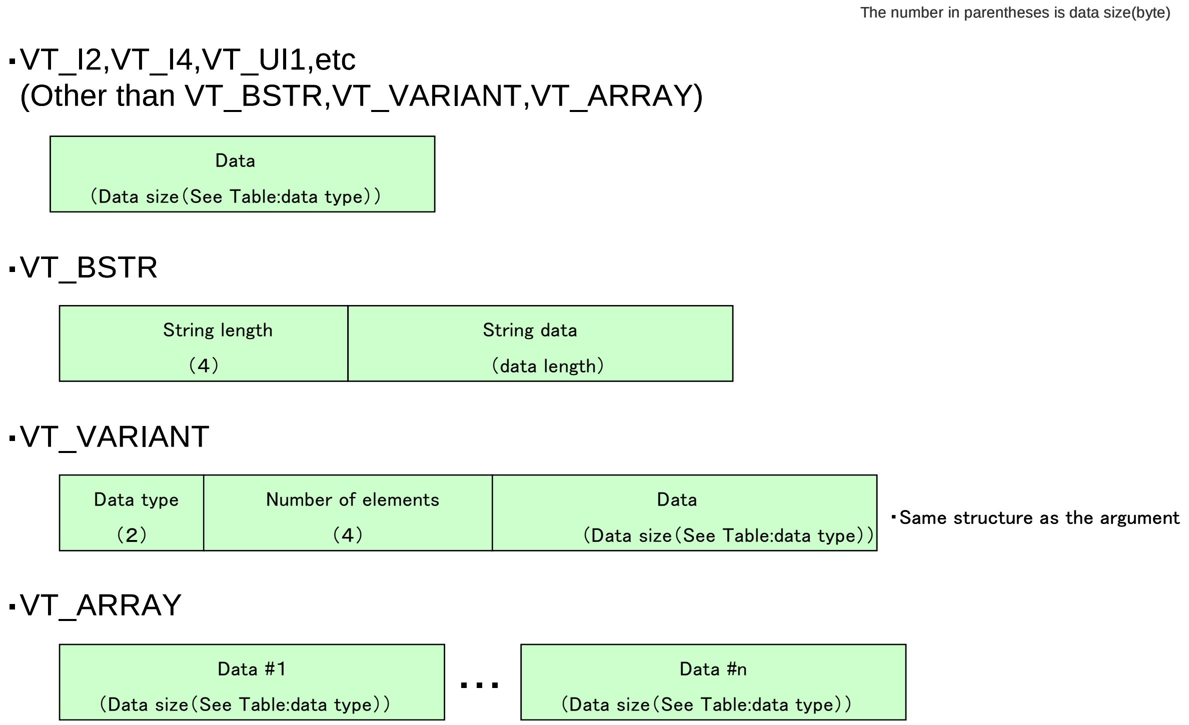

The number in parentheses is data size(byte)
・VT_I2,VT_I4,VT_UI1,etc
(Other than VT_BSTR,VT_VARIANT,VT_ARRAY)
Data
(Data size(See Table:data type))
・VT_BSTR
String length
(4)
String data
(data length)
・VT_VARIANT
Data type
(2)
Number of elements
(4)
Data
(Data size(See Table:data type))
・Same structure as the argument
・VT_ARRAY
Data #1
(Data size(See Table:data type))
・・・
Data #n
(Data size(See Table:data type))

Figure 3–3 Data structure

3.3. Function ID

For b-CAP, function IDs are assigned as follows.

Table 3–2 Function ID assignment

Function ID	Description
1 - 137	For given functions
138 - 255	For reserved area
256 -	For user-defined functions

ID for user-defined functions can be used for functions which are not shown in Table 3-3.

The following is the list of given functions of b-CAP.

Table 3–3 Given functions

Function ID	Function Name	Description
1	Service_Start	Starts the b-CAP service
2	Service_Stop	Stops the b-CAP service
3	Controller_Connect	Connect to a robot controller

4	Controller_Disconnect	Disconnect from the robot controller
7	Controller_GetRobot	Get a robot object
9	Controller_GetVariable	Get a variable object
17	Controller_Execute	Execute the extension function of controller
62	Robot_GetVariable	Get a variable object of the robot
64	Robot_Execute	Execute a command of the robot
84	Robot_Release	Release the robot object
101	Variable_GetValue	Get a value of the variable
111	Variable_Release	Release the variable object

3.4. Return Code

For b-CAP, return codes are assigned as follows.

Table 3-4 Return code assignment

Return code	Description
0x00000000～0x8000FFFF 0x80010000～0x800101FF 0x80070000～0x8007FFFF	For given return codes and reserved area
0x80040200～0x8004FFFF	For user-defined errors

Any code for user-defined errors can be used for errors which are not shown in Table 5.

Table 3-5 Given return codes

Return code	Error	Description
0x00000000	S_OK	OK
0x80004001	E_NOTIMPL	Not implemented
0x80004004	E_ABORT	Function aborted
0x80004005	E_FAIL	Function failed
0x8000FFFF	E_UNEXPECTED	Fatal Error occurred
0x80010001	E_INVALIDRCVPACKET	Invalid packet is received. When this error is occurred, robot disconnect from client immediately. Please make sure the packet that you sent.
0x80010002	E_INVALIDSNDPACKET	Invalid packet is sent

0x80010003	E_INVALIDARGTYPE	Invalid argument type
0x80010004	E_ROBOTISBUSY	Robot is busy (Wait for a while)
0x80010005	E_INVALIDCOMMAND	Invalid command string is received
0x80010011	E_PACKETSIZEOVER	Received packet size over (> 16Mbytes)
0x80010012	E_ARGSIZEOVER	An argument siez over of the received packet. (> 16Mbytes)
0x80070005	E_ACCESSDENIED	Access denied
0x80070006	E_HANDLE	Invalid handle
0x8007000E	E_OUTOFMEMORY	Out of memory
0x80070057	E_INVALIDARG	Invalid argument

4. Communication procedure

4.1. Communication sequence

Sequence of b-CAP begins with sending a request packet from a client.

The robot performs the request packet function and sends a response packet to the client.

There is not regulations for time from the reception of the demand packet on the robot side to the reply of the answer either. Therefore, it is necessary to note it because the time-out will be generated on the client side when the time-out detection time of the client is short when the processing time of the robot side is long.

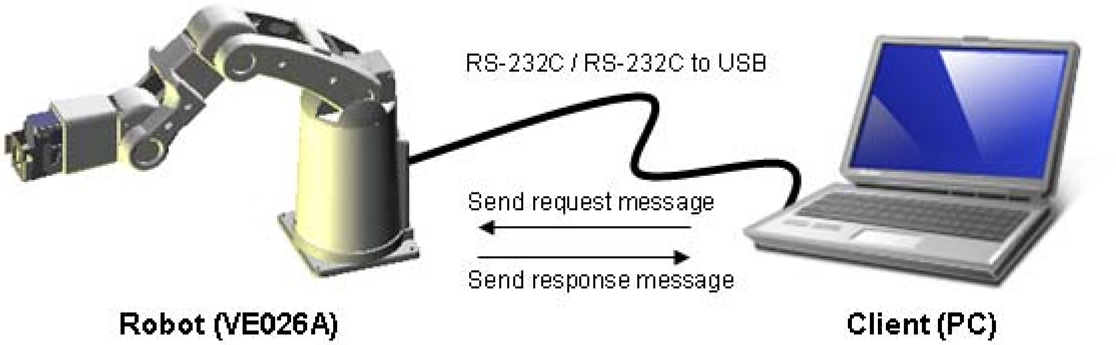


Figure 4-1 Communication sequence

4.2. Communication procedure for robot(VE026A)

The following is the outline of communication procedure.

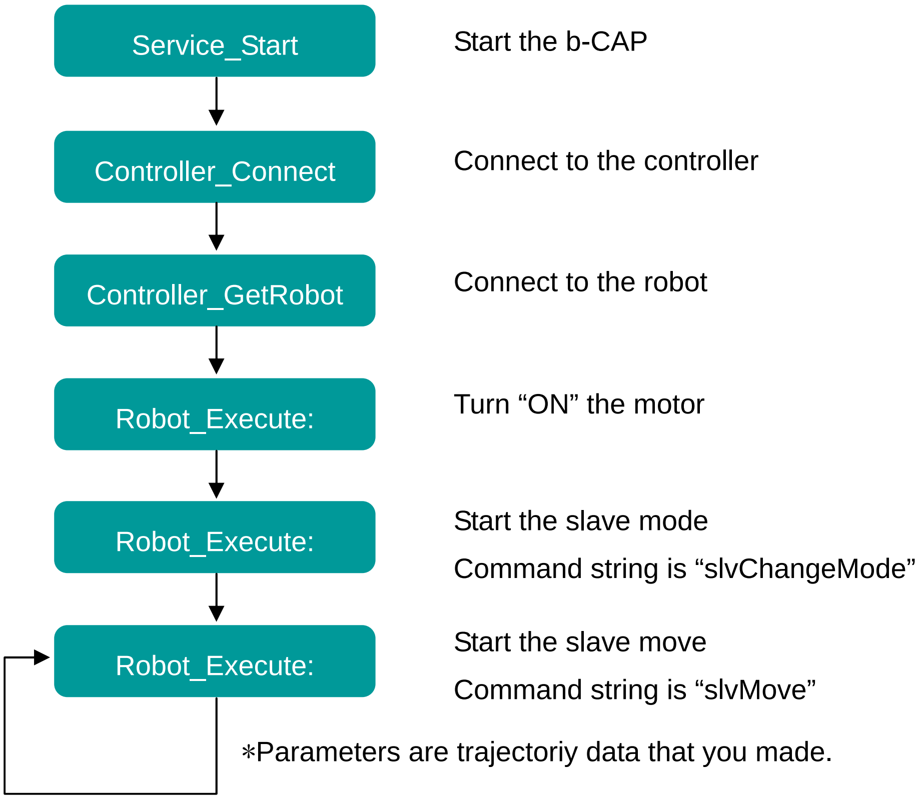


Figure 4–2 Communication procedure

Client establishes a session by connecting to the robot, then sends a request message of performed function and waits for a response message from the robot.

Client needs to perform time-out processing when no response comes from the robot. However, response time varies depending on job, therefore, attention needs to be paid to setting the time-out period.

4.3. Err Clear

When some function returns a error,Clear the error in the following sequence.

*If Return code NOT EQUAL S_OK (S_OK = 0x0),then this means error has been occurred.

Please run the "ClearError" of Controller_Execute. Then, clear the error, please return to the slave mode again.

5. b-CAP communication function

5.1. Start and stop of b-CAP service

5.1.1. Service_Start

Function	HRESULT Service_Start()
Function ID	1
Argument	No argument
Return Value	See Table 3-5 Given return codes
Description	Starts b-CAP service
See also	Service_Stop

<table>
<tr><td colspan="5">Communication sample 1</td></tr>
<tr><td colspan="5">This sample procedure starts the b-CAP server.</td></tr>
<tr><td rowspan="5">Packet
TX</td><td colspan="4">TX(Client->Server):
01 12 00 00 00 01 00 00   00 01 00 00 00 00 00 BA
B3 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">No-Args</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">-</td></tr>
<tr><td rowspan="5">Packet
RX</td><td colspan="4">RX(Server->Client):
01 12 00 00 00 01 00 00   00 00 00 00 00 00 00 1A
F6 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">No-Args</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">-</td></tr>
</table>

5.1.2. Service_Stop

Function	HRESULT Service_Stop ()
Function ID	2
Argument	No argument
Return Value	See Table 3-5 Given return codes
Description	Stops b-CAP service
See also	Service_Stop

Communication sample 1				
This sample procedure stops the b-CAP server.				
Packet TX	TX(Client->Server): 01 12 00 00 00 07 00 00 00 02 00 00 00 00 00 E5 0E 04			
	Name	Description	Type	Value
		Binary		
	No-Args	-	-	-
		-		
Packet RX	RX(Server->Client): 01 12 00 00 00 07 00 00 00 00 00 00 00 00 00 A5 85 04			
	Name	Description	Type	Value
		Binary		
	No-Args	-	-	-
		-		

5.2. Controller objects

5.2.1. Controller_Connect

Function	HRESULT Controller_Connect ()			
Function ID	3			
Argument	[in]	BSTR	bstrCtrlName	Controller name(Not used)
	[in]	BSTR	bstrProvName	Provider name (Not used)
	[in]	BSTR	bstrPcName	Provider execution machine name(Not used)
	[in]	BSTR	bstrOption	Option character string = "<option1>, <option2>, …" (Not used)
	[out]	long	hController	Handle of controller
Return Value	See Table 3-5 Given return codes			
Description	Gets the handle of the controller object "hController" by connecting to the controller			
See also	Controller_Disconnect			

Communication sample 1				
This function returns the handle of the controller - hCOntroller.				
Packet TX	TX(Client->Server): 01 4A 00 00 00 03 00 00 00 03 00 00 00 04 00 0A 00 00 00 08 00 01 00 00 00 00 00 00 00 0A 00 00 00 08 00 01 00 00 00 00 00 00 00 0A 00 00 00 08 00 01 00 00 00 00 00 00 00 0A 00 00 00 08 00 01 00 00 00 00 00 00 00 FA EB 04			
	Name	Description	Type	Value
		Binary		
	bstrCtrlName	The name of the controller. (Not used)	VT_BSTR	Null String
		0A 00 00 00 08 00 01 00 00 00 00 00 00 00 0A 00 00 00 08 00 01 00 00 00		
	bstrProvName	The name of the provider. (Not used)	VT_BSTR	Null String
		00 00 00 00 0A 00 00 00 08 00 01 00 00 00		
	bstrPcName	The name of the client PC. (Not used)	VT_BSTR	Null String
		00 00 00 00 0A 00 00 00 08 00 01 00 00 00		
	bstrOption	The connecting option. (Not used)	VT_BSTR	Null String
		00 00 00 00		
Packet RX	RX(Server->Client): 01 20 00 00 00 03 00 00 00 00 00 00 00 01 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 BB 12 04			
	Name	Description	Type	Value
		Binary		
	hController	The handle of controller	VT_I4	0x000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		

5.2.2. Controller_Disconnect

Function	HRESULT Controller_Disconnect ()			
Function ID	4			
Argument	[in]	long	hController	Handle of controller
Return Value	See Table 3-5 Given return codes			
Description	Disconnects from the controller specified by "hController"			
See also	Controller_Connect			

<table>
<tr><th colspan="5">Communication sample 1</th></tr>
<tr><td colspan="5">This function disconnects from the controller that indicated by handle of controller - hCOntroller.</td></tr>
<tr><td rowspan="5">Packet
TX</td><td colspan="4">TX(Client->Server):
01 20 00 00 00 05 00 00   00 04 00 00 00 01 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 7D 76 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">hController</td><td>The handle of the controller
(See also Controller_Connect)</td><td>VT_I4</td><td>0x0000001</td></tr>
<tr><td colspan="3">0A
00 00 00 03 00 01 00 00   00 01 00 00 00 7D</td></tr>
<tr><td rowspan="5">Packet
RX</td><td colspan="4">RX(Server->Client):
01 12 00 00 00 05 00 00   00 00 00 00 00 00 00 2F
5B 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">No-Args</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="3">-</td></tr>
</table>

5.2.3. Controller_GetVariable

Function	HRESULT Controller_GetVariable ()			
Function ID	9			
Argument	[in]	long	hController	Handle of controller
	[in]	BSTR	bstrName	Variable name
	[in]	BSTR	bstrOption	Option character string
	[out]	long	hVariable	Handle of variable
Return Value	See Table 3-5 Given return codes			
Description	Gets the handle of the controller system variable object "hVariable"			

By using this handle, The application software can accesses to resources of the variable.

See also Variable_GetValue
Robot_GetVariable

In this function, variables in controller can be obtained by using the following variable names for "bstrName"

Table 5-1 User-accessible variables for controller class

Variable Name	Data type	Description	Attribute	
			get	put
@VERSION	VT_BSTR	Version string of the robot controller	∨	-

Communication sample 1
This function returns the handle of the variable - hVariable. By using this hanlde, application can access to variable.

Packet TX

TX(Client->Server):

```
01 4C 00 00 00 09 00 09   00 09 00 00 00 03 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 1A 00 00
00 08 00 01 00 00 00 10   00 00 00 40 00 56 00 45
00 52 00 53 00 49 00 4F   00 4E 00 0A 00 00 00 08
00 01 00 00 00 00 00 00   00 85 80 04
```

Name	Description	Type	Value
	Binary		
hController	The handle of the controller (See also Controller_Connect)	VT_I4	0x0000001
	0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
bstrName	The variable name	VT_BSTR	"I1"
	1A 00 00 00 08 00 01 00 00 00 10 00 00 00 40 00 56 00 45 00 52 00 53 00 49 00 4F 00 4E 00		
bstrOption	bstrOption	VT_BSTR	Null String
	0A 00 00 00 08 00 01 00 00 00 00 00 00 00 85 80 04		

Packet RX

RX(Server->Client):

```
01 20 00 00 00 09 00 09   00 00 00 00 00 01 00 0A
00 00 00 03 00 01 00 00   00 04 00 00 00 BA 5B 04
```

Name	Description	Type	Value
	Binary		

	hVariable	The handle of variable "I1" is returned.	VT_I4	0x00020001
		0A 00 00 00 03 00 01 00 00 00 04 00 00 00		

5.2.4. Controller_Execute

Function HRESULT Controller_Execute()

Function ID 17

Argument	[in]	long	hController	Handle of controller
	[in]	BSTR	bstrCommand	Command name
	[in]	VARIANT	vntParam	Parameter
	[out]	long	pVal	Result value

Return Value See Table 3-5 Given return codes

Description Execute the command of the controller "hController"

In this function, commands can be executed by using the following command names for "bstrCommand".

Table 5-2 Implemented command list

Command	Parameter	Return value	Operation
ClearError	:VT_I4: ErrorCode	VT_I2: Result Code	Clear error Note that "ClearError" is time-consuming command. It may take about 1 sec.

<table>
<tr><th colspan="5">Communication sample 1</th></tr>
<tr><td colspan="5">This function executes a command of a controller.
In this sample, "ClearError" is executed.</td></tr>
<tr><td rowspan="4">Packet
TX</td><td colspan="4">TX(Client->Server):
01 4E 00 00 00 0B 00 0B   00 11 00 00 00 03 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 1E 00 00
00 08 00 01 00 00 00 14   00 00 00 43 00 6C 00 65
00 61 00 72 00 45 00 72   00 72 00 6F 00 72 00 08
00 00 00 02 00 01 00 00   00 00 00 47 BB 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td>hController</td><td>The handle of the controller
(See also Controller_Connect)</td><td>VT_I4</td><td>0x0000001</td></tr>
</table>

		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrCommand	The command string	VT_BSTR	“ClearError”
		1E 00 00 00 08 00 01 00 00 00 14 00 00 00 43 00 6C 00 65 00 61 00 72 00 45 00 72 00 72 00 6F 00 72 00		
	vntParam	The Error code that should be cleared. (In RC7, This value is not used, please use 0x00000000)	VT_I4	0x00000000
		08 00 00 00 02 00 01 00 00 00 00 00		
Packet RX	RX(Server->Client): 01 12 00 00 00 0B 00 0B 00 00 00 00 00 00 00 74 B4 04			
	Name	Description	Type	Value
		Binary		
	Result	-	-	-
		−		

5.3. Robot objects

5.3.1. Controller_GetRobot

Function	HRESULT Controller_GetRobot ()			
Function ID	7			
Argument	[in]	long	hController	Handle of controller
	[in]	BSTR	bstrName	Name of robot
	[in]	BSTR	bstrOption	Option character string
	[out]	long	hRobot	Handle of robot
Return Value	See Table 3-5 Given return codes			
Description	Get the handle of robot object “hRobot” By using this handle, The application software can accesses to resources of the robot.			
See also	Controller_Connect			

Communication sample 1				
This function returns the handle of robot - hRobot.				
Packet TX	TX(Client->Server): 01 48 00 00 00 10 00 10 00 07 00 00 00 03 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 16 00 00 00 08 00 01 00 00 00 0C 00 00 00 56 00 45 00 30 00 32 00 36 00 41 00 0A 00 00 00 08 00 01 00 00 00 00 00 00 00 B8 90 04			
	Name	Description	Type	Value
		Binary		
	hController	The handle of the controller (See also Controller_Connect)	VT_I4	0x0000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrName	The name of the robot (In RC7, This value is not used)	VT_BSTR	Null String
		16 00 00 00 08 00 01 00 00 00 0C 00 00 00 56 00 45 00 30 00 32 00 36 00 41 00		
	bstrOption	bstrOption	VT_BSTR	Null String
		0A 00 00 00 08 00 01 00 00 00 00 00 00 00		
Packet RX	RX(Server->Client): 01 20 00 00 00 10 00 10 00 00 00 00 00 01 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 AB B6 04			
	Name	Description	Type	Value
		Binary		
	hRobot	The handle of the robot	VT_I4	0x000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		

5.3.2. Robot_Release

Function	HRESULT Robot_Release ()			
Function ID	84			
Argument	[in]	long	hRobot	Handle of robot
Return Value	See Table 3-5 Given return codes			
Description	Release the robot objets specified by "hRobot"			

5.3.3. Robot_GetVariable

Function HRESULT Robot_GetVariable ()x

Function ID 62

Argument	[in]	long	hRobot	Handle of robot
	[in]	BSTR	bstrName	Variable name
	[in]	BSTR	bstrOption	Option character string
	[out]	long	hVariable	Handle of variable

Return Value See Table 3-5 Given return codes

Description Get the handle of robot system variable object "hVariable"

By using this handle, The application software can accesses to resources of the variable.

See also Variable_GetValue

Controller_GetVariable

In this function, information on robots can be obtained by using the following variable names for "bstrName".

Table 5-3 Robot class system variable list

Variable Name	Data type	Description	Attribute	
			get	put
@CURRENT_ANGLE	VT_ARRAY | VT_R4	Current position of the robot （Angle of each axis）	∨	-
@SERVO_ON	VT_I2	Servo state, 0=OFF,1= ON	∨	-
@TYPE	VT_I4	Robot type	∨	-

Communication sample 1

This function returns the handle of variable - hVariable.

Packet TX

TX(Client->Server):

```
01 58 00 00 00 13 00 13   00 3E 00 00 00 03 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 26 00 00
00 08 00 01 00 00 00 1C   00 00 00 40 00 43 00 55
00 52 00 52 00 45 00 4E   00 54 00 5F 00 41 00 4E
00 47 00 4C 00 45 00 0A   00 00 00 08 00 01 00 00
00 00 00 00 00 73 F8 04
```

Name	Description	Type	Value
	Binary		
hRobot	The handle of robot (See also Controller_GetRobot)	VT_I4	0x0000001

		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrName	The name of the varibale	VT_BSTR	"@CURRENT_ANGLE"
		26 00 00 00 08 00 01 00 00 00 1C 00 00 00 40 00 43 00 55 00 52 00 52 00 45 00 4E 00 54 00 5F 00 41 00 4E 00 47 00 4C 00 45 00		
	bstrOption	Option	VT_BSTR	Null String
		0A 00 00 00 08 00 01 00 00 00 00 00 00 00		
Packet RX	RX(Server->Client): 01 20 00 00 00 13 00 13 00 00 00 00 00 01 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 C6 59 04			
	Name	Description	Type	Value
		Binary		
	hVariable	The handle of the variable	VT_I4	0x300000
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		

5.3.4. Robot_Execute

Function	HRESULT Robot_Execute()			
Function ID	64			
Argument	[in]	long	hRobot	Handle of robot
	[in]	BSTR	bstrCommand	Command name
	[in]	VARIANT	vntParam	Parameter
	[out]	VARIANT	pVal	Result
Return Value	See Table 3-5 Given return codes			
Description	Execute the command of the robot "hRobot"			

In this function, commands can be executed by using the following command names for "bstrCommand".

Table 5-3 Implemented command list

Command	Parameter	Return value	Operation
Motor	VT_I2: State(1:ON / 0:OFF)	None	Motor ON/OFF Note that "Motor ON/OFF" is time-consuming command. It may take

			about 3 sec.
slvChangeMode	VT_I2 or VT_I4: 0x0 : Stop SlaveMode 0x2 : Synchronization (J type) 0x102 : Asynchronization (J type)	None	Switch the Slave Mode. Note: When switch to the Slave mode, this command wait until stop of the robot motion. (The b-CAP Slave function)
slvMove	VT_R4|ARRAY : P/J/T type Or VT_R8|ARRAY : P/J/T type	<Current position(J type): VT_R4|ARRAY>	Execute "SlaveMove". The type of this argument (P/J/T) is specified by the "SlvChangeMode". (The b-CAP Slave function)

Communication sample 1 – Motor ON/OFF				
This command turns on or off the motor.				
Packet TX	TX(Client->Server): 01 44 00 00 00 16 00 16 00 40 00 00 00 03 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 14 00 00 00 08 00 01 00 00 00 0A 00 00 00 4D 00 6F 00 74 00 6F 00 72 00 08 00 00 00 02 00 01 00 00 00 01 00 D4 10 04			
	Name	Description	Type	Value
		Binary		
	hRobot	The handle of robot (See also Controller_GetRobot)	VT_I4	0x0000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrCommand	Command string	VT_BSTR	"MOTOR"
		14 00 00 00 08 00 01 00 00 00 0A 00 00 00 4D 00 6F 00 74 00 6F 00 72 00		
	vntParam	Parameter of command	VT_I2	0x0001
		08 00 00 00 02 00 01 00 00 00 01 00		
Packet RX	RX(Server->Client): 01 12 00 00 00 16 00 16 00 00 00 00 00 00 00 09 C3 04			
	Name	Description	Type	Value

		Binary		
	No-Args	-	-	-
		-		

Communication sample 2 – slvChangeMode				
This sample execute “slvChangeMode” command　: slvChangeMode 0x102				
Packet TX	TX(Client->Server): 01 56 00 00 00 17 00 17 00 40 00 00 00 03 00 0A 00 00 00 03 00 01 00 00 00 01 00 00 00 24 00 00 00 08 00 01 00 00 00 1A 00 00 00 73 00 6C 00 76 00 43 00 68 00 61 00 6E 00 67 00 65 00 4D 00 6F 00 64 00 65 00 0A 00 00 00 03 00 01 00 00 00 02 01 00 00 49 AC 04			
	Name	Description	Type	Value
		Binary		
	hRobot	The handle of robot (See also Controller_GetRobot)	VT_I4	0x0000001
		0A 00 00 00 03 00 01 00 00 00 01 00 00 00		
	bstrCommand	Command string	VT_BSTR	“slvChangeMode”
		24 00 00 00 08 00 01 00 00 00 1A 00 00 00 73 00 6C 00 76 00 43 00 68 00 61 00 6E 00 67 00 65 00 4D 00 6F 00 64 00 65 00		
	vntParam	Option parameter	VT_I2 or VT_I4	0x102
		0A 00 00 00 03 00 01 00 00 00 02 01 00 00		
Packet RX	RX(Server->Client): 01 12 00 00 00 17 00 17 00 00 00 00 00 00 00 9F EB 04			
	Name	Description	Type	Value
		Binary		
	pVal	-	-	-
		–		

Communication sample 3 – slvMove
This sample execute “slvMove” command　: SlvMove 0,30,120,0,30,0,25,0

<table>
<tr><td rowspan="9">Packet
TX</td><td colspan="4">TX(Client->Server):
01 66 00 00 00 46 00 46   00 40 00 00 00 03 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 18 00 00
00 08 00 01 00 00 00 0E   00 00 00 73 00 6C 00 76
00 4D 00 6F 00 76 00 65   00 26 00 00 00 04 20 08
00 00 00 00 00 00 00 00   00 F0 41 00 00 F0 42 00
00 00 00 00 00 F0 41 00   00 00 00 00 00 C8 41 00
00 00 00 E8 24 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">hRobot</td><td>The handle of robot
(See also Controller_GetRobot)</td><td>VT_I4</td><td>0x0000001</td></tr>
<tr><td colspan="3">0A
00 00 00 03 00 01 00 00   00 01 00 00 00</td></tr>
<tr><td rowspan="2">bstrCommand</td><td>Command string</td><td>VT_BSTR</td><td>“slvMove”</td></tr>
<tr><td colspan="3">18 00 00
00 08 00 01 00 00 00 0E   00 00 00 73 00 6C 00 76
00 4D 00 6F 00 76 00 65   00</td></tr>
<tr><td rowspan="2">vntParam</td><td>Option parameter</td><td>VT_R4 |
ARRAY</td><td>0,30,120,0,30,0,25,0</td></tr>
<tr><td colspan="3">26 00 00 00 04 20 08
00 00 00 00 00 00 00 00   00 F0 41 00 00 F0 42 00
00 00 00 00 00 F0 41 00   00 00 00 00 00 C8 41 00
00 00 00</td></tr>
<tr><td rowspan="5">Packet
RX</td><td colspan="4">RX(Server->Client):
01 3C 00 00 00 46 00 46   00 00 00 00 00 01 00 26
00 00 00 04 20 08 00 00   00 00 00 00 00 32 33 F7
41 66 66 F1 42 CC CC CC   3D 66 66 EE 41 CC CC 4C
3E 66 66 C6 41 00 00 00   00 B4 5B 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">pVal</td><td>Current position(J type)</td><td>VT_R4 |
ARRAY</td><td></td></tr>
<tr><td colspan="3">26
00 00 00 04 20 08 00 00   00 00 00 00 00 32 33 F7
41 66 66 F1 42 CC CC CC   3D 66 66 EE 41 CC CC 4C
3E 66 66 C6 41 00 00 00   00</td></tr>
</table>

5.4. Variable object

Variable object allows to read variables.

5.4.1. Variable_Release

Function	HRESULT Variable_Release ()			
Function ID	111			
Argument	[in]	long	hVar	Handle of Variable
Return Value	See Table 3-5 Given return codes			
Description	Releases the variable object specified by "hVar"			

5.4.2. Variable_GetValue

Gets the value of a variable that specified by the handle of the variable.

Function	HRESULT Variable_GetValue ()			
Function ID	101			
Argument	[in]	long	hVariable	Handle of variable
	[out]	VARIANT	pVal	Value
Return Value	See Table 3-5 Given return codes			
Description	Gets the value of the variable object specified by hVariable			
See also	Controller_GetVariable			
	Robot_GetVariable			

<table>
<tr><td colspan="5">Communication sample 1 – Variable_GetValue ("@CURRENT_ANGLE")</td></tr>
<tr><td colspan="5">This sample procedure gets the value of robot varibale "@CURRENT_ANGLE"</td></tr>
<tr><td rowspan="5">Packet
TX</td><td colspan="4">TX(Client->Server):
01 20 00 00 00 47 00 47   00 65 00 00 00 01 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 1D 93 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
<tr><td rowspan="2">HVariable</td><td>The handle of the variable</td><td>VT_I4</td><td>0x30020B</td></tr>
<tr><td colspan="3">0A
00 00 00 03 00 01 00 00   00 01 00 00 00</td></tr>
<tr><td rowspan="3">Packet
RX</td><td colspan="4">RX(Server->Client):
01 3C 00 00 00 47 00 47   00 00 00 00 00 01 00 26
00 00 00 04 20 08 00 00   00 CC CC CC 3D 32 33 F7
41 66 66 F1 42 CC CC CC   3D 66 66 EE 41 CC CC 4C
3E 99 99 C5 41 00 00 00   00 3E 8C 04</td></tr>
<tr><td rowspan="2">Name</td><td>Description</td><td>Type</td><td>Value</td></tr>
<tr><td colspan="3">Binary</td></tr>
</table>

	pVal	Current position(J type) with time stamp	VT_R4|ARRAY	9.99E-02,30.9,120.7, 9.99E-02,29.8,0.2, 24.7,0
		26 00 00 00 04 20 08 00 00 00 CC CC CC 3D 32 33 F7 41 66 66 F1 42 CC CC CC 3D 66 66 EE 41 CC CC 4C 3E 99 99 C5 41 00 00 00 00		

6. b-CAP Tester

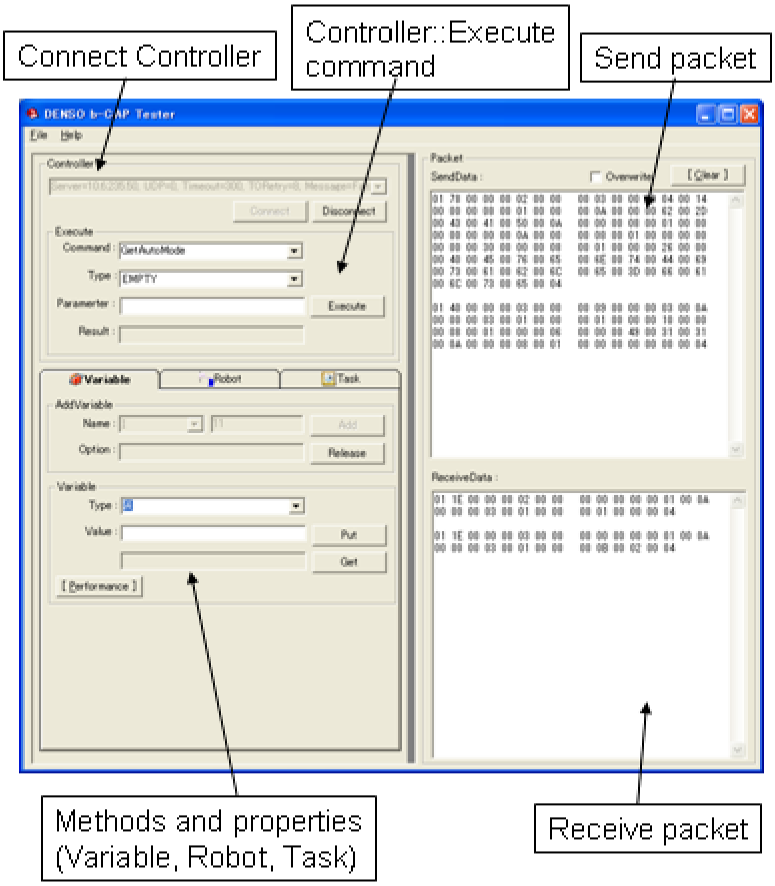
6.1. b-CAP Tester

b-CAP tester is a testing tool for sending and receiving b-CAP packet.

b-CAP tester is in the following folder:

ORiN2¥C AP¥b-CAP¥CapLib¥DENSO¥Bin

Functions of b-CAP tester are like bellow. (See Fig6-1)



Controller: Server=10.6.235.50, UDP=0, Timeout=300, TORetry=8, Message=F

Command : GetAutoMode

Type : EMPTY

Parameter :

Result :

Name :

Option :

Type :

Value :

SendData :

```
01 78 00 00 00 02 00 00   00 03 00 00 00 04 00 14
00 00 00 00 00 01 00 00   00 0A 00 00 00 62 00 2D
00 43 00 41 00 50 00 0A   00 00 00 00 00 01 00 00
00 00 00 00 00 0A 00 00   00 00 00 01 00 00 00 00
00 00 00 30 00 00 00 00   00 01 00 00 00 26 00 00
00 48 00 45 00 76 00 65   00 6E 00 74 00 44 00 69
00 73 00 61 00 62 00 6C   00 65 00 3D 00 66 00 61
00 6C 00 73 00 65 00 04

01 40 00 00 00 03 00 00   00 09 00 00 00 03 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 10 00 00
00 00 00 01 00 00 00 08   00 00 00 49 00 31 00 31
00 0A 00 00 00 00 00 01   00 00 00 00 00 00 00 04
```

ReceiveData :

```
01 1E 00 00 00 02 00 00   00 00 00 00 00 01 00 0A
00 00 00 03 00 01 00 00   00 01 00 00 00 04

01 1E 00 00 00 03 00 00   00 00 00 00 00 01 00 0A
00 00 00 03 00 01 00 00   00 0B 00 02 00 04
```

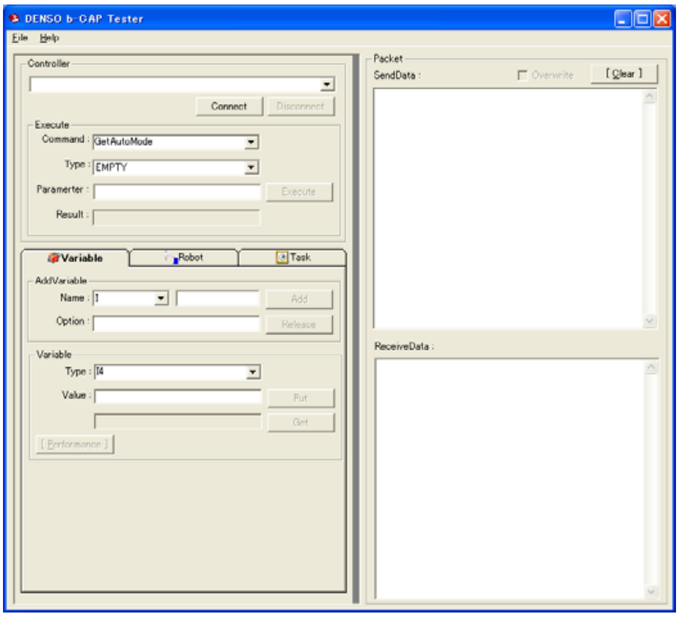
Figure 6-1 Function introduction of b-CAP Tester

6.2. Checking b-CAP communication with b-CAP Tester.

After connecting PC and VE026A, b-CAP communication can be checked with b-CAP Tester in the following procedure.

(1) Start b-CAP Tester.

※The program is installed at ORiN2¥CAP¥b-CAP¥CapLib¥DENSO¥Bin¥b-CAP Tester.exe.



Command : GetAutoMode

Type : EMPTY

Parameter :

Result :

Name : I

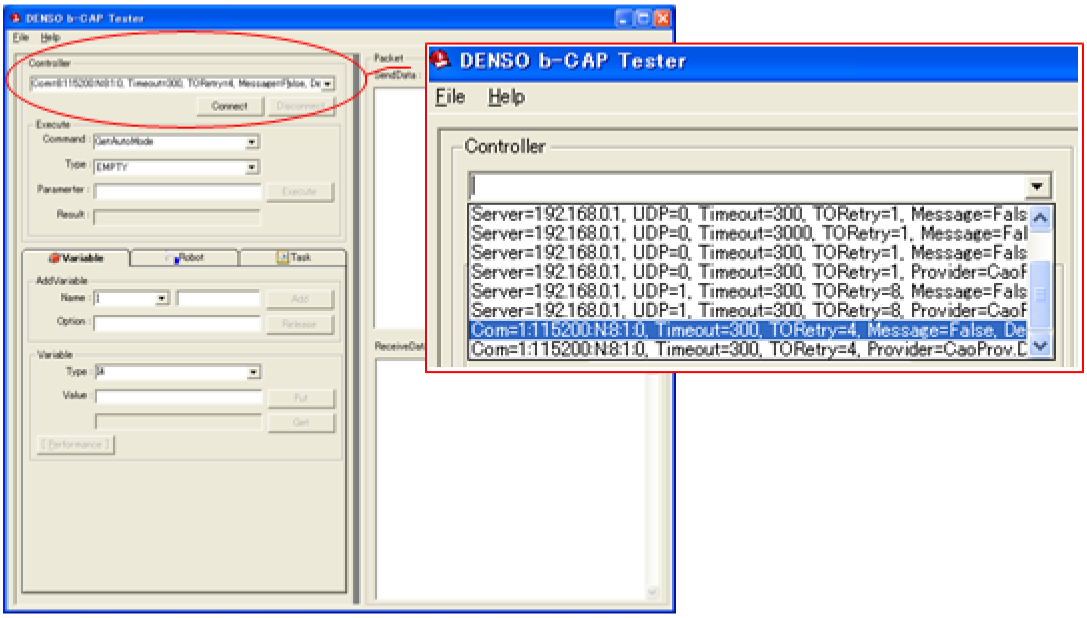
Option :

Type : I4

Value :

図 6-2 Initial screen of b-CAP Tester

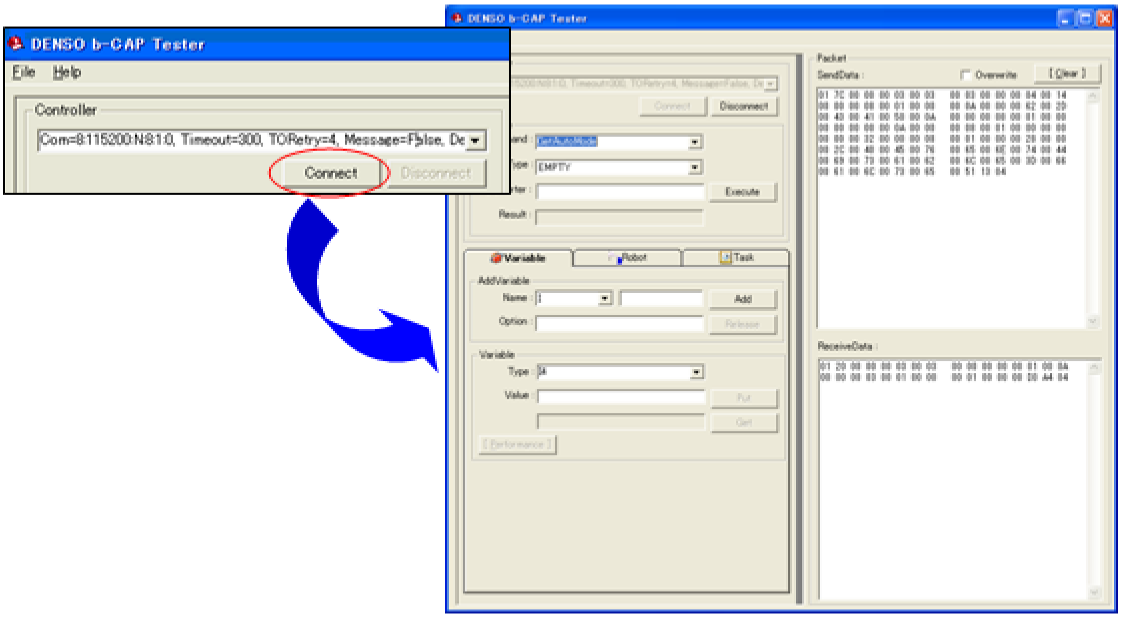
(2) At the Controller connection setting pull-down menu, select the second item from the bottom, and change the COM port to the recognized port number.



```
Server=192.168.0.1, UDP=0, Timeout=300, TORetry=1, Message=Fals
Server=192.168.0.1, UDP=0, Timeout=3000, TORetry=1, Message=Fal
Server=192.168.0.1, UDP=0, Timeout=300, TORetry=1, Message=Fals
Server=192.168.0.1, UDP=0, Timeout=300, TORetry=1, Provider=CaoF
Server=192.168.0.1, UDP=1, Timeout=300, TORetry=8, Message=Fals
Server=192.168.0.1, UDP=1, Timeout=300, TORetry=8, Provider=CaoF
Com=1:115200:N:8:1:0, Timeout=300, TORetry=4, Message=False, De
Com=1:115200:N:8:1:0, Timeout=300, TORetry=4, Provider=CaoProv.D
```

図 6-3 Connection setting

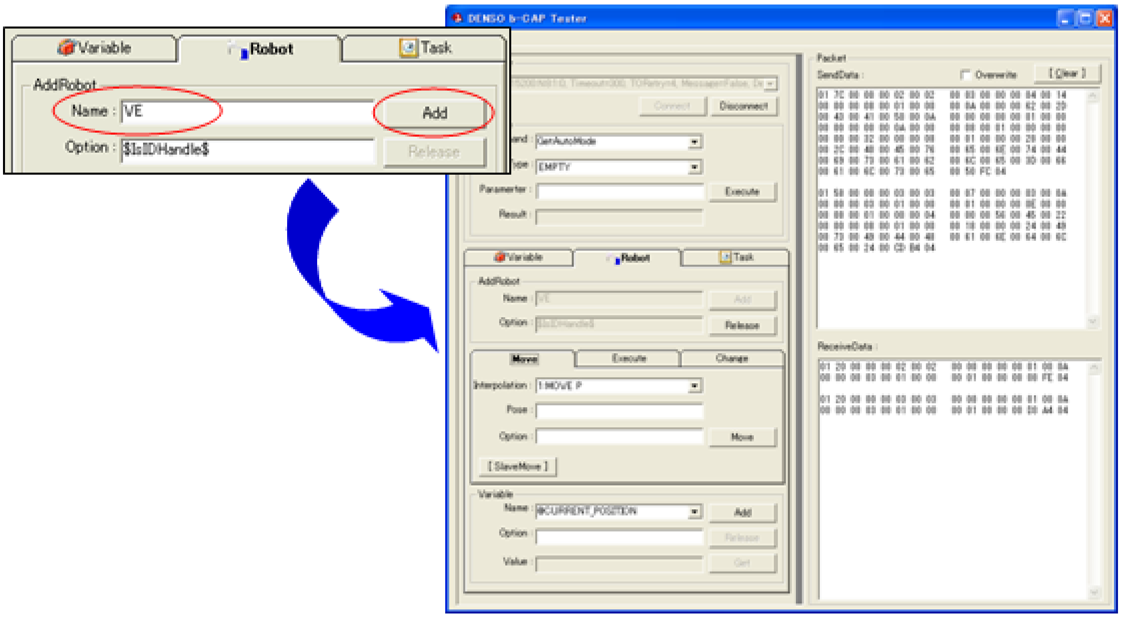
(3) Press "Connect" button to connect to VE026A.



Com=8:115200:N:8:1:0, Timeout=300, TORetry=4, Message=False, De

図 6-4 Starting b-CAP communication and controller class connection

(4) Select "Robot" tag, input a controller name (ex. VE) at Name area of "AddRobot", and press "Add" button.

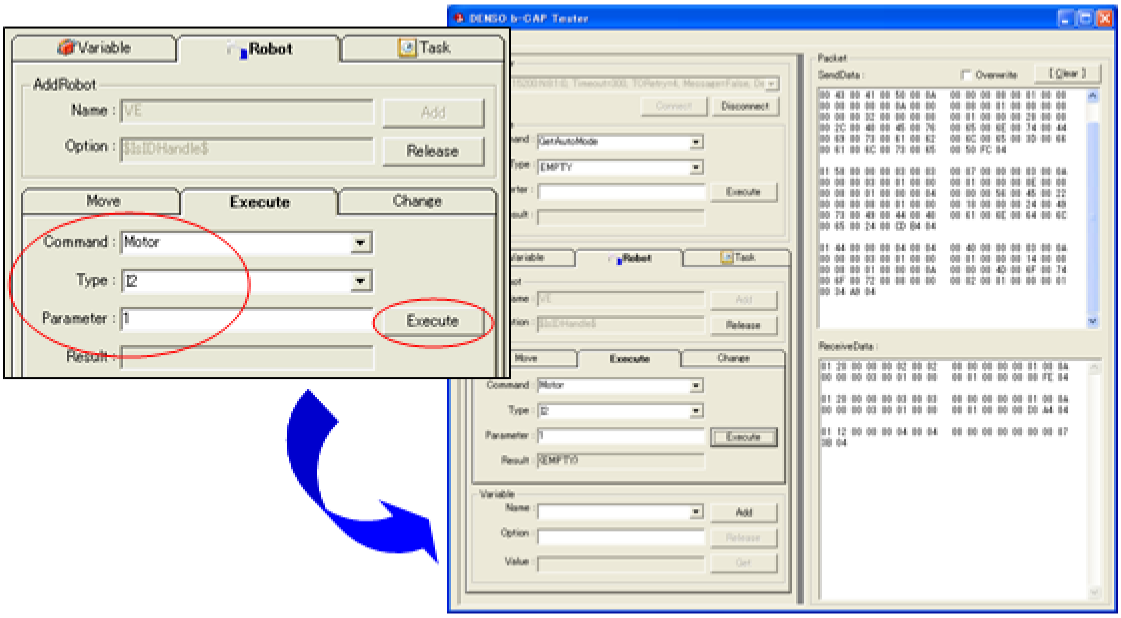


Name : VE

Option : $IsIDHandle$

図 6-5 Robot class connection

(5) In the "Robot" tag, select "Execution" tag. At "Command" pull-down menu, select "Motor", input '1' to "Parameter" area, and then press "Execute" button. This will turn on the motor of the robot.

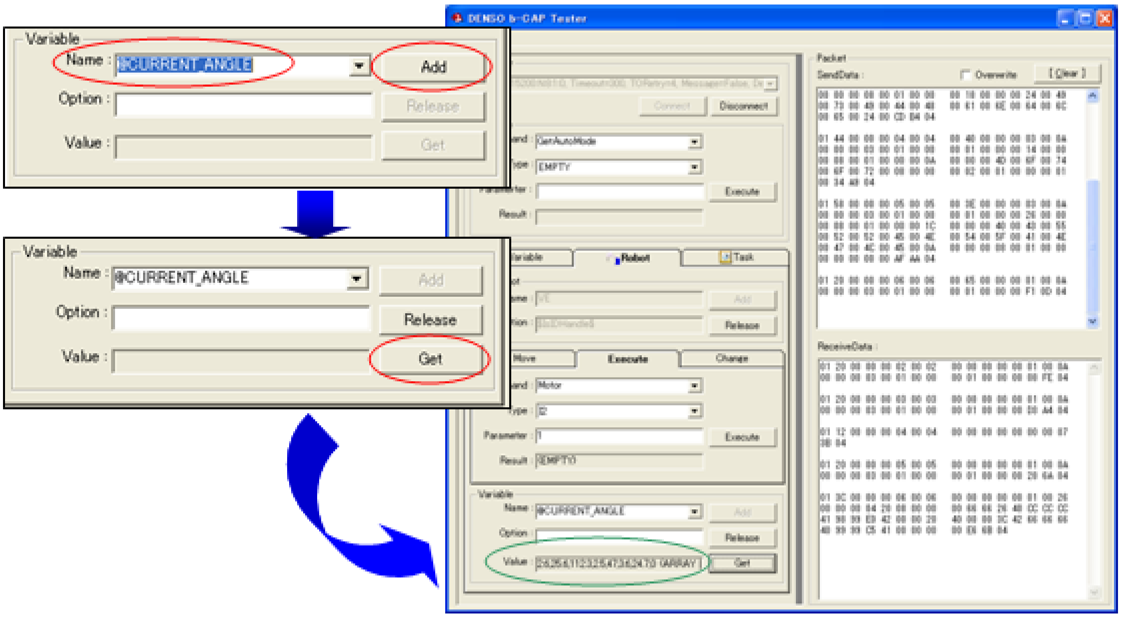


AddRobot
Name : VE
Option : $IsIDHandle$
Command : Motor
Type : I2
Parameter : 1
Result :

図 6-6 Turning-on the robot motor

(6) In the "Robot" tag, select "Variable" tag. At "Name" area, input "@ CURRENT_ANGLE" and press "Add" button. Then press "Get" button.

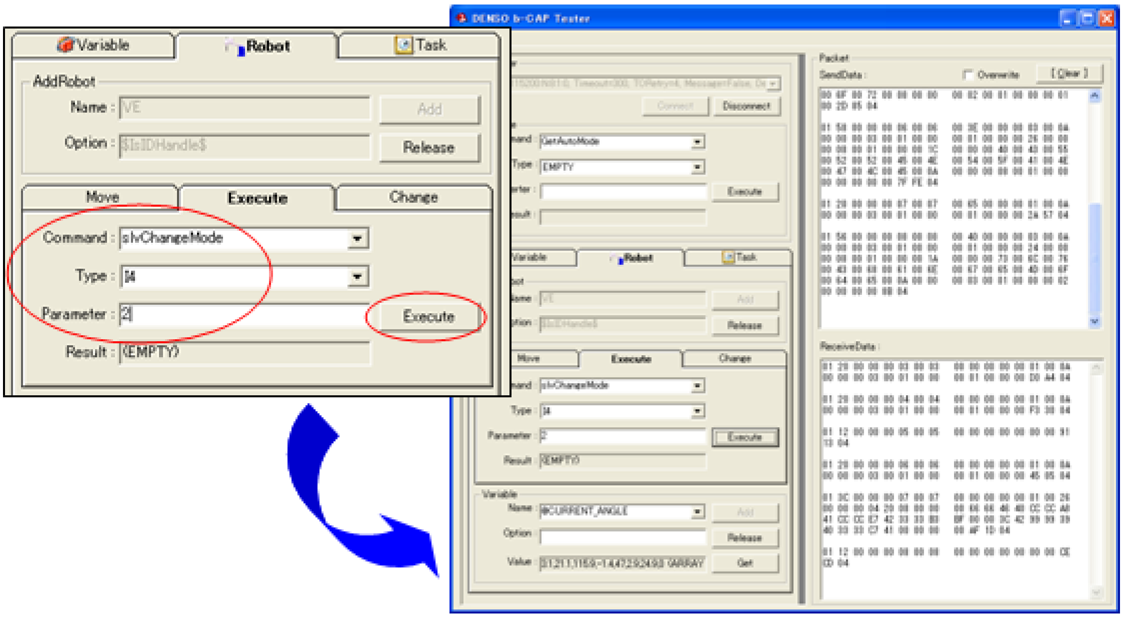
※Copy the value at "Value" area.



Variable
Name : @CURRENT_ANGLE
Option :
Value :

図 6-7 Getting robot current position (Joint angle)

(7) In “Robot” tag, select “Execute” tag. At “Command” pull-down menu, select “slvChangeMode”, input ‘2’ to ”Parameter” area, and then press “Execute” button.



Command : slvChangeMode

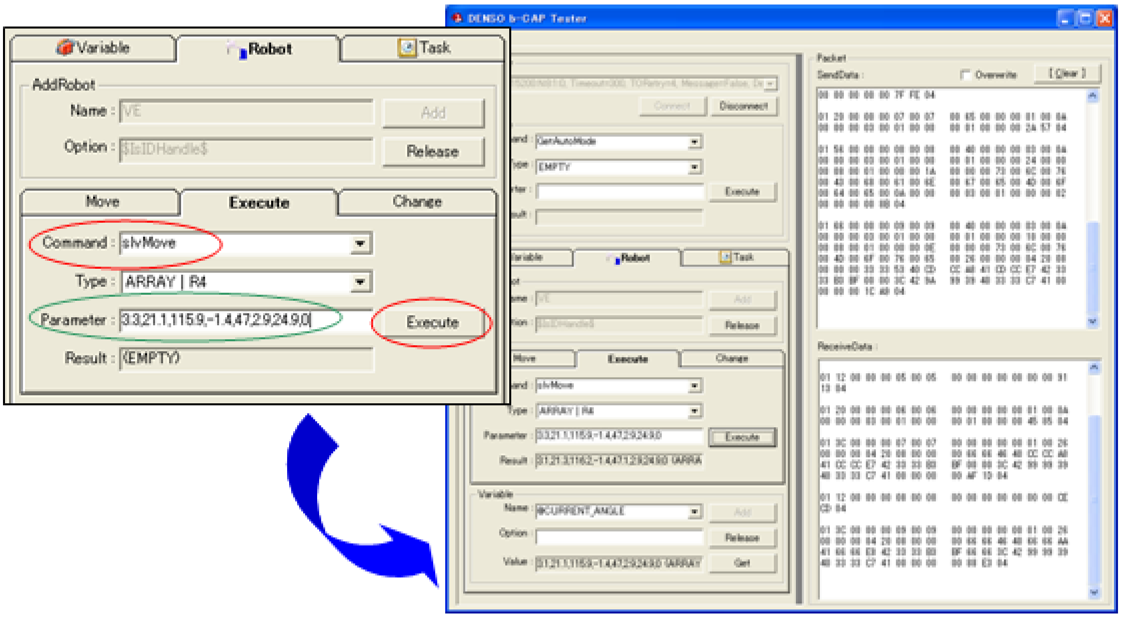
Type : I4

Parameter : 2

Result : (EMPTY)

図 6-8 Starting slave mode

(8) In “Robot” tag, select “Execute” tag. At “Command” pull-down menu, select “slvMove”, then paste the copied value at step (6) to ”Parameter” area, and then press “Execute” button.



Command : slvMove

Type : ARRAY | R4

Parameter : 3.3,21.1,115.9,-1.4,47,2.9,24.9,0

Result : (EMPTY)

図 6-9 Execute slave mode

※After finishing above operations, make sure to change “MOTOR” to ‘OFF’, or main power to ‘OFF’. Keeping the same pose with motor power ON will generate motor temperature error, and the robot cannot move