TCmini providers

Shibaura Machine PLC

Version 1.1.1

User's Guide

August 30, 2021

[Notes]

[Revision History]

Version	Date	Description
1.0.0.0	2006-02-24	First edition
1.0.0.1	2010-02-12	Addition of error code
1.0.0	2012-07-17	Changing Document Version Rules
1.1.0	2020-02-26	Bug fixes for system variables. Modified providers summary. Modified defaults for Conn parameters. Added use case (C#). Communication protocol correspondence table was added.
1.1.1	2021-08-30	Changed the company name of Toshiba Machine in the document to Shibaura Machine.

[Operation check device]

Model	Version	POINTS OF CAUTION
TCmini		

Table of Contents

1. Introduction

This manual is the user's guide of TCmini provider, a CAO provider for the small programmable controller TCmini made by Shibaura.

TCmini provider is a provider that uses CAOs' interfaces to send and receive TCmini commands.

2. Provider Overview

2.1. Introduction

TCmini providers generate and send commands in classes of variables. This allows clients to easily set and retrieve values to TCmini relays.

Fig. 2-1 shows the correspondence between provider classes and TCmini. CaoProvController corresponds to TCmini main unit, and CaoProvVariable corresponds to TCmini relay address and data register address.

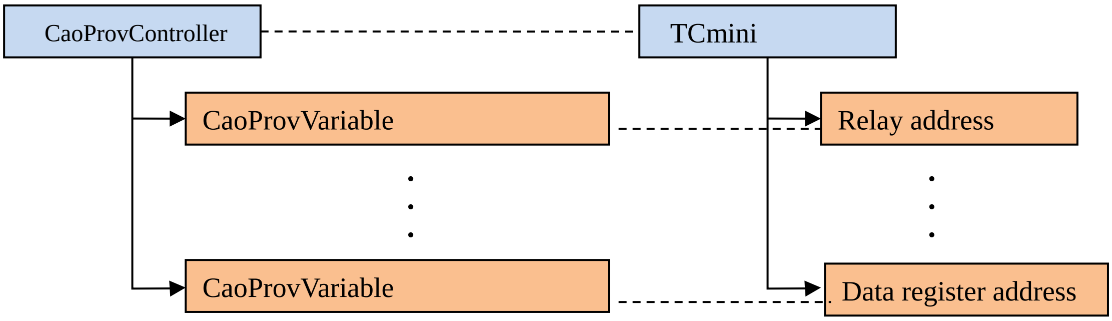

Fig. 2-1 Correspondence Between Providers Classes and TCmini

Fig. 2-2 shows the connections between the providers and TCmini. Connect TCmini to be connected by serial connection (RS232C).

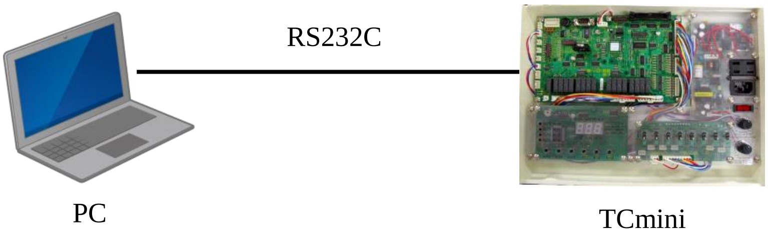

Fig. 2-2 Connections Between Providers and TCmini

TCmini provider have a DLL (Dynamic Link Library) file format, which is detailed in Table 2-1.

Table 2-1 TCmini Provider

File name	CaoProvTCmini.dll
ProgID	CaoProv.TCmini
Registry registration[1]	Regsvr32 CaoProvTCmini.dll
Delet registration	Regsvr32 /u CaoProvTCmini.dll

[1] You do not need to manually register or remove it if you install it using ORiN SDK.

2.2. Method Properties

2.2.1. CaoWorkspace::AddController method

TCmini providers refer to the connection parameters for communication at the time of AddController and make the communication connection. Specify the connection parameters and timeout settings in the option string. Table 2-2 shows the options that are described in option strings.

Format

```
AddController
(
    "<Controller>",          // Controller name
    "CaoProv.TCmini",        // Provider name. Fixed.
    "<machine name>",        // The machine name on which the provider is running.
    "<options>"              // Option string
)
```

Table 2-2 CaoWorkspace::AddController Optional strings

OPTIONS	Meaning
Conn = <Connections Parameters>	Required. Set the communication mode and its connection parameters. Details are shown in Section 2.2.1.1.
Timeout[=<timeout>]	Specify the timeout period for transmission/reception from 0 to 4294967295ms. (Default: 500 ms)

2.2.1.1. Conn parameter

The following are Parameter optional connect parameter strings: Square brackets ("[]") indicate optional characters. The underlined portion of the explanation of each parameter indicates the default value when no option is specified.

"com:<COM Port>[:<BaudRate>[:<Parity>:<DataBits>:<StopBits>[:<Flow>]]]"

<COM Port>	: COM port number. '1'-COM1,'2'-COM2, …
<BaudRate>	: Communication speed. 4800,9600,19200,38400,57600,115200.
<Parity>	: Parity . 'N'-NONE,'E'-EVEN,'O'-ODD
<DataBits>	: Number of data bits. '7'-7bit,'8'-8bit.
<StopBits>	: No. of Stop Bits . '1'-1bit,'2'-2bit.
<Flow>	: Flow control. '0'-no flow-control, '1'-Xon/Xoff, '2'-Hardware control. (OR operation specifiable)

(ex.1) "com:1"	= COM 1 (, 19200 bps, None, 8 bits, 2 bits)
(ex.2) "com:2:9600"	= COM2, 9600 bps (, None, 8 bits, 2 bits)
(ex.3) "com:3:38400:N:8:2"	= COM3, 38400 bps, None, 8 bits, 2 bits

Usage example (C#)

```
using ORiN2.ManagedCAO;
CCaoEngine eng = new CCaoEngine();
CCaoWorkspace ws = eng.AddWorkspace("SampleWorkspace", "");
CCaoController ctrl = ws.AddController("controller1","CaoProv.TCmini", string.Empty,
                                                    "Conn=com:1,Timeout=1000");
```

2.2.2. CaoController::get_VariableNames Property

Retrieves system variables available to TCmini providers. You can obtain the list in Table 2-3.

Return Type	Description
VT_ARRAY | VT_BSTR	You can obtain a list of variables that can be used for AddVariable variable names in a CaoController.

Usage example (C#)

```
string[] VariableNames = ctrl.GetVariableNames(string.Empty);
Debug.WriteLine(VariableNames[0]);        // @CURRENT_TIME
Debug.WriteLine(VariableNames[1]);        // @ERROR_DESCRIPTION
```

Table 2-3 Controller class system variables

Variable name	Data Type	Description	Attribute	
			Get	Put
@CURRENT_TIME	VT_DATE	Present time of TCmini (Note: Relay address A009=1 must be set.)	○	-
@ERROR_DESCRIPTION	VT_BSTR	Alarm messages of TCmini	○	-
@MAKER_NAME	VT_BSTR	Manufacturer name. Returns "Shibaura Machine".	○	-
@TYPE	VT_BSTR	Model name. Returns "TCmini".	○	-
@VERSION	VT_BSTR	The version of the provider.	○	-

2.2.3. CaoController::AddVariable method

Generate CaoVariable objects from CaoController. You can specify the system and user variables in Table 2-3 as variable names. No option string is required.Table 2-3 Controller class system variables

Format

```
AddVariable
(
    "<variable name>"              // Variable name
    "<options>"                    // Option string (not used)
)
```

You can specify user variables as follows:

Format

Category code (1 character) + address (3 hexadecimal digits)

Item	Relay/register	Possible values
Category code	Relay (1bit)	X,Y,Z,R,E,L,T,C,A
	Register (16 bit)	D,P,V
Address	Relay (1bit)	Shows the address as a three-digit hexadecimal number. By setting the third digit to 'W', 'L', or 'H', it becomes 16bit or 8bit. W... Means 16 bits. L... Means lower 8 bits. H... Means upper 8 bits.
	Register (16 bit)	Shows the address as a three-digit hexadecimal number.

(Ex.1) "X100","Y020", "D150", "T12A"

(Ex.2) "X10W" ・・・ In the range of X100- X 10 F

(Ex.3) "x10L" ・・・ In the range of X100- X107[2]

(Ex.4) "X10H" ・・・ In the range of X108- X 10 F

Optional communication commands can also be specified as variable names. When get_Value is executed, messages of <ESC>+<variable name>+<CR> are sent according to TCmini protocols. put_Value is not supported.

(Ex.1) "2,X100" ・・・ X100 I/O read

[2] User variable names are not case-sensitive. All uppercase letters are treated.

(Ex.2) "9,X100,X200" ・・・ Forced X100,X200 I/O Resets

Table 2-4 lists the data types and get/put attributes of user variables.

Table 2-4 List of Variable Class User Variables

Variable name	Data Type	Attribute	
		Get	Put
Data memory Ex) "X100","D000"	VT_I2 (If you run get with a nonexistent address (such as "X999"), VT_BSTR returns the string "Parameter Error".)	○	○
COMMUNICATION COMMANDS Ex) "2,X100"	VT_BSTR (When an invalid command is executed, the string "Parameter error" is returned.)	○	-

Usage example (C#)

```
CCaoVariable varCurrentTime = ctrl.AddVariable("@CURRENT_TIME", "");
CCaoVariable varX100 = ctrl.AddVariable("X100", "");
CCaoVariable varX10W = ctrl.AddVariable("X10W", "");
CCaoVariable varD150 = ctrl.AddVariable("D150", "");
CCaoVariable varX100X200 = ctrl.AddVariable("2,X100,X200", "");
```

2.2.4. CaoVariable::put_Value Property

Set the variables that you created with AddVariable method.

Usage example (C#)

```
varX100.Value = 1;            // X100 : ON
varX10W.Value = -1;           // X100 ~ X10F : All ON
varD150.Value = 32767;        // D150 : 32767
```

2.2.5. CaoVariable::get_Value Property

Retrieves values from variables created by AddVariable method.

Usage example (C#)

```
Debug.WriteLine(varCurrentTime.Value);      // 2020/01/01 12:00:00
Debug.WriteLine(varX100.Value);             // When X100 is OFF : 0
                                            // When X100 is ON : 1
```

```
            Debug.WriteLine(varX10W.Value);                 // When X100~X10F are all OFF：-1
                                                            // When X100~X10F are all ON：0
            Debug.WriteLine(varD150.Value);                 // -32768~32767
            Debug.WriteLine(varX100X200.Value);             // When X100,X200 are OFF：  0,0
```

2.3. Error Codes

There are no unique error codes for TCmini providers. For information about ORiN2 common error, see the error code chapters in ORiN2 Programming Guide.

3. Sample program (C#)

The following shows an example of getting the present time of TCmini and setting and getting the value of the address "D000".

```
using ORiN2.ManagedCAO;

// Create CaoEngin, CaoWorkspace, CaoController
CCaoEngine eng = new CCaoEngine();
CCaoWorkspace ws = eng.AddWorkspace("sample", "");
CCaoController ctrl = ws.AddController("ctrl1", "CaoProv.TCmini", string.Empty,
                                                        "Conn=com:1,Timeout=1000");


// Create system variable(@CURRENT_TIME)
string[] VariableNames = ctrl.GetVariableNames(string.Empty);
CCaoVariable varCurrentTime = ctrl.AddVariable(VariableNames[0].ToString(), "");
Debug.WriteLine(varCurrentTime.Value);                  // 2020/01/01 12:00:00

// Create user variable(D000)
CCaoVariable varD000 = ctrl.AddVariable("D000", "");
varD000.Value = 1;                          // Write 1 to D000
Debug.WriteLine(varD000.Value);             // 1

// Remove object
if(eng != null)
{
    eng.Dispose();
}
eng = null;
ws = null;
ctrl = null;
```

4. Communication Protocol Command Correspondence Table

Table 4-1 shows the correspondence between Variable variables implemented by this provider and the communication commands with TCmini.

Table 4-1 Communication Command Correspondence Table

Variable name	Get/put	COMMUNICATION COMMANDS
@CURRENT_TIME	Get	5,D126,D125,D124,D123,D122,D121,D120
@ERROR_DESCRIPTION	Get	1
○*** (○...X/Y/Z/R/E/L/T/C/A) (***...3 hexadecimal digits)	Get	2, (X/Y/Z/R/E/L/T/C/A) ***
	Put	Change to ON status: 8, ○ *** Change to OFF status: 9, ○ ***
○** △ (○...X/Y/Z/R/E/L/T/C/A) (**...2 hexadecimal digits) (△...W/L/H)	Get	5,○**△
	Put	10,○**△
○*** (○...D/P/V) (***...3 hexadecimal digits)	Get	5,○***
	Put	10,○***