APP-NOTE 500048
Beckhoff Application Note
Date: 2/14/2002
Document Status: First Draft

BECKHOFF AUTOMATION LLC
12204 Nicollet Avenue S.
Minneapolis, MN 55337
Phone 952 890 0000
Fax 952 890 2888

Determining the Terminal Composition Using the PLC Interface and Modbus TCP

OVERVIEW

This document describes how to use the PLC interface registers at Modbus TCP 0x100A and 0x110A. This PLC interface can be used to read/write any registers in the bus coupler or in the attached terminals. The PLC interface is referred to on p. 49 of the BC9000 manual or p. 44 of the BK9000 manual.

SAMPLE FILES USED WITH THIS DOCUMENT

Sample Files	How to Use the Sample Files
FrmTerminals.frm	A visual basic form used to read these values is included for reference. The form will need to be opened with Visual Basic, then the sequence of code can be analyzed and altered to your custom purposes. For a more detailed example, please contact technical support.

KEY CONCEPTS AND PROCEDURES

1. KS2000 configuration software is essential to developing an application that uses the PLC interface for testing and debugging. For example, you can use KS2000 to view the table 9 (terminal configuration) values and verify results.

2. If available, use an additional task set up in System Manager to first manually manipulate and view the PLC interface. (As of this writing, however, the BC/BK9000's PLC interface is not available in System Manager. An example with a lightbus coupler is included).

3. Ideally, the status byte of the PLC interface should be read back and confirmed that the toggle bit changed after each of the steps outlined below. In the least, a short pause must be taken after each write to make sure the BK9000 has time to map the data over to the Modbus image.

PROCEDURE

1. PLC interface

The PLC interface is used to access register data on coupler as well as on the terminals. This interfaces needs 2 byte in both the process data input and output image. On a BC/BK9000 the two input and output bytes are mapped to 0x100A and 0x110A respectively.

The FIFO functionality is not implemented yet.

Byte	Bit Addresses			Values
Output Byte 0	0.0	Toggelbit		0/1
	0.1	--		
	0.2	Toggelbit Reset Request		0/1
	0.3 - 0.7	Function Code		
Output Byte 1	Databyte if Write Request			0 - 255
Input Byte 0	0.0	Toggelbit		0/1
	0.1	--		
	0.2	Reset Acknowledge Bit		0/1
	0.3	Errorbit (Routing / Write protect)		0/1
	0.4	Status FIFO		0/1
		0	no data in FIFO	0
		1	Valid data in FIFO	1
Input Byte 1	Databyte if Read Request			0 - 255

Function Codes	0.7	0.6	0.5	0.4	0.3
Send Terminal number	1	0	0	0	0
Send table number	1	0	0	0	1
Send Register number	1	0	0	1	0
Write Low Byte	1	0	0	1	1
Write High Byte	1	0	1	0	0
Write High Byte and increment register address	1	0	1	0	1
Datenbyte in FIFO eintragen	1	1	0	X	X
FIFO Eintrag mit String Startkennung	1	1	0	X	1
FIFO Eintrag mit String Endekennung	1	1	0	1	X
Read Low Byte	0	0	0	1	1
Read High Byte	0	0	1	0	0
Read High Byte and increment register address	0	0	1	0	1
Datenbyte aus FIFO auslsen	0	1	0	0	0

2. Sequence Overview for Reading the Terminal Composition

```
1. Use the SendTerminalNumber function to tell the PLC interface to use terminal 0,
   which is the BC/BK9000 itself.

2. Use the SendTableNumber function to tell the PLC interface to use Table 9. Table
   9 can be viewed with KS2000 so results in the next steps can be verified.

3. Use a loop and the SendRegisterNumber function to step through the registers
   reading the low and high byte for each as laid out in the next steps, until a
   value of 0x000 is seen, which means there are no more terminals.

4. Use the ReadLowByte function to get the lower value of the register.

5. Use the ReadHighByte function to get the high byte of the register.

6. Combine the low and high byte into one number, convert to decimal, and the
   number should match a terminal type, for example KL2502.

7. Repeat steps 3-6 until the combined value is 0x0000, which means the terminals
   have all bee determined.
```

3. Details of Each Step and Example

The following example assumes we have a BK9000 with two terminals, a KL2502 and a KL3312 attached. Using KS2000, it is very easy to view this configuration in table 9 of the register structure. Notice that the first terminal, shown in register 0 - 2 of table 9 have the following values.

Table 9 in a BK9000

Register Number	Value	Description
0	0x2328 (9000).	Describes the BK9000 itself
1	0x09C6 (2502)	2 channel PWM output terminal
2	0x0CF0 (3312)	2 channel thermocouple input

a. Write to Register 0x110A (4362) Send Terminal Number 0

Since table 9 lies in the BC/BK9000, the terminal number must be set to zero by writing the following value to the PLC interface at Modbus offset 0x110A.

Note: Be sure to toggle bit zero of the low byte to initiate the action within the coupler. If the bit was last at a 1, then it must be set to a 0. The present state of the bit can be viewed by reading the input register 0x100A (4106).

Byte 1 ***Data Byte***	Byte 0 ***Control Byte***
Send value of 0, which is the terminal number of the BC/BK9000.	Write using the SendTerminalNumber function (10000)
	1000 0001 or 1000 0000
0x00	0x81 (129) or 0x80 (128)

b. Write to Register 0x110A (4362) Send Table Number 9

Table 9 is set by writing a value of 0x0988 (or **0x0989** depending on the last state of the toggle bit). This corresponds to a decimal value of **2440**.

Byte 1 ***Cata Byte***	Byte 0 ***Control Byte***
Send value of 9, which specifies table 9 as can be seen with KS2000	Write using the SendTableNumber function (10001)
	1000 1001 or 1000 1000
0x09	0x89 (137) or 0x88 (136)

c. Write to Register 0x110A (4362) Send Register Number 1, 2, 3, …

At this point, each register in table nine needs to be read until a value of 0x00 is seen. For example, setting it at register 1 would be a value of **0x0190**, which corresponds to **400** decimal.

Byte 1 ***Cata Byte***	Byte 0 ***Control Byte***
Send value of 1, which corresponds to register 1 or the KL2502.	Write using the SendRegisterNumber function (10010)
	1001 0001 or 1001 0000
0x01	0x91 (144) or 0x90 (145)

Note: An equation for setting the control byte value and stepping through the registers would be

144 + 256 x TermNum

d. Write to Register 0x110A (4362) Read Low Byte

Now reading the low and high byte should work properly since everything is set in the PLC interface above.

0x110A Byte 1 ***Cata Byte***	0x110A Byte 0 ***Control Byte***
Not used	Write using the ReadByteLow function (00011)
	0001 1001 or 0001 1000
0x00	0x19 (25) or 0x18 (24)

After the read is completed by the BK9000, it places the result in the high byte of Modbus register 0x100A (4106). The result seen should match that shown below.

0x100A Byte 1 ***Cata Byte***	0x100A Byte 0 ***Status Byte***
High byte shows the value of the low byte of 2502 in hex. (0x09C6)	Bit 0 will match the toggle bit from the control byte, or return an error if there were problems.
0xC6	0x01 or 0x00

e. Write to Register 0x110A (4362) Read High Byte

Now reading the high byte should work the same as the last read, only a different function code is used.

0x110A Byte 1 ***Cata Byte***	0x110A Byte 0 ***Control Byte***
Not used	Write using the ReadByteHigh function (00011)
	0001 0001 or 0001 0000
0x00	0x21 (33) or 0x20 (32)

After the read is completed by the BK9000, it places the result in the high byte of Modbus register 0x100A (4106). The result seen should match that shown below for a KL2502 in position one on the terminal row.

0x100A Byte 1 ***Cata Byte***	0x100A Byte 0 ***Status Byte***
High byte shows the value of the high byte of 2502 in hex. (0x09C6)	Bit 0 will match the toggle bit from the control byte, or return an error if there were problems.
0x09	0x01 or 0x00

f. Combine the High Byte (0x09) 9 and Low Byte (0xC6) 198

9 X 256 + 198 = 2502 as in KL2502

g. Repeat Steps c - f until all Terminals have been determined

These values can then be used in combination with registers 0x1010 – 0x1013 to automatically map the data in the Modbus TCP client. For example, since a KL2502 ends with a 2, we know it is a two-channel terminal. We know there are 4 bytes per channel on a KL2502 and can automatically determine it’s share of the process image.

SUPPORTING DOCUMENTS AND REFERENCES

1. Supporting documentation can be found at www.beckhoff.com > downloads > bus terminals:
Beckhoff BC9000 Documentation
Beckhoff BK9000 Documentation

2. Refer to other Beckhoff application notes for Modbus TCP using the BC/BK9000

3. Supporting software can be obtained from Beckhoff USA technical support, supportusa@beckhoff.com:
Beckhoff KS2000 Software - shows register structure and values
Beckhoff Modbus Monitor Software - used to troubleshoot Modbus TCP
Beckhoff BK Manager Software - used to view process image layout of coupler