# RS-485 Port Communication Protocol for Multi-channel Temperature Scanner

1. Modbus Communications
   1) 19200 bits per second
   2) 8 data bits, 1 stop bits, no parity.
   3) Modbus RTU slave, address 1 to 247.

2. Modbus Input Registers
   1) General: The scanner has a number of Modbus input registers. These registers are used to read the temperatures measured by the 1~16 channel inputs.

   2) Registers: The following table lists the Modbus input registers;

| Register | Description | Values |
|---|---|---|
| 30001 | INPUT 1 | Temperature in degrees Celsius |
| 30002 | INPUT 2 | Temperature in degrees Celsius |
| 30003 | INPUT 3 | Temperature in degrees Celsius |
| 30004 | INPUT 4 | Temperature in degrees Celsius |
| 30005 | INPUT 5 | Temperature in degrees Celsius |
| 30006 | INPUT 6 | Temperature in degrees Celsius |
| 30007 | INPUT 7 | Temperature in degrees Celsius |
| 30008 | INPUT 8 | Temperature in degrees Celsius |
| 30009 | INPUT 9 | Temperature in degrees Celsius |
| 30010 | INPUT 10 | Temperature in degrees Celsius |
| 30011 | INPUT 11 | Temperature in degrees Celsius |
| 30012 | INPUT 12 | Temperature in degrees Celsius |
| 30013 | INPUT 13 | Temperature in degrees Celsius |
| 30014 | INPUT 14 | Temperature in degrees Celsius |
| 30015 | INPUT 15 | Temperature in degrees Celsius |
| 30016 | INPUT 16 | Temperature in degrees Celsius |
| 30017 | Shorted Inputs | A bit is set if an input has a shorted lead. Otherwise, it is clear. The set bit is as follows;<br>Bit 0: Input 1 shorted. Bit 8: Input 9 shorted.<br>Bit 1: Input 2 shorted. Bit 9: Input 10 shorted.<br>Bit 2: Input 3 shorted. Bit 10: Input 11 shorted.<br>Bit 3: Input 4 shorted. Bit 11: Input 12 shorted.<br>Bit 4: Input 5 shorted. Bit 12: Input 13 shorted.<br>Bit 5: Input 6 shorted. Bit 13: Input 14 shorted.<br>Bit 6: Input 7 shorted. Bit 14: Input 15 shorted.<br>Bit 7: Input 8 shorted. Bit 15: Input 16 shorted. |
| 30018 | Open Inputs | A bit is set if an input has an open lead. Otherwise, it is clear. The set bit is as follows;<br>Bit 0: Input 1 open. Bit 8: Input 9 open.<br>Bit 1: Input 2 open. Bit 9: Input 10 open.<br>Bit 2: Input 3 open. Bit 10: Input 11 open.<br>Bit 3: Input 4 open. Bit 11: Input 12 open.<br>Bit 4: Input 5 open. Bit 12: Input 13 open.<br>Bit 5: Input 6 open. Bit 13: Input 14 open.<br>Bit 6: Input 7 open. Bit 14: Input 15 open.<br>Bit 7: Input 8 open. Bit 15: Input 16 open. |

3. Modbus Holding Registers

1) General: The scanner has a single Modbus holding register. This register is used to set the operating mode of the scanner. In normal mode, the scanner scans all channels (the channel numbers are preset in factory) at an update rate (1 channel / 2 seconds is preset in factory). In customized mode, it scans the chosen channels (at least 1 channel).

2) Registers: The following table list the scanner holding register values;

| Register | Description | Values | |
|---|---|---|---|
| 40001 | Operating Mode | If the bit is set, the input is monitored. Otherwise, the input is ignored. The relation between bits and inputs are as follows; | |
| | | Bit 0: Input 4. | Bit 8: Input 12. |
| | | Bit 1: Input 3. | Bit 9: Input 11. |
| | | Bit 2: Input 2. | Bit 10: Input 10. |
| | | Bit 3: Input 1. | Bit 11: Input 9. |
| | | Bit 4: Input 8. | Bit 12: Input 16. |
| | | Bit 5: Input 7. | Bit 13: Input 15. |
| | | Bit 6: Input 6. | Bit 14: Input 14. |
| | | Bit 7: Input 5. | Bit 15: Input 13. |

4. Modbus Commands

1) General: The scanner supports the following Modbus commands;

- Read input registers (function code 04h).
- Read holding registers (function code 03h).
- Preset single register (function code 06h).

2) Format: All Modbus messages have the following format;

| Address | Function Code | Data | CRC-16 |
|---|---|---|---|
| 1 byte | 1 byte | n-bytes | 2 bytes |

3) Read Input Registers:
Function code 04h is used to read one or more consecutive input registers. The first register read is 30001 plus the offset. For example, to read the temperature of input 4 (input register 30004), send the command;

| Address | Function Code | Register Offset | Number of Registers | CRC-16 |
|---|---|---|---|---|
| 10h | 04h | 00 03h | 00 01h | C2 8Bh |

If the temperature being read was 123$^{o}$C, the scanner will respond with the following;

| Address | Function Code | Byte Count | Register Data | CRC-16 |
|---|---|---|---|---|
| 10h | 04h | 02h | 00 7Bh | 05 10h |

4) Read Holding Registers:
Function code 03h is used to read the content of the single scanner holding register. Since there is only one register available to read, using the register offset of 00 00h will read the contents of register 40001. For example, to check the mode of the scanner, send the command;

| Address | Function Code | Register Offset | Number of Registers | CRC-16 |
|---|---|---|---|---|
| 10h | 03h | 00 00h | 00 01h | 87 4Bh |

If the scanner monitors input 2 only, it will respond with the following;

| Address | Function Code | Byte Count | Register Data | CRC-16 |
|---|---|---|---|---|
| 10h | 03h | 02h | 00 04h | 45 84h |

5) Preset Single Register:
Function code 06h allows the Modbus master to write to the holding register in the scanner. This command allows the mode of the scanner to be set. For example, to set the scanner to scan input 2 ~ input 4, send the command;

| Address | Function Code | Register Offset | Register Data | CRC-16 |
|---|---|---|---|---|
| 10h | 06h | 00 00h | 00 07h | CB 49h |

The scanner will respond with this echo of the command;

| Address | Function Code | Register Offset | Register Data | CRC-16 |
|---|---|---|---|---|
| 10h | 06h | 00 00h | 00 07h | CB 49h |

5. Modbus Exception Responses

1) General: There are four different communictions errors the scanner will recognize and respond to. They are;
- CRC Error.
- Illegal Function (exception code 01).
- Illegal Data Address (exception code 02).
- Illegal Data Value (exception code 03).

2) CRC Error: If the scanner receives a message that contains a CRC error, the received message is ignored. No response is sent.

3) Illegal Function: If the scanner receives a message that is anything other than read holding registers, read input registers, or preset single register (functions 03,04,06), it will respond with an illegal function exception response. This response will be in the form;
- Byte 1: Scanner Address.
- Byte 2: The requested function code with the MSB set to 1 to indicate an exception.
- Byte 3: Exception code 01h indicating an illegal function code.
- Byte 4: LSB of CRC code.
- Byte 5: MSB of CRC code.

For example, if a read coil status function (function code 01) is sent to the scanner, the following would be the response;

| Address | Function Code | Exception Code | CRC code |
|---|---|---|---|
| 10h | 81h | 01h | D1 95h |

4) Illegal Data Address: If the scanner receives a supported command that refers to unsupported registers, it will respond with an illegal data address exception response. This response will be in the form;
- Byte 1: Scanner Address.
- Byte 2: Function code with the MSB set to 1 to indicate an exception.
- Byte 3: Exception code 02h indicating an illegal data address exception.
- Byte 4: LSB of CRC code.
- Byte 5: MSB of CRC code.

For example, if an attempt is made to read a holding register that does not have a valid address, the scanner will respond with the following;

| Address | Function Code | Exception Code | CRC code |
|---|---|---|---|
| 10h | 83h | 02h | 90 F4h |

5) Illegal Data Value: If the scanner receives a preset single register command that contains an illegal value, it will respond with an illegal data value exception response. This response will be in the form;
- Byte 1: Scanner Address.
- Byte 2: Function code with the MSB set to 1 to indicate an exception.
- Byte 3: Exception code 03h indicating an illegal data value exception.
- Byte 4: LSB of CRC code.
- Byte 5: MSB of CRC code.

For example, if the scanner receives an invalid preset single register command, it will respond with the following;

| **Address** | **Function Code** | **Exception Code** | **CRC code** |
|---|---|---|---|
| 10h | 86h | 03h | 52 64h |

6. Reference program for CRC-16

```
uint crcjy_(uchar *str_,uchar crc_cd)
{
   uchar idata crc_gs;
   uchar idata crc_w;
   uint    idata crc_sj;
   crc_sj=0xffff;
  for(crc_gs=0;crc_gs<crc_cd;crc_gs++)
  {
    crc_sj=((crc_sj^str_[crc_gs])&0x00ff)+(crc_sj&0xff00);
    for(crc_w=0;crc_w<8;crc_w++)
    {
     if((crc_sj&0x0001)==0x0000)
     {
      crc_sj>>=1;
     }
     else
     {
      crc_sj>>=1;
      crc_sj^=0xa001;
     }
    }
  }
   crc_sj=((crc_sj&0x00ff)<<8)+((crc_sj&0xff00)>>8);
   return(crc_sj);
}
```