

Autonics 485 Communication Series Application

Manual for TZ(N)

Autonics Coporation
2004/7

Autonics
Sensors & Controllers

Bldg. 402 3rd Fl., Bucheon Techno Park. 193, Yakdae-dong, Wonmi-gu, Bucheon-si, Gyeonggi-do,
420-734, Korea

TEL:032-329-0722, FAX:032-329-0728

<http://www.autonics.com>

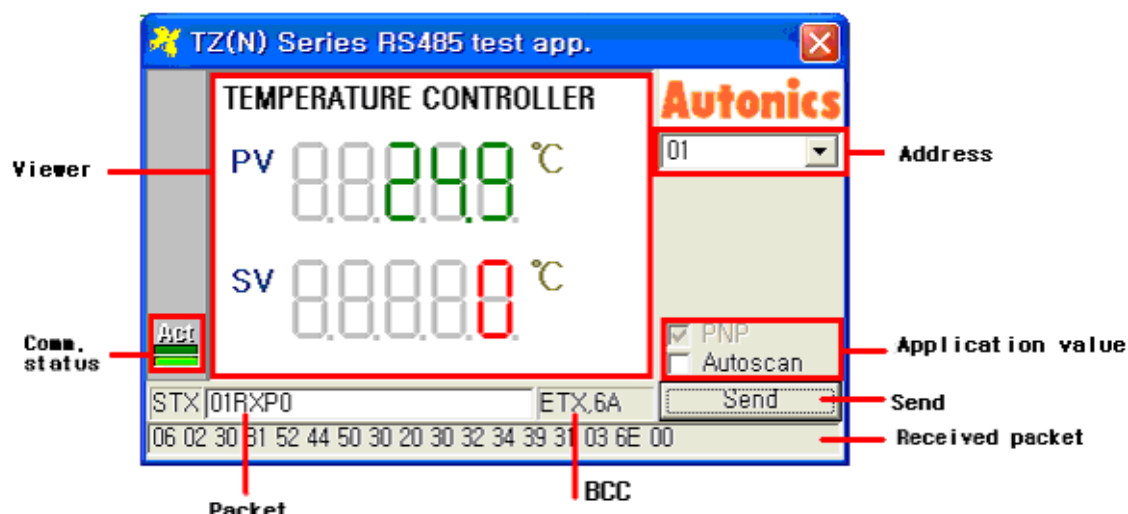


Fig.1 Main screen

Comm. status: TX (Higher LED)/RX (Lower LED)

Green: Succeed Yellow: Fail

Viewer : Display selected address and the P0 value by 7-segment type

Clicking a viewer, Fig. 2 is appeared.

Application value : Bank1 – Select Bank 1

PNP – Increase address automatically

Autoscan – Polling every constant time

Address : Display the searched address, add or delete

Send : Send a packet data to comm. line. Display the processing status.

When Autoscan setting is unlocked, it enables to input.

CRC : Show the calculated value of sending packet plus ETX CRC

Received packet: Display the data, received from comm. line, by HEX value.

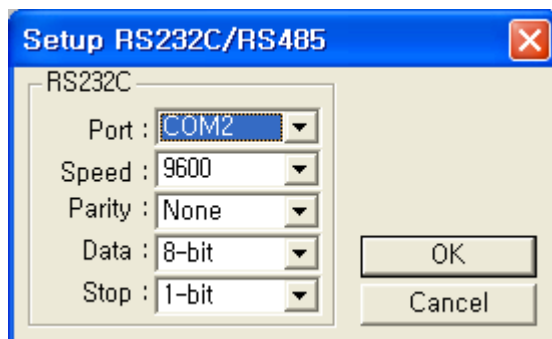


Fig.2 comm. conditions

Setting a communication condition is available.

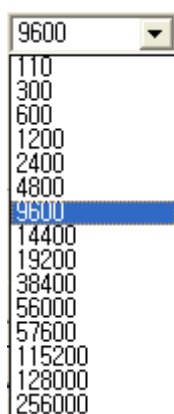
* Port



A dropdown menu for selecting a COM port. The current selection is 'COM2'. The list of options includes COM1, COM2, COM3, COM4, COM5, COM6, COM7, and COM8.

COM2
COM1
COM2
COM3
COM4
COM5
COM6
COM7
COM8

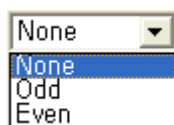
* Speed



A dropdown menu for selecting a baud rate. The current selection is '9600'. The list of options includes 110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000, 57600, 115200, 128000, and 256000.

9600
110
300
600
1200
2400
4800
9600
14400
19200
38400
56000
57600
115200
128000
256000

* Parity



A dropdown menu for selecting a parity setting. The current selection is 'None'. The list of options includes None, Odd, and Even.

None
None
Odd
Even

* Data



A dropdown menu for selecting the number of data bits. The current selection is '8-bit'. The list of options includes 8-bit, 7-bit, and 6-bit.

8-bit
7-bit
6-bit

* Stop



A dropdown menu for selecting the number of stop bits. The current selection is '1-bit'. The list of options includes 1-bit and 2-bit.

1-bit
1-bit
2-bit

OK : When changing comm. conditions, close the comm. port and then open it again.

Cancel: Cancel the set value

CRC (Cyclic Redundancy Check)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	00	5E	BC	E2	61	3F	DD	83	C2	9C	7E	20	A3	FD	1F	41
10	9D	C3	21	7F	FC	A2	40	1E	5F	01	E3	BD	3E	60	82	DC
20	23	7D	9F	C1	42	1C	FE	A0	E1	BF	5D	03	80	DE	3C	62
30	BE	E0	02	5C	DF	81	63	3D	7C	22	C0	9E	1D	43	A1	FF
40	46	18	FA	A4	27	79	9B	C5	84	DA	38	66	E5	BB	59	07
50	DB	85	67	39	BA	E4	06	58	19	47	A5	FB	78	26	C4	9A
60	65	3B	D9	87	04	5A	B8	E6	A7	F9	1B	45	C6	98	7A	24
70	F8	A6	44	1A	99	C7	25	7B	3A	64	86	D8	5B	05	E7	B9
80	8C	D2	30	6E	ED	B3	51	0F	4E	10	F2	AC	2F	71	93	CD
90	11	4F	AD	F3	70	2E	CC	92	D3	8D	6F	31	B2	EC	0E	50
A0	AF	F1	13	4D	CE	90	72	2C	6D	33	D1	8F	0C	52	B0	EE
B0	32	6C	8E	D0	53	0D	EF	B1	F0	AE	4C	12	91	CF	2D	73
C0	CA	94	76	28	AB	F5	17	49	08	56	B4	EA	69	37	D5	8B
D0	57	09	EB	B5	36	68	8A	D4	95	CB	29	77	F4	AA	48	16
E0	E9	B7	55	0B	88	D6	34	6A	2B	75	97	C9	4A	14	F6	A8
F0	74	2A	C8	96	15	4B	A9	F7	B6	E8	0A	54	D7	89	6B	35

Chart1. CRC8 Table

Ex) Calculate below MT4 series packet by Cyclic Redundancy Check

	STX	Addr.		Command		Code		ETX	CRC
ASCII	STX	0	6	R	X	P	0	ETX	CRC
Hexadecimal	02h	30h	36h	52h	58h	50h	30h	03h	2Dh
Binary	00000010	00110000	00110110	01010010	01011000	01010000	00110000	00000011	00101101

1. CRC checking from the next of STX to the before of CRC char by using the table.
 - a. XOR with a calculated value. (In case of the head char., XOR with 00)
 - b. Use the calculated value as an index.
 - c. Get a matched value with index. Repeat it from 'a' until it's finished.

Table[00 ⊕ 30] = BE
 Table[BE ⊕ 36] = 4E
 Table[4E ⊕ 52] = 3E
 Table[3E ⊕ 58] = B8
 Table[B8 ⊕ 50] = 2B
 Table[2B ⊕ 30] = BD
 Table[BD ⊕ 03] = 2D

2. 2D, the last value of #1, is used as CRC value.

<Implement CRC code by ANSI-C>

```

const unsigned char CRC8_table[256] = {
    0x00, 0x5e, 0xbc, 0xe2, 0x61, 0x3f, 0xdd, 0x83,
    0xc2, 0x9c, 0x7e, 0x20, 0xa3, 0xfd, 0x1f, 0x41,
    0x9d, 0xc3, 0x21, 0x7f, 0xfc, 0xa2, 0x40, 0x1e,
    0x5f, 0x01, 0xe3, 0xbd, 0x3e, 0x60, 0x82, 0xdc,
    0x23, 0x7d, 0x9f, 0xc1, 0x42, 0x1c, 0xfe, 0xa0,
    0xe1, 0xbf, 0x5d, 0x03, 0x80, 0xde, 0x3c, 0x62,
    0xbe, 0xe0, 0x02, 0x5c, 0xdf, 0x81, 0x63, 0x3d,
    0x7c, 0x22, 0xc0, 0x9e, 0x1d, 0x43, 0xa1, 0xff,
    0x46, 0x18, 0xfa, 0xa4, 0x27, 0x79, 0x9b, 0xc5,
    0x84, 0xda, 0x38, 0x66, 0xe5, 0xbb, 0x59, 0x07,
    0xdb, 0x85, 0x67, 0x39, 0xba, 0xe4, 0x06, 0x58,
    0x19, 0x47, 0xa5, 0xfb, 0x78, 0x26, 0xc4, 0x9a,
    0x65, 0x3b, 0xd9, 0x87, 0x04, 0x5a, 0xb8, 0xe6,
    0xa7, 0xf9, 0x1b, 0x45, 0xc6, 0x98, 0x7a, 0x24,
    0xf8, 0xa6, 0x44, 0x1a, 0x99, 0xc7, 0x25, 0x7b,
    0x3a, 0x64, 0x86, 0xd8, 0x5b, 0x05, 0xe7, 0xb9,
    0x8c, 0xd2, 0x30, 0x6e, 0xed, 0xb3, 0x51, 0x0f,
    0x4e, 0x10, 0xf2, 0xac, 0x2f, 0x71, 0x93, 0xcd,
    0x11, 0x4f, 0xad, 0xf3, 0x70, 0x2e, 0xcc, 0x92,
    0xd3, 0x8d, 0x6f, 0x31, 0xb2, 0xec, 0x0e, 0x50,
    0xaf, 0xf1, 0x13, 0x4d, 0xce, 0x90, 0x72, 0x2c,
    0x6d, 0x33, 0xd1, 0x8f, 0x0c, 0x52, 0xb0, 0xee,
    0x32, 0x6c, 0x8e, 0xd0, 0x53, 0x0d, 0xef, 0xb1,
    0xf0, 0xae, 0x4c, 0x12, 0x91, 0xcf, 0x2d, 0x73,
    0xca, 0x94, 0x76, 0x28, 0xab, 0xf5, 0x17, 0x49,
    0x08, 0x56, 0xb4, 0xea, 0x69, 0x37, 0xd5, 0x8b,
    0x57, 0x09, 0xeb, 0xb5, 0x36, 0x68, 0x8a, 0xd4,
    0x95, 0xcb, 0x29, 0x77, 0xf4, 0xaa, 0x48, 0x16,
    0xe9, 0xb7, 0x55, 0x0b, 0x88, 0xd6, 0x34, 0x6a,
    0x2b, 0x75, 0x97, 0xc9, 0x4a, 0x14, 0xf6, 0xa8,
    0x74, 0x2a, 0xc8, 0x96, 0x15, 0x4b, 0xa9, 0xf7,
    0xb6, 0xe8, 0x0a, 0x54, 0xd7, 0x89, 0x6b, 0x35
};

/* The function for checking one CRC */
unsigned char CRC8_Check(unsigned char CRC, unsigned char data)
{

```

```

        return CRC8_table[CRC ^ data];
    }

    /* The function for checking the packet of CRC */
    unsigned char CRC8_Packet(unsigned char* packet, int size)
    {
        unsigned char CRC = 0x00;
        while(size--) { CRC = CRC8_table[CRC ^ *packet++]; }
        return CRC;
    }

```

BCC (Block Check Character)

Ex) Calculate TZ(N) series packet by Block Check Character

	STX	Addr.		Command		Code		ETX	BCC
ASCII	STX	0	6	R	X	P	0	ETX	BCC
Hexadecimal	02h	30h	36h	52h	58h	50h	30h	03h	6Fh
Binary	00000010	00110000	00110110	01010010	01011000	01010000	00110000	00000011	01101111

Operate XOR from the next letter of STX to the letter before BCC.

$$30 \oplus 36 \oplus 52 \oplus 58 \oplus 50 \oplus 30 \oplus 03 = 6F$$

Implement BCC code by ANSI-C

```

    /* The function for checking the packet of BCC */
    unsigned char BCC_packet(unsigned char *packet, int size)
    {
        unsigned char BCC = 0x00;
        while(size--) { BCC ^= *packet++; }
        return BCC;
    }

```