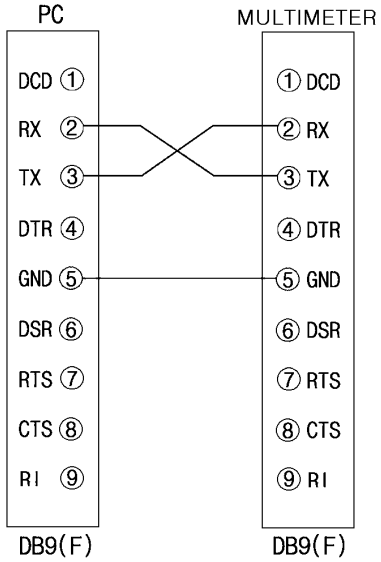
CHAPTER 5. Remote Control

5.1 RS-232 Interface

- **Connect RS-232 interface to 9 pin serial connector of real panel.**
  Communication protocol should be set in the menu of front panel as follows. (refer to menu set function of 4.6, ___ mark is factory set)
  BAUD RATE : 300, 600, 1200, 2400, 4800, 9600 bps
  PARITY : NONE/ 8bits, EVEN/ 7bits, Odd/ 7bits
  START Bit : 1 · STOP Bit : 2
- **Connect with computer**
  The computer is DTE device (Data Terminal Equipment) and the multimeter is DTE device as well. Therefore, use DTE-DTE interface cable (as known as Cross Over cable).



RS-232 CONNECT CABLE

- **Control Command**
  The multimeter complies with the rules and regulations of the present version of SCPI (Standard Command for Programmable Instruments) For more information, please contact to head office of EZ digital co.,Ltd.

5.3 Command

5.3-1 SOME SCPI SYNTAX CONVENTIONS:

Throughout this manual, the following conventions are used for SCPI command syntax. Square brackets (**[ ]**) indicate optional key words or parameters. Braces (**{ }**) enclose parameters within a command string. Triangle brackets (**< >**) indicate that you must substitute a value for the enclosed parameter.

FUNCtion Means you MUST use either all the upper case letters or the entire word. The lower case letters are optional. For example, FUNC and FUNCTION are both valid. However, FUN is not valid.

(Note FUNCtion is used here as an example, but this convention is true for all SCPI commands.)

5.3-2 SCPI COMMAND TERMINATORS

A command string sent to the counter must terminate with a <new line> character. The IEEE-488 EOI (end-or-identify) message is interpreted as a <new line> character and can be used to terminate a command string in place of a <new line> character. A <carriage return> followed by a <new line> is also accepted. Command string termination will always reset the current SCPI command path to the root level.

5.3-3 SIMPLIFIED OVERVIEW

- Sending the MEASure? command is the same as sending a CONFigure command followed immediately by a READ? command.

- Sending the READ? command is like sending the INITiate command followed immediately by the FETCh? command, except readings are not buffered internally.

- With the MEASure? and CONFigure commands, you can select the measurement function, range, and resolution all in one command. Use the range parameter to specify the expected value of the input signal. The multimeter then selects the correct measurement range. Use the resolution parameter to specify the desired resolution for the measurement. Specify the resolution in the same units as the measurement function, not in number of digits. For example, for dc volts, specify the resolution in volts. You must specify a range to use the resolution parameter.

For the range parameter, MIN selects the lowest range for the selected function; MAX selects the highest range; DEF selects auto ranging. For the resolution parameter, MIN select the smallest value accepted, which gives the best resolution; MAX selects the largest value accepted, which gives the least resolution; DEF selects the DEFault resolution which is 4½ digits.

- Example

MEAS:VOLT:DC? 10, 0.001 : 10V range 4½ digit
bus enter statement

CONFigure:VOLT:DC 10, 0.000 : 10V range 5½ digit
READ?
bus enter statement

CONFigure:VOLT:DC 10, 0.0001 : 10V range 6½ digit
INIT
FETCh?
bus enter statement

5.3-4 COMMAND SUMMARY

MEASure:VOLTage:DC?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make a dc voltage measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:VOLTage:AC?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make an ac voltage measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:Current:DC?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make a dc current measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:Current:AC?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make an ac current measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:RESistance?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make a 2-wire ohms measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:FRESistance?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make a 4-wire ohms measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:FREQuency?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make a frequency measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:PERiod?
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Make a period measurement with the specified range and resolution. The reading is sent to the output buffer.

MEASure:CONTinuity?
Make a continuity measurement. The reading is sent to the output buffer.

MEASure:DIODe?
Make a diode measurement. The reading is sent to the output buffer.

MEASure:Temperature?
Make a Temperature measurement.The reading is sent to the output buffer.

CONFigure:VOLTage:DC
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure the multimeter for dc voltage measurements with the specified range and resolution. This command does not initiate the measurement.

CONFigure:VOLTage:AC
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure the multimeter for ac voltage measurements with the specified range and resolution. This command does not initiate the measurement.

CONFigure:Current:DC
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure the multimeter for dc current measurements with the specified range and resolution. This command does not initiate the measurement.

CONFigure:Current:AC
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure the multimeter for ac current measurements with the specified range and resolution. This command does not initiate the measurement.

CONFigure:RESistance
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure the multimeter for 2-wire ohms measurements with the specified range and resolution. This command does not initiate the measurement.

CONFigure:FRESistance
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure the multimeter for 4-wire ohms measurements with the specified range and resolution. This command does not initiate the measurement.

CONFigure:FREQuency
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure a frequency measurement with the specified range and resolution. This command does not initiate the measurement.

CONFigure:PERiod
{<range> |MIN |MAX |DEF},{<resolution> |MIN |MAX |DEF}
Configure a period measurement with the specified range and resolution. This command does not initiate the measurement.

CONFigure:CONTinuity
Configure the multimeter for continuity measurements. This command does not initiate the measurement.

CONFigure:DIODe
Configure the multimeter for diode measurements. This command does not initiate the measurement.

CONFigure:Temperature
Configure the multimeter for temperature measurements. This command does not initiate the measurement.

CONFigure?
Query the multimeter's present configuration and return a quoted string.

FUNCtion "<function>"
Select a measurement function. The function must be enclosed in quotes in the command string (FUNC "VOLT:DC"). Specify one of the following strings.

VOLTage:DC
VOLTage:AC
CURRent:DC
CURRent:AC
RESistance (2-wire ohms)
FRESistance (4-wire ohms)
FREQuency
PERiod
CONTinuity
DIODe

FUNCtion?
Query the measurement function and return a quoted string.

<function>**:RANGe {**<range>**|MINimum |MAXimum}**
Select the range for the selected function. For frequency and period measurements, ranging applies to the signal's input voltage, not its frequency (use FREQuency:VOLTage or PERiod:VOLTage). MIN selects the lowest range for the selected function. MAX selects the highest range. [Stored in volatile memory]

<function>:RANGe? [MINimum |MAXimum]
Query the range for the selected function.

<function>:RANGe:AUTO {OFF |ON}
Disable or enable auto ranging for the selected function.
For frequency and period, use FREQuency:VOLTage or PERiod:VOLTage.
[Stored in volatile memory]

<function>:RANGe:AUTO?
Query the auto range setting. Returns "0"(OFF) or "1"(ON).

<function>:RESolution {<resolution> |MINimum |MAXimum}
Select the resolution for the specified function (not valid for frequency, period, or ratio). Specify the resolution in the same units as the measurement function, not in number of digits. MIN selects the smallest value accepted, which gives the most resolution. MAX selects the largest value accepted which gives the least resolution.
[Stored in volatile memory]

<function>:RESolution? [MINimum |MAXimum]
Query the resolution for the selected function. For frequency or period measurements, the multimeter returns a resolution setting based upon a 3 Hz input frequency.

CALCulate:FUNCtion {NULL |DB |DBM |AVERage |LIMit}
Select the math function. Only one function can be enabled at a time. The default function is null. [Stored in volatile memory]

CALCulate:FUNCtion?
Query the present math function. Returns NULL, DB, DBM, AVER, or LIM.

CALCulate:STATe {OFF |ON}
Disable or enable the selected math function. [Stored in volatile memory]

CALCulate:STATe?
Query the state of the math function. Returns "0"(OFF) or "1"(ON).

CALCulate:AVERage:MINimum?
Read the minimum value found during a min-max operation.
The multimeter clears the value when min-max is turned on, when power has been off, or after a remote interface reset.
[Stored in volatile memory]

CALCulate:AVERage:MAXimum?
Read the maximum value found during a min-max operation. The multimeter clears the value when min-max is turned on, when power has been off, or after a remote interface reset. [Stored in volatile memory]

CALCulate:AVERage:AVERage?
Read the average of all readings taken since min-max was enabled. The multimeter clears the value when min-max is turned on, when power has been off, or after a remote interface reset. [Stored in volatile memory]

CALCulate:AVERage:COUNt?
Read the number of readings taken since min-max was enabled. The multimeter clears the value when min-max is turned on, when power has been off, or after a remote interface reset. [Stored in volatile memory]

CALCulate:DBM:REFerence {<value>**|MINimum |MAXimum}**
Select the dBm reference value. Choose from: 50, 75, 93, 110, 124, 125, 135, 150, 250, 300, 500, 600, 800, 900, 1000, 1200, or 8000 ohms. [Stored in non-volatile memory]

CALCulate:DBM:REFerence? [MINimum |MAXimum]
Query the dBm reference resistance.

CALCulate:LIMit:LOWer {<value >|MINimum|MAXimum}
Set the lower limit for limit testing.[Stored in volatile memory]

CALCulate:LIMit:LOWer? [MINimum |MAXimum]
Query the lower limit.

CALCulate:LIMit:UPPer {<value >|MINimum|MAXimum}
Set the lower limit for limit testing.[Stored in volatile memory]

CALCulate:LIMit:UPPer? [MINimum |MAXimum]
Query the upper limit.

INITiate

Causes the instrument to initiate measurements.

FETCh?

Queries the result. This Query returns the measurement taken by the INITiate commands.

READ?

Initiates measurement, and queries for the result.

TRIGger:SOURce {BUS |IMMediate |EXTernal}

Select the source from which the multimeter will accept a trigger. The multimeter will accept a software (bus) trigger, an immediate internal trigger (this is the default source), or a hardware trigger from the rear-panel Ext Trig (external trigger) terminal.
[Stored in volatile memory]

TRIGger:SOURce?

Query the present trigger source. Returns "BUS","IMM",or "EXT".

SYSTem:BEEPer

Issue a single beep immediately.

SYSTem:BEEPer:STATe {OFF |ON}

Disable or enable the front-panel beeper.
[Stored in non-volatile memory] When you disable the beeper, the multimeter will not emit a tone when:

1) a new minimum or maximum is found in a min-max test.
2) a stable reading is captured in reading hold.
3) a limit is exceeded in a limit test.
4) a forward-biased diode is measured in the diode test function.

SYSTem:BEEPer:STATe?

Query the state of the front-panel beeper. Returns "0"(OFF) or "1"(ON).

SYSTem:ERRor?

Query the multimeter's error queue. Up to 20 errors can be stored in the queue. Errors are retrieved in first-in-first out (FIFO) order. Each error string may contain up to 80 characters.

SYSTem:LOCal

Place the multimeter in the local mode for RS-232 operation. All keys on the front panel are fully functional.

SYSTem:REMote

Place the multimeter in the remote mode for RS-232 operation. All keys on the front panel, except the LOCAL key, are disabled. It is very important that you send the **SYSTem:REMote** command to place the multimeter in the remote mode. Sending or receiving data over the RS-232 interface when not configured for remote operation can cause unpredictable results.

SYSTem:RWLock

Place the multimeter in the remote mode for RS-232 operation. This command is the same as the SYSTem:REMote command except that all keys on the front panel are disabled, including the LOCAL key.

STATus:QUEStionable:ENABle <enable value >

Enable bits in the Questionable Data enable register. The selected bits are then reported to the Status Byte.

STATus:QUEStionable:ENABle?

Query the Questionable Data enable register. The multimeter returns a binary-weighted decimal representing the bits set in the enable register.

STATus:QUEStionable:EVENt?

Query the Questionable Data event register. The multimeter returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

STATus:PRESet

Clear all bits in the Questionable Data enable register.

***CLS**

Clear the Status Byte summary register and all event registers.

***ESE** <enable value>

Enable bits in the Standard Event enable register. The selected bits are then reported to the Status Byte.

***ESE?**

Query the Standard Event enable register. The counter returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

***ESR?**

Query the Standard Event register. The counter returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

***OPC**

Sets the "operation complete" bits(bit 0) in the Standard Event after the command is executed.

***OPC?**

Returns "1" to the output buffer after the command is executed.

***PSC {0 | 1}**

Power-on status clear. Clear the Status Byte and Standard Event register enable masks when power is turned on (*PSC 1). When *PSC 0 is in effect, the Status Byte and Standard Event register enable masks are not cleared when

***PSC?**

Query the power-on status clear setting. Returns "0"(*PSC 0) or "1"(*PSC 1).

***RST**

Reset the counter to its power-on configuration.

***SRE** <enable value>

Enable bits in the Status Byte enable register.

***SRE?**

Query the Status Byte enable register. The counter returns a decimal value which corresponds to the binary-weighted sum of all bits set in the register.

***STB?**

Query the Status Byte summary register. The *STB? command is similar to a serial poll but it is processed like any other instrument command. The *STB? command returns the same return as a serial poll but the "request service" bits(bit 6) is not cleared if a serial poll has occurred.

5.3-5 Command Example

syst:rem -> system to remote

conf:volt:dc <range> -> Function to DCV
<range>
0.1: 100mV range
1: 1V range
10: 10V range
100: 100V range
1000: 1000V range

read? -> Read the results when it is triggered.

conf:res <range> -> Function to 2W ohm
read? -> Read the results when it is triggered.
<range>
100: 100 ohm range
1000: 1k ohm range
10000: 10k ohm range
100000: 100k ohm range
1000000: 1M ohm range
10000000: 10M ohm range
100000000: 100M ohm range

conf:volt:ac <range> -> Function to ACV.
read? -> Read the results when it is triggered.

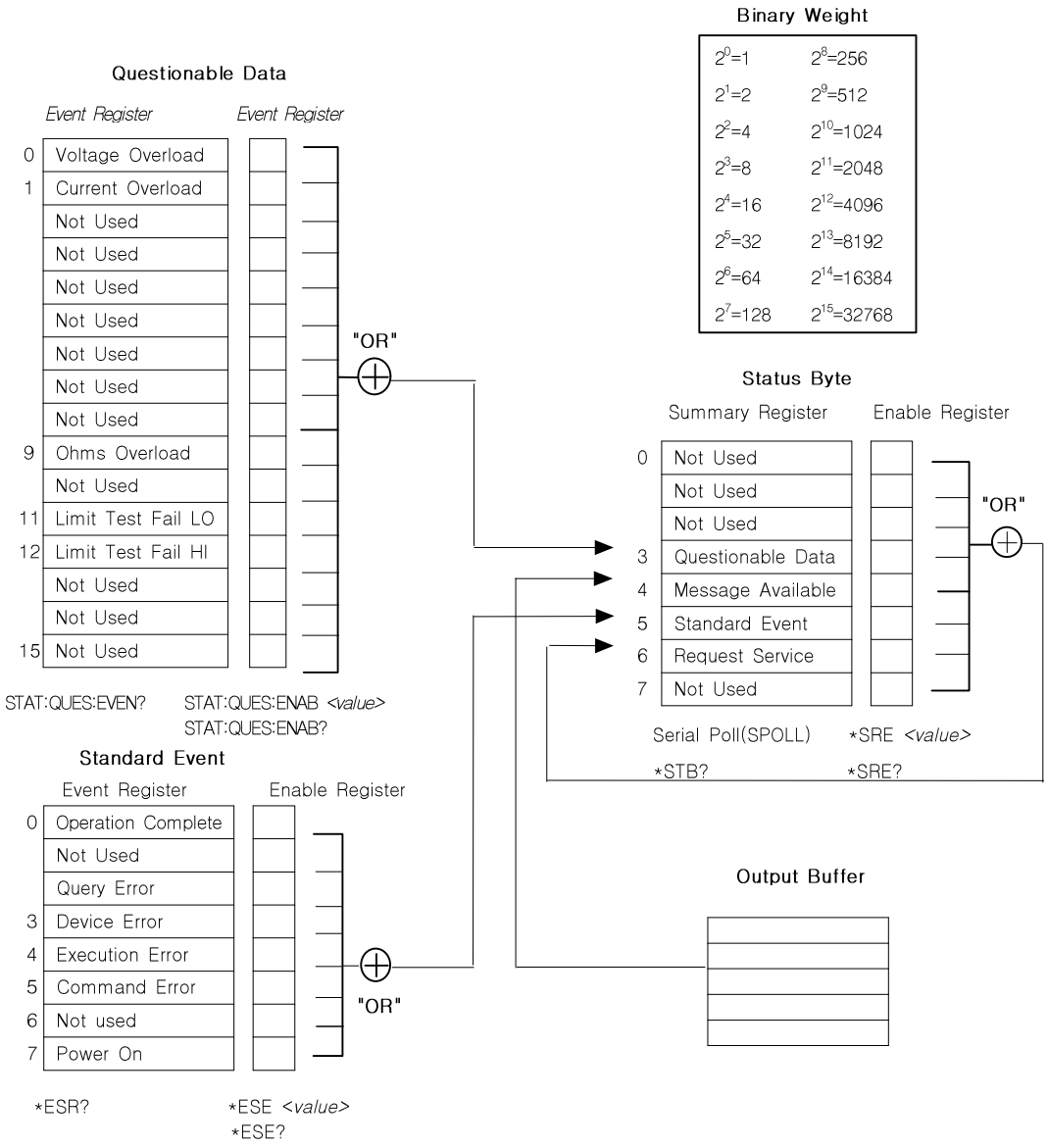
conf:curr:dc <range> -> Function to DCI.
read? -> Read the results when it is triggered.

conf:curr:ac <range> -> Function to ACI.
read? -> Read the results when it is triggered.

conf:freq <range> -> Function to FREQ.
read? -> Read the results when it is triggered.

* terminator code is 0x0a.

5.3-6 SCPI STATUS SYSTEM



5.4 THE SCPI STATUS MODEL

5.4-1 THE STATUS BYTE

The status byte summary register reports conditions from other status registers. Query data that is waiting in the multi meter's output buffer is immediately reported through the "message available" bit (bit 4). Bits in the summary registers are not latched. Clearing an event register will clear the corresponding bits in the status byte summary register. Reading all messages in the output buffer, including any pending queries, will clear the message available bit.

Bit Definitions - Status Byte Register

Bit	Decimal Value	Definition
0 Not Used	1	Always set to 0.
1 Not Used	2	Always set to 0.
2 Not Used	4	Always set to 0.
3 Questionable Data	8	One or more bits are set in the Questionable Data register (bits must be "enabled" in enable register).
4 Message Available	16	Data is available in the multimeter's output buffer.
5 Standard Event	32	One or more bits are set in the Standard Event register (bits must be "enabled" in enable register).
6 Request Service	64	The counter is requesting service (serial poll).
7 Not Used	128	Always set to 0.

The status byte summary register is cleared when:

- You execute a *CLS (clear status) command.
- Querying the standard event and questionable data registers will clear only the respective bits in the summary register.

The status byte enable register (request service) is cleared when:

- You turn on the power and you have previously configured the multimter using the *PSC 1 command.
- You execute a *SRE 0 command.

The status byte enable register will not be cleared at power-on if you have previously configured the multimeter using *PSC 0.

5.4-2 THE STANDARD EVENT REGISTER

The standard event register reports the following types of instrument events: power-on detected, command syntax errors, command execution errors, self-test or calibration errors, query errors, or when an *OPC command is executed. Any or all of these conditions can be reported in the standard event summary bit through the enable register. You must write a decimal value using the *ESE (event status enable) command to set the enable register mask.

Bit Definitions - Standard Event Register

Bit	Decimal Value	Definition
0 Operation Complete	1	All commands prior to and including an *OPC command have been executed.
1 Not Used	2	Always set to 0.
2 Query Error	4	The counter tried to read the output buffer but it was empty. Or, a new command line was received before a previous query has been read.
3 Device Error	8	An Event bit which indicates an operation did not properly complete due to some condition of Counter.
4 Execution Error	16	An execution error occurred (see error numbers -211 through -230).
5 Command Error	32	A command syntax error occurred (see error numbers -101 through -158).
6 Not Used	64	Always set to 0.
7 Power On	128	Power has been turned off and on since the last time the event register was read or cleared.

The standard event register is cleared when:

- You send a *CLS (clear status) command.

- You query the event register using the *ESR? (event status register) command.

The standard event enable register is cleared when:

- You turn on the power and you have previously configured the multimeter using the *PSC 1 command.
- You execute a *ESE 0 command.

The standard event enable register will not be cleared at power-on if you have previously configured the multimeter using *PSC 0.

5.4-3 THE QUESTIONABLE DATA REGISTER

The questionable data register provides information about the quality of the multimeter's measurement results. High/Low limit test result is reported. Any or all of these conditions can be reported in the questionable data summary bit through the enable register.

You must write a decimal value using the STATus:QUEStionable:ENABle command to set the enable register mask.

Bit Definitions - Questionable Data Register

Bit	Decimal Value	Definition
0 Voltage Overload	1	Range overload on dc volts, ac volts, frequency, period, diode function
1 Current Overload	2	Range overload on dc or ac current function
2 Not Used	4	Always set to 0.
3 Not Used	8	Always set to 0.
4 Not Used	16	Always set to 0.
5 Not Used	32	Always set to 0.
6 Not Used	64	Always set to 0.
7 Not Used	128	Always set to 0.
8 Not Used	256	Always set to 0.
9 Ohms Overload	512	Range overload on 2-wire or 4-wire ohms
10 Not Used	1024	Always set to 0.
11 Limit Fail LO	2048	Reading is less than lower limit in limit test.
12 Limit Fail HI	4096	Reading exceeds upper limit in limit test.
13 Not Used	8192	Always set to 0.
14 Not Used	16384	Always set to 0.
15 Not Used	32768	Always set to 0.

The questionable data event register is cleared when:

- You execute a *CLS (clear status) command.
- You query the event register using STATus:QUEStionable:EVENt?.

The questionable data enable register is cleared when:

- You turn on the power (*PSC does not apply).
- You execute the STATus:PRESet command.
- You execute the STATus:QUEStionable:ENABle 0 command.

5.5 HOW TO PROGRAM THE MULTIMETER RESETTING THE MULTIMETER AND CLEARING THE GPIB INTERFACE

Before attempting any programming, it is a good idea to set the Multimeter to a known state. The following command grouping shows how to reset the Multimeter. Before issuing these commands, execute a device clear to reset the interface and Multimeter. Consult your interface card's documentation for how to issue a device clear since the device clear command will be specific to the interface you are using.

Perform the following:

1. Issue a Device Clear
   (See your computer or interface card documentation for how to issue this command).
2. Issue the following commands:
   *RST
   *CLS
   *SRE 0
   *ESE0
   :STAT:PRES

Using the Standard Event Status Register to Trap an Incorrect GPIB command

The following command grouping shows how to use the Standard Event Status Register and the Status Byte Register to alert the computer when an incorrect command is sent to the Multimeter. The command *ESE 32 tells the Multimeter to summarize the command error bit (bit 5 of the Event Status Register) in the Status Byte Register. The command error bit is set when an incorrect command is received by the Multimeter. The command *SRE 32 tells the Multimeter to assert the SRQ line when the Event Status Register summary bit is set to 1. If the Multimeter is serial polled after a command error, the serial poll result will be 96.

*ESE 32 Enable for bad command.
*SRE 32 Assert SRQ from Standard Event Status Register summary.

Using the *OPC Command to Assert SRQ

This method is recommended when the Multimeter is on the GPIB with many other instruments, any of which can assert SRQ. The commands *OPC, *ESE 1 and *SRE 32 are used to assert the SRQ line to alert the computer that the Multimeter has completed a measurement. It is up to the computer to use the serial poll command to determine which of the instruments on the bus requested service.

*ESE 1	Summarize OPC bit for Status Byte Register.
*SRE 32	SRQ when event summary bit is 1 .
	Set up program to specify service routine and enable interrupt when SRQ is asserted.
:INIT	Start measurements.
*OPC	Enable OPC bit.

Program could be doing other things while waiting for SRQ. When SRQ occurs and the Multimeter has been identified as the cause of the SRQ, ask for the data:

:FETCH?	Ask for measurement.

5.6 ERROR MESSAGES

-101 Invalid character

An invalid character was found in the command string. You may have inserted a character such as #, $, or % in the command header or within a parameter.

Example: CONF:VOLT#DC

-102 Syntax error

Invalid syntax was found in the command string. You may have inserted a blank space before or after a colon in the command header, or before a comma.

-103 Invalid separator

An invalid separator was found in the command string. You may have used a comma instead of a colon, semicolon, or blank space or you may have used a blank space instead of a comma.

Example: CONF:FREQ 1000 0.1

-108 Parameter not allowed

More parameters were received than expected for the command.

You may have entered an extra parameter, or you added a parameter to a command that does not accept a parameter.

Example: READ? 10

-109 Missing parameter

Fewer parameters were received than expected for the command.

You omitted one or more parameters that are required for this command.

-113 Undefined header

A command was received that is not valid for this multimeter.

You may have misspelled the command or it may not be a valid command. If you are using the short form of the command, remember that it may contain up to four letters.

-121 Invalid character in number

An invalid character was found in the number specified for a parameter value.

Example: STAT:QUES:ENAB #B01010102

-131 Invalid suffix

A suffix was incorrectly specified for a numeric parameter.

You may have misspelled the suffix.

-138 Suffix not allowed

A suffix was received following a numeric parameter which does not accept a suffix.

-148 Character data not allowed

A discrete parameter was received but a character string or a numeric parameter was expected. Check the list of parameters to verify that you have used a valid parameter type.

-151 Invalid string data

An invalid character string was received. Check to see if you have enclosed the character string in single or double quotes and that the string contains valid ASCII characters.

-221 Settings conflict

This error can be generated in one of the following situations:

· You sent a CONFigure or MEASure command with autorange enabled and with a fixed resolution.
Example: CONF:VOLT:DC DEF,0.1

· You turned math on (CALC:STAT ON) and then changed to a math operation that was not valid with the present measurement function. For example, dB measurements are not allowed with 2-wire ohms. The math state is turned off as a result of this condition.

-222 Data out of range

A numeric parameter value is outside the valid range for the command.

-224 Illegal parameter value

A discrete parameter was received which was not a valid choice for the command. You may have used an invalid parameter choice.
Example: CALC:FUNC SCALE (SCALE is not a valid choice).

-230 Data stale

A FETCh? command was received but internal reading memory was empty. The reading retrieved may be invalid.

-410 Query INTERRUPTED

A command was received which sends data to the output buffer, but the output buffer contained data from a previous command (the previous data is not overwritten). The output buffer is cleared when power has been off, or after a *RST (reset) command has been executed.

514 Command allowed only with RS-232

There are three commands which are only allowed with the RS-232 interface: SYSTem:LOCal, SYSTem:REMote, and SYSTem:RWLock.

550 Command not allowed in local

The multimeter received a READ? command while in the local mode. For RS-232 operation, you should always execute the SYSTem:REMote command before sending other commands over the interface.

551 Command not allowed in remote

552 Limit command not allowed

5.7 Program Example

5.7.1 Turbo C Source using RS-232 interface

```
#include <stdio.h>
#include <dos.h>
#include <stdarg.h>
#include <alloc.h>
#include <conio.h>
#include <stdlib.h>
#include <bios.h>
#include "ascnc.h"

#define     F1KEY       315
#define     ESCKEY      27
#define     SUCCESS     1
#define     FAIL        0
#define     DCD_MSK     0x80            /* data carrier detected */
#define     BUFLEN      0x01000         /* length of data buffer */
#define     DATA        0x3f8           /* data buffer reg. of com1
                        0x2f8 : com2
                        0x3e8 : com3
                        0x2e8 : com4    */
#define     IER         0x3f9           /* interrupt enable reg. of com1 */
#define     IIR         0x3fa       /* interrupt identification reg. of com1 */
#define     MCR         0x3fc       /* modem control reg. of com1 */
#define     MSR         0x3fe       /* modem status reg. of com1 */
#define     LSR         0x3fd       /* line status reg. of com1 */
#define     LCR         0x3fb       /* line control reg. of com1 */
#define     ONMASK1     0xef        /* turn IRQ4(serial port1) on */
#define     ONMASK2     0xf7        /* turn IRQ3(serial port2) on */
#define     OFFMASK1    0x10        /* turn IRQ4(serial port1) off */
#define     OFFMASK2    0x08        /* turn IRQ3(serial port2) off */
#define     PICMASK     0x21        /* 8259 programmable interrupt controller
                                    mask reg. */
#define     PICEOI      0x20        /* 8259 PIC EOI instruction */
#define     XON         0x11        /* define cross on character */
#define     XOFF        0x13        /* define cross off character */
#define     IRQVECT1    0x0c        /* interrupt service routine */

#define     IRQVECT2    0x0b
```

```
int base;

char *ibuffer;
int irqvect;
int offmask;
int front, rear;
unsigned char tick;
char IsDataReady;
void interrupt (*oldtick)(void);
void interrupt (*oldvect)();

void interrupt ticksr (void)
{
    if (tick) tick--;
    outportb(PICEOI, 0x20);
}
void comDataOut(char ch)
{
    char stat;
  do {
       /* line status reg. 0x20 = 00100000b
        bit 5 : THR(transport hold reg) empty
        data buffer reg. empty */
        stat = inportb(LSR - base);
         stat &= 0x20;
    } while (!stat);                    /* waiting for THR empty        */
    outportb(DATA - base, ch);     /* DATA: data buffer reg. of */
}                                      /*       0x3f8 : com1        */

int SendFmtToSci (const char *format, ...)  /*      0x2f8 : com2  */
{                                           /*       0x3e8 : com3   */
    va_list argptr;                        /*        0x2e8 : com4 * /
    char buffer[100], *p;
    int cnt;
    va_start(argptr, format);
    cnt = vsprintf(buffer, format, argptr);
   va_end(argptr);
    p = buffer;
    while (*p) comDataOut(*p++);
```

```
    return(cnt);
}
void EndTick (void)
{

    setvect(0x1c, oldtick);
}
void InitTick (void)
{
    oldtick = getvect(0x1c);
    setvect(0x1c, ticksr);
}
void comEndPort(void)
{
    /* interrupt enable reg. :
        no interrupt on data receive */
    outportb(IER - base, 0x00);
    outportb(MCR - base, 0x00);
    outportb(PICMASK, (inportb(PICMASK) | offmask));
    setvect(irqvect, oldvect);
    free(ibuffer);
}

void ExitProg (void)
{
    comEndPort();
    EndTick();
   exit(0);
}
int comIsDataReady(void)
{
    return((front != rear) ? SUCCESS : FAIL);
}
unsigned char comDataIn(void)
{
    front %= BUFLEN;          /* length of data buffer : 0x01000 */
    return(*(ibuffer + front++));

}
```

```
unsigned char getdata(void)
{
     unsigned long int timelimit = 0;
     IsDataReady = 1;
     while (!comIsDataReady()) {
          if (timelimit++ > 500000) {
               IsDataReady = 0;
               return(0);
          }
   }
     return(comDataIn());
}
void Delay2 (unsigned ms)
{
   char c;
   tick = 18.2 * ms / 1000;
     while (tick) {
          if (kbhit()) ExitProg();
          if (comIsDataReady()) {
                    do {
                  c = getdata();
                            cprintf("%c", c);
               } while (comIsDataReady());
            }
   }
}
int getkey(void)
{
    int key, lo, hi;
    key = bioskey(0);
    lo = key & 0x00ff;
    hi = (key & 0xff00) >> 8;
    return((lo == 0) ? hi + 256 : lo);
}
void ReadPort(int argc, char szport[])
{

   int p = atoi(szport);
   if (argc < 2) port.port = 0;
```

```
    if (p >= 0 && p <= 3)
            port.port = p;
             port.baud = 3;                    /*Baud rate to 9600bps*/
    port.parity = 0;
    port.databits = 8;
    port.stopbits = 1;
}
void main(int argc, char *argv[])
{
     int key;
     InitTick();
     ReadPort(argc, argv[1]);
     comInitPort();
     do{
          if(kbhit()){
               key = getkey();
               switch(key){
                                  /* DMM control command */
             case F1KEY:  SendFmtToSci("conf:volt:dc₩n");
 Delay2(1000);
                                  SendFmtToSci("conf:volt:ac₩n");
 Delay2(1000);
                                  SendFmtToSci("conf:res₩n");
 Delay2(1000);
                                  SendFmtToSci("conf:fres₩n");
 Delay2(1000);
                                  SendFmtToSci("conf:curr:dc₩n");
 Delay2(1000);
                                  SendFmtToSci("conf:curr:ac₩n");
 Delay2(1000);
                                  SendFmtToSci("conf:temp₩n");
 Delay2(1000);
                                  break;
                    case ESCKEY: comEndPort();
                                  EndTick();
                                  exit(0);
               }
          }
     }while(1);
}
```

5.7-2 Basic program using GPIB

```
18 '     Read DC Voltage from DM-500 address 2
19 '     GPIB card is ETC-1201A po chang co,. ltd (CHIA JEI TECHNOLOGY CORP)
20       DEF SEG=&HDC00
30       GPINIT=&H0:SC5=0:MAD%=21
40       GPOUTPUT=&H80:GPINPUT=&H1E0
50       CALL GPINIT(SC%,MAD%)
60       R$="2;CONF:VOLT:DC[1]"
70       CALL GPOUTPUT(R$,ER%)
100      FOR I=1 TO 10
105      F$="2;READ?[1]"
106      CALL GPOUTPUT(F$,ER%)
109      EOS%=0: TK%=2:S$=SPACE$(20)
110      CALL GPINPUT(EOS%,TK%,S$,LN%,ER%)
115      PRINT S$
130      NEXT I
170      END
```

NOTE: statement no 60,105 [1]at the end of the string defines the terminator to CR + EOI