USB to GPIB-CONTROLLER KI-GC1201, KI-GC3201, KI-GB1201,KI-GB1201X,KI-GB1201R and RS232 to GPIB-CONTROLLER KI-GX1201,KI-GX3201 Operational Manual

＊ Functions held with IEEE488.1 and IEEE488.2
＊ Use taken of RS232 “COM?:128000,N,8,1”
＊ Application program written easily
＊ Driver installed simply
＊ Merit of swift response as GPIB

KI Work - Office
http://www.phkaku.com
http://en.phkaku.com
e-mail: kitec@phkaku.com

The GPIB-CONTROLLER of KI-GC1201, KI-GC3201, KI-GB1201, KI-GB1201X, KI-GB1201R , KI-GX1201 and KI-GX3201 (Call it as GPIB-CONTROLLER below) introduced as following:

CONTENTS

The design concept for IEEE488 GPIB-CONTROLLER

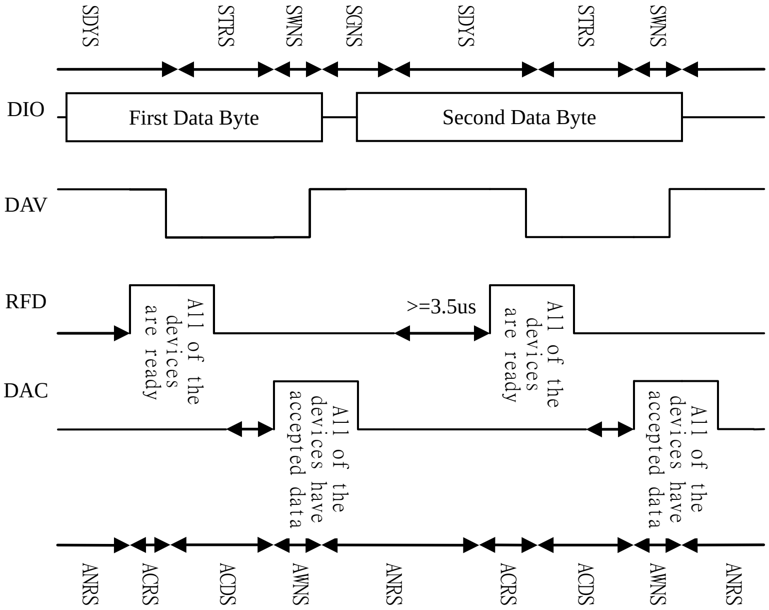
1.The operation of high efficiency,high speed ,high stability, high reliability and high compatibility:

GPIB-CONTROLLER take the Raw-Data format for tranceiving data to or from instrument and can transmite data to multiple instruments at the same time. It waste little delay time for interpreting and waiting data on tranceiving, and the posibility of the tranceiving data error will be low.

GPIB-CONTROLLER utilize the merit of CPLD (or FPGA), strict with the Dynamic-Timing for the communication of GPIB-CONTROLLER, CPU and all instrument and make fine tune on the firmware of CPLD (or FPGA) to get the best communication stability and the best compatibility between the GPIB-CONTROLLER and all the instruments with GPIB-Interface.

The basic rule of GPIB-CONTROLLER

Handshake timing for GPIB-CONTROLLER


SDYS
STRS
SWNS
SGNS
SDYS
STRS
SWNS
DIO
First Data Byte
Second Data Byte
DAV
RFD
All of the devices are ready
>=3.5us
All of the devices are ready
DAC
All of the devices have accepted data
All of the devices have accepted data
ANRS
ACRS
ACDS
AWNS
ANRS
ACRS
ACDS
AWNS
ANRS

1. There are GPIB-CONTROLLER Messages listed as below:
   UNL, LAD, TAD, SPD, SPE, LLO, GET, DCL, SDC, PPC, PPU, ATN, EOI, SRQ, IFC, REN, DAB, STB…
2. The sequence of the Messages transceiving for transferring the Data :
   ATN1, UNL, LAD…, TAD, ATN0, and then DAB… transfered
3. The sequence of the Messages transceiving for receiving the Data :
   ATN1, UNL, LAD, TAD, ATN0, and then DAB… received
4. The sequence of the Messages transceiving for Serial Poll :
   ATN1, UNL, SPE, LAD, TAD, ATN0, and then the STB… for the Serial Poll received, and then SPD

5. The sequence of the Messages transceiving for configuring the Parallel Poll :
   ATN1, UNL, LAD, PPC PPE, and then UNL
6. The sequence of the Messages transceiving for the Response of Parallel Poll :
   ATN1, EOI1 and then the STB for the Parallel Poll received
7. The sequence of the Messages transceiving for configuring the state as Remote :
   ATN1, REN1, LLO, LAD… and then the last message of LAD

2.Instruction is easier to understand and have powerful function once utilized in designing of application program.
The high-level-instruction-interpreter is in SM59264(MPU-8051) instead of being in the GPIB-DRIVER(ATL) of PC. The low level instructions which are output directly from SM59264 through IC CPLD-LC-4128V-75T128C (or FPGA-LCMXO256) quickly will be proply processed by IC and then transmitting signal to GPIB-BUS or receiving signal from GPIB-BUS according to the result obtained from the IC mentioned above.
For working on the Operation-System of PC, only the driver of IC-CP2102 (USB to UART Converter) is sufficient.
The insturctions of High-Level-TEXT-Type is applied.
There are many PROGRAMMING-TOOL for designing the application program of the KI-GCx201 and KI-GB1201series GPIB-CONTROLLER, such as:
VISUAL C++, VISUAL BASIC, VISUAL C#, LabVIEW …

3.Instruments synchronous and sequential measurement can be made easy through GPIB-CONTROLLER.
Synchronous measurement for multiple instruments
For example:
SEND 5 6 12 \'VSET 1,4.5
Instruction of 'VSET 1,4.5' will be transmited to POWER-SUPPLY of address 5, 6 and 12 throuth GPIB-CONTROLLER at a same time.
Program is simple, for example, the program code for the format of VISUAL C++ as below:

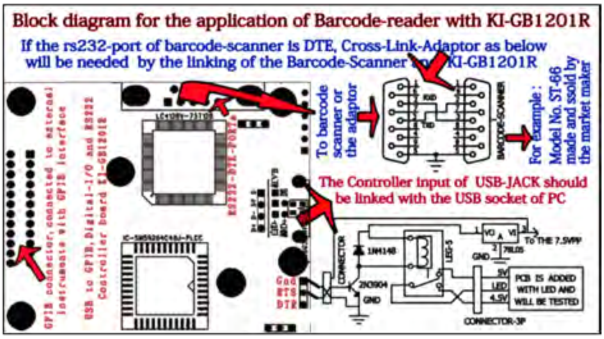
```
WriteFile
(hComm,
 "SEND 5 6 12 \'VSET 1,4.5\'\r\n"
 ,26,&nByteWrite,NULL
);
```

SEND? 7 11 14 ‘MEAS?’
Instruction of 'MEAS? ' will be transmited to Digit-Multimeter of address 7, 11 and 14 throuth GPIB-CONTROLLER at a same time.
Sequencial measurement for multiple instruments,
For example:
SEND 5 'VSET 1,4.5'; SEND 6 'VSET 1,4.5'; SEND 12 'VSET 1,4.5'
Instruction of 'VSET 1,4.5' will be transmited to POWER-SUPPLY of address 5 and then 6 and then 12 throuth GPIB-CONTROLLER.
SEND? 7 'MEAS? '; SEND? 11 'MEAS? '; SEND? 14 'MEAS?'
Instruction of 'MEAS? ' will be transmited to Digit-Multimeter of address 7 and then 11 and then 14 throuth GPIB-CONTROLLER.

4.Intelligent operation for Delay-Time:
The not necessary Delay-Time will be excluded,
for example:
SEND? 7 17 23 24 'MEAS:AC?' 500
Description:
GPIB-CONTROLLER will transmite instruction 'MEAS:AC?' to instrument of Address-7, Address-17, Address-23 and Address-24 at a same time,

and then take a delay of 500ms, and then sequencially read data from the instruments Output-Buffer of Address-7, Address-17, Address-23 and Address-24 and send the data back to PC.
The delay time of 500ms before GPIB-CONTROLLER reading data from the Output-Buffer of Address-7 is necessary, but the delay time of 500ms before reading that of Address-17, Address-23 and Address-24 are not necessary and the taking of those not necessary delay time will be canceled automatically.

5.Have controller function for multiple kinds of interface(RS232,GPIB…) ,and the design of application program for GPIB-CONTROLLER will be easy.
The interface of RS232 will usually be used by the project of electric experiment, most especially, the instrument and the supplement circuit board which is only with the interface of RS232.
KI-GB1201 or KI-GCx201 series of GPIB-CONTROLLER have multiple kinds of controller functions, such as: interfaces GPIB, RS232 and Digital input, output and I/O control pins, and the hardware of GPIB-CONTROLLER will be shared by all those interfaces.
The automatic-test-fixture may contain the instruments, equipments and supplementary-circuit-board with interfaces of GPIB or RS232.
The PC application program design for that automatic-test-fixture will be difficult;however,it take the KI-GB1201 or KI-GCx201 series of GPIB-CONTROLLER link the interface of GPIB, RS232 and Digital input, output and I/O control pin, and then get the control for all those interfaces through PC with a GPIB application program, under this situation, this GPIB application program design will have little restrictions and more supports on the selection of tool, such as VISUAL C++,C#, VISUAL BASIC and so forth, and the whole process for it will become easy. There is a project which link many devices with GPIB or RS232 interface and need external Control-PIN. From the point of view of the program-design-cost and the performance and efficient of hardware, it should be a very good idea to use the KI-GB1201 or KI-GCx201 series of GPIB- CONTROLLER for running the project, for example as below :
a simple program of PC can control a Barcode-reader with the RS232 and multiple instruments with the GPIB and support control pin for circuit board of private design through KI-GB1201R.


Block diagram for the application of Barcode-reader with KI-GB1201R
If the rs232-port of barcode-scanner is DTE, Cross-Link-Adaptor as below will be needed by the linking of the Barcode-Scanner to KI-GB1201R
To barcode scanner or the adaptor
BARCODE-SCANNER
For example : Model No. ST-66 made and ssold by the market maker
The Controller input of USB-JACK should be linked with the USB socket of PC
GPIB connector,connected to external instruments with GPIB interface
USB to GPIB,Digital-I/O and RS232 Controller board KI-GB1201R
RS232-DTE-PORTA
IC-SM59264C40J-PLCC
Gnd
RTS
DTR
CONNECTOR
1N4148
2N3904
GND
LEG-5
78L05
To THE 7.5VPP
5V
LED
4.5V
PCB IS ADDED WITH LED AND WILL BE TESTED
CONNECTOR-3P

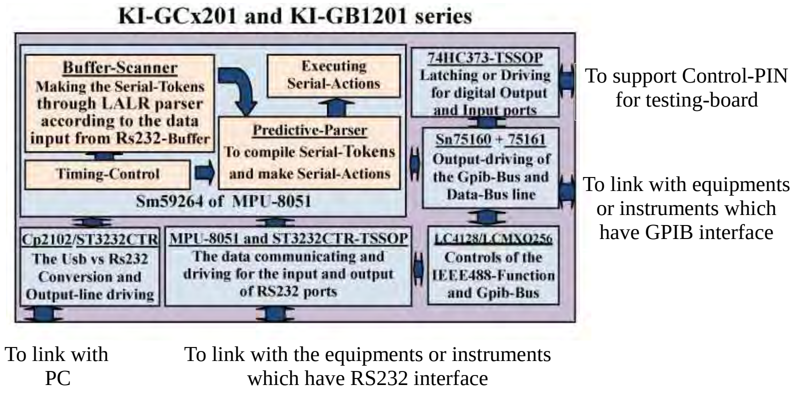
6.Watch-Dog-Controller:
Watch-Dog-Controller will be continuously monitoring the working of the CPU SM59264 and CP2102. if those CPU are interfered by the Electro-Magnetic and crashed and that crash is detected by the Watch-Dog-Controller. The Watch-Dog-Controller will be re-initial the CPU SM59264 and CP2102.

7.High performance and quality parts are used.
   a. The products of Lattice Semiconductor Corporation:
      LC-4128V-75T128C :
      (Firmware of CPLD is synthesized by tool of ABEL or VHDL)
                                  or
      LCMXO256 : (Firmware of FPGA is synthesized by tool of VHDL)
   b. The products of Silicon Labs
      CP2102 : (USB to UART Converter)
   c. The products of SyncMOS Technologies Inc. :
      SM59264 (MPU 8051)

8.Both designing and testing are standardized
   a.The firmware of main parts for GPIB-CONTROLLER are synthesized by tool of **VHDL** and **C**. Even if main parts are replaced with new type of parts in future, firmware for the new type of parts can be easy obtained from re-synthesizing the **VHDL** and **C** source code of the firmware, and then the function and performance of GPIB-CONTROLLER would be guarantied not to be changed.
   b.The full-automatic-testing-program KiGirax.exe is used to test GPIB-CONTROLLER and guarantee the quality of GPIB-CONTROLLER.

   The main functions for the KiGirax.exe
   1. To transmite instruction to GPIB-CONTROLLER
   2. To receive data from GPIB-CONTROLLER and show the attribute about the received-data as follows on the main window of KiGirax.exe
      a. Quantities for the received-data
         (it will be used to check the Quantities for the BLOCK-Data).
      b. The hex code of Check-Sum for the received-data. It can be used to get the message whether the received-data is ok or not ok.
      c. The codes of ANSI/ASCII and Hex for received-data.
   3. To execute the global-test-procedure together with single or multiple instruments for GPIB-CONTROLLER.
   ...

9.The BLOCK-DIAGRAM for KI-GCx201 and KI-GB1201 series GPIB-CONTROLLER and the Operation of GPIB-CONTROLLER Visual Serial Communication PORT are described as follows:
   The PC driver of CP2102 will make a Visual Serial Communication PORT of 'COMn:12800,n,8,1' for PC-Operation-System, and the design of the PC application program for the GPIB-CONTROLLER is same with that for the Serial-Port of RS232.


KI-GCx201 and KI-GB1201 series
Buffer-Scanner
Making the Serial-Tokens through LALR parser according to the data input from Rs232-Buffer
Executing Serial-Actions
74HC373-TSSOP
Latching or Driving for digital Output and Input ports
To support Control-PIN for testing-board
Predictive-Parser
To compile Serial-Tokens and make Serial-Actions
Timing-Control
Sn75160 + 75161
Output-driving of the Gpib-Bus and Data-Bus line
To link with equipments or instruments which have GPIB interface
Sm59264 of MPU-8051
Cp2102/ST3232CTR
The Usb vs Rs232 Conversion and Output-line driving
MPU-8051 and ST3232CTR-TSSOP
The data communicating and driving for the input and output of RS232 ports
LC4128/LCMXO256
Controls of the IEEE488-Function and Gpib-Bus
To link with PC
To link with the equipments or instruments which have RS232 interface

The specification of GPIB-CONTROLLER

The Electrical specification of KI-GC1201

Power voltage	: DC 5V (Using the power of USB-PORT)
Power current	: DC 200 mA (typical) and DC 250 mA (Max)
Input-Interface	: USB 2.0
Output-Interface	: IEEE488-GPIB-CONTROLLER (GPIB SOCKET of type-male).

Support to GPIB-CONTROLLER-Function :
SH1, AH1, L4, T6, C1, C3, C26

Unsupport to GPIB-CONTROLLER-Message :
TCT

Usable GPIB-Address for external instrument:
from 1 to 30

Serial RS232 protocol for the Input-Interface of USB:
COMn:128000,n,8,1 and None-Handshake

The Electrical specification of KI-GB1201

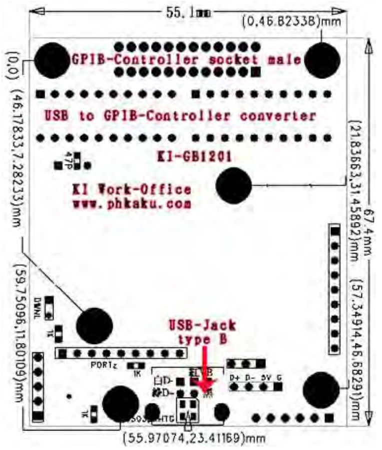
Power voltage	: DC 5V (Using the power of USB-PORT)
Power current	: DC 200 mA (typical) and DC 250 mA (Max)
Input-Interface	: USB 2.0
Output-Interface	: IEEE488-GPIB-CONTROLLER (GPIB SOCKET of type-male).

Support to GPIB-CONTROLLER-Function :
SH1, AH1, L4, T6, C1, C3, C26

Unsupport to GPIB-CONTROLLER-Message :
TCT

Usable GPIB-Address for external instrument:
from 1 to 30

Serial RS232 protocol for the Input-Interface of USB:
COMn:128000,n,8,1 and None-Handshake



The Electrical specification of KI-GB1201X

Power voltage	: DC 5V (Using the power of USB-PORT)
Power current	: DC 200 mA (typical) and DC 300 mA (Max)
Input-Interface	: USB 2.0
Output-Interface	: IEEE488-GPIB-CONTROLLER (socket male) and the Digital Input and Output port.

Support to GPIB-CONTROLLER-Function :

SH1, AH1, L4, T6, C1, C3, C26

Unsupport to GPIB-CONTROLLER-Message :

TCT

Usable GPIB-Address for external instrument:

from 1 to 29

Non-GPIB-CONTROLLER-Function for gpib-address 30:

SH, AH, TE, LE

One 2 bit Digital-Output-Ports :

Control pin : It is the RTS and DTR pin of the Input-Interface of USB

Output current : 3 mA/Max (Every Pin of Portx)

Output voltage : 3.3 V (No Load, the electrical impedance is 1K ohm.)

Note: Each output of these two pin is linked through a 1K ohm resistor.

One 4 bit Digital-Input-Ports :

Control pin : Two bits is the RI and DSR pin for the Input-Interface of USB, and the other two bits is the GPIB primary address 30 with secondary address 6 and 7

Iutput voltage : TTL

Note: Each output of these four pin is linked through a 1K ohm resistor

Two 8 bit Digital-Output-Port (with the function of latch):

Port address of PORTx and PORTy : Address primary 30 secondary 1 and 4

Output current	: 3 mA/Max (Every Pin of Porta and Portb)
Output voltage	: 5 V

One 8 bit TTL Digital-I/O port:

Port address of Portz	: Address primary 30 secondary 5
Output Voltage	: 5 V/Max

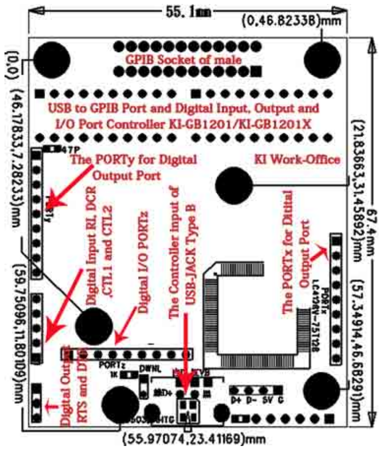
Note:

This multiple function port can also be customized as the SPI and I2C controller of the ADC, DAC and EEPROM… chip and have the GPIB address with primary 30 and secondary from 10 to 30 based on the type of chip, for example:

Address primary 30 and secondary 10 will be configured as AD5302ARMZ.
Address primary 30 and secondary 11 will be configured as AD5304BRM
… etc.

Serial RS232 protocol for the Input-Interface of USB:

COMn:128000,n,8,1 and None-Handshake



The Electrical specification of KI-GB1201R

Power voltage	: DC 5V (Using the power of USB-PORT)
Power current	: DC 200 mA (typical), and DC 300 mA (Max)
Input-Interface	: USB 2.0
Output-Interface	: IEEE488-GPIB-CONTROLLER (socket male) and RS232 Port and Digital Output and Input Control pin.

Support to GPIB-CONTROLLER-Function :

SH1, AH1, L4, T6, C1, C3, C26

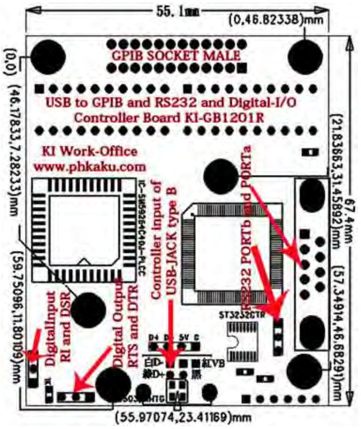
Unsupport to GPIB-CONTROLLER-Message :

TCT

Usable GPIB-Address for external instrument:

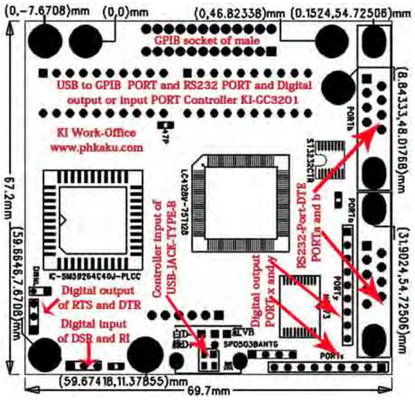
from 1 to 29

```
Non-GPIB-CONTROLLER-Function for gpib-address 30:
       SH, AH, TE, LE
One 2 bit Digital-Output-Ports :
  Control pin    : It is the RTS and DTR pin of the Input-Interface of USB
  Output current : 3 mA/Max (Every Pin of Portx)
  Output voltage : 3.3 V (No Load, the electrical impedance is 1K ohm.)
  Note: Each output of these two pin is linked through a 1K ohm resistor.
One 2 bit Digital-Iutput-Ports :
  Control pin    : It is the RI and DSR pin of the Input-Interface of USB
  Iutput voltage : TTL
  Note: Each iutput of these two pin is linked through a 1K ohm resistor.
Two RS232-Ports (Drived by IC of st3232ctr):
  Port address            : Address primary 30 secondary 2 and 3
  Method of connector     : It is DTE (RXD:Pin2 TXD:Pin3) for PORTa
  Size of input-buffer    : 748 bytes
  Communication procotol  :
    Handshake     : none.
    Baudrate      : 2400,9600(default),14400,19200,28800,
                    38400,57600,115200,128000
    Parity        : none
    Bit-No        : 8
    Stop-Bit-No   : 1
Serial RS232 protocol for the Input-Interface of USB:
    COMn:128000,n,8,1 and None-Handshake
```



```
The Electrical specification of KI-GC3201
  Power voltage        : DC 5V (Using the power of USB-PORT)
  Power current        : DC 200 mA (typical)
                         DC 300 mA (Max)
  Input-Interface      : USB 2.0
  Output-Interface     : IEEE488-GPIB-CONTROLLER (socket male)
```

```
                     and RS232 and Digital-Output port.
Support to GPIB-CONTROLLER-Function :
       SH1, AH1, L4, T6, C1, C3, C26
Unsupport to GPIB-CONTROLLER-Message :
       TCT
Usable GPIB-Address for external instrument:
       from 1 to 29
Non-GPIB-CONTROLLER-Function for gpib-address 30:
       SH, AH, TE, LE
One 2 bit Digital-Output-Port :
  Control pin    : It is the RTS and DTR pin of the Input-Interface of USB
  Output current : 3 mA/Max (Every Pin of Portx)
  Output voltage : 3.3 V (No Load, the electrical impedance is 1K ohm.)
  Note: Each output of these two pin is linked through a 1K ohm resistor.
One 2 bit Digital-Input-Port :
  Control pin    : It is the RI and DSR pin of the Input-Interface of USB
  Iutput voltage : TTL
  Note: Each output of these two pin is linked through a 1K ohm resistor.
Two 8 bit Digital-Output-Port (with the function of latch):
  Port address of PORTx and y : Address primary 30 secondary 4 and 5
  Output current              : 3 mA/Max (Every Pin of Portx)
  Output voltage              : 5 V
Two RS232-Port (Drived by IC of st3232ctr):
  Port address of PORTa and b : Address primary 30 secondary 2 and 3
  Method of connector         : DTE (RXD:Pin2 TXD:Pin3)
  Size of input-buffer        : 748
  Communication procotol  :
    Handshake    : none
    Baudrate     : 2400,9600(default),14400,19200,28800,
                   38400,57600,115200,128000
    Parity       : none
    Bit-No       : 8
    Stop-Bit-No  : 1
Serial RS232 protocol for the Input-Interface of USB:
  COMn:128000,n,8,1 and None-Handshake
```



The Electrical specification of KI-GX1201

Power voltage(Input from External) :

Power voltage : from DC 6.8V to DC 7.5V

Power current : from DC 200 mA to DC 250 mA

Input-Interface : RS232

Output-Interface : IEEE488-GPIB-CONTROLLER (GPIB SOCKET of type-male).

Support to GPIB-CONTROLLER-Function :

SH1, AH1, L4, T6, C1, C3, C26

Unsupport to GPIB-CONTROLLER-Message :

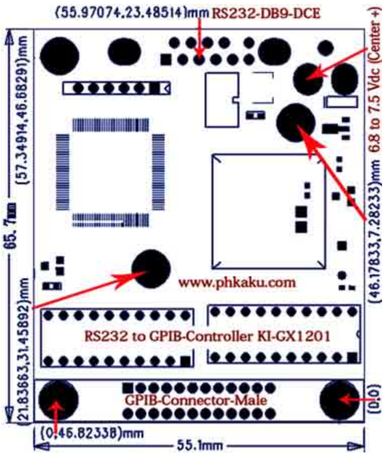
TCT

Usable GPIB-Address for external instrument:

from 1 to 30

Serial RS232 protocol for the Input-Interface of RS232:

COMn:128000,n,8,1 and None-Handshake



The Electrical specification of KI-GX3201

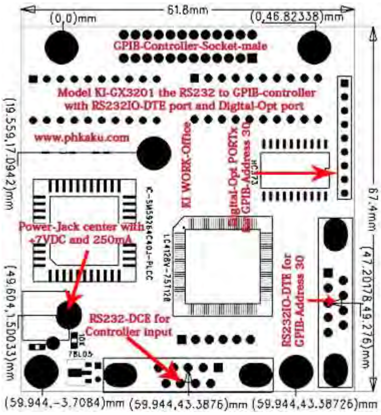
```
   Power voltage(Input from External) :
     Power voltage       : from DC 6.8V to DC 7.5V
     Power current       : from DC 200 mA to DC 250 mA
   Input-Interface       : RS232
   Power current         : DC 200 mA (typical)
                           DC 300 mA (Max)
   Input-Interface       : USB 2.0
   Output-Interface      : IEEE488-GPIB-CONTROLLER (socket male)
                           and RS232 and Digital-Output port.
   Support to GPIB-CONTROLLER-Function :
         SH1, AH1, L4, T6, C1, C3, C26
   Unsupport to GPIB-CONTROLLER-Message :
         TCT
   Usable GPIB-Address for external instrument:
         from 1 to 29
   Non-GPIB-CONTROLLER-Function for gpib-address 30:
         SH, AH, TE, LE
   One 8 bit Digital-Output-Ports (with the function of latch):
     Port address of Portx    : Address primary 30 secondary 1
     Output current           : 3 mA/Max (Every Pin of Portx)
     Output voltage           : 5 V
   One RS232-Port (Drived by IC of st3232ctr):
     Port address of Porta    : Address primary 30 secondary 2
     Method of connector      : DTE (RXD:Pin2 TXD:Pin3)
     Size of input-buffer     : 748
```

Communication procotol :

Handshake	: none
Baudrate	: 2400,9600(default),14400,19200,28800, 38400,57600,115200,128000
Parity	: none
Bit-No	: 8
Stop-Bit-No	: 1

Serial RS232 protocol for the Input-Interface of RS232:

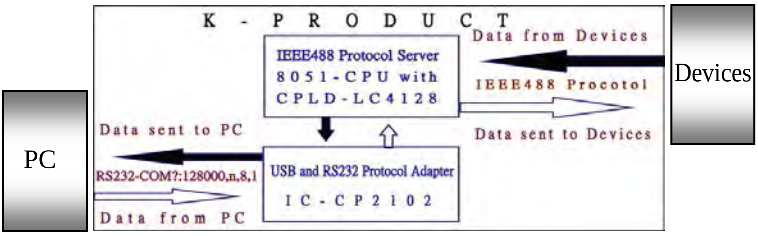
COMn:128000,n,8,1 and None-Handshake



61.8mm
(0,0)mm
(0,46.82338)mm
GPIB-Controller-Socket-male
Model KI-GX3201 the RS232 to GPIB-controller with RS232IO-DTE port and Digital-Opt port
www.phkaku.com
KI WORK-Office
Digital-Opt PORTx for GPIB-Address 30
Power-Jack center with +7VDC and 250mA
RS232-DCE for Controller input
RS232IO-DTE for GPIB-Address 30
(19.559,17.0942)mm
(49.604,1.50033)mm
67.4mm
(47.20178,49.276)mm
(59.944,-3.7084)mm
(59.944,43.3876)mm
(59.944,43.38726)mm

Introduction

The idea of GPIB-CONTROLLER is for improvement of designing application program and promotion of various functions for instruments,the illustration of signal input and output as below:



According to diagram shown as above,
The **IC-CP2102** and its driver (made and designed by SILICON LABORATORIES) and **RS232** protocol are used to act as the bridge between GPIB-CONTROLLER and **PC** for communication(left side of diagram), also the **IC8051-CPU**,**LC4128**(CPLD) as well as **IEEE488** are utilized as correspondence between GPIB-CONTROLLER and instruments, all new design is made to re-adjust and integrate the inside of GPIB-CONTROLLER to supply the functions of communication for requirement.
owing to **RS232** protocol with left side of GPIB-CONTROLLER are very general and applicable for most **PC** operating systems, in addition, the instructions of

GPIB-CONTROLLER have powerful functions and easy use after integration and re-adjustment of internal GPIB-CONTROLLER, so there are many features stated as below :

(1) Compatibility

Details referred to item 2-2.

(2) Efficient upward in data process

IEEE488.1 communicating protocol used by the GPIB-BUS side of GPIB-CONTROLLER, have the merits as below.

(a) The data can be transmitted fast to each instrument from **PC**, and also data in the output buffer of each instrument can be transmitted back to **PC** quickly.

(b) Multiple instruments can be efficiently controlled at the same times by **PC** through GPIB-CONTROLLER

(3) Application program can be designed with simplification.

GPIB-CONTROLLER is inclusive of the functions of both **IEEE488.1** and **IEEE488.2**, so it is not necessary for **PC** program designer to waste much time for making attention about the communication between **USB** and **GPIB**, in addition there are the merits of RS232 as following.

(a) The RS232 is used very generally for communicating between **PC** and various instruments

(b) There are many tool programs to support the design of RS232`s application program.

(c) **The design of Application program of RS232** is simple. Most of **PC** programmers have technology to easily write RS232 applications programs therefore the GPIB-CONTROLLER is used to control many instruments to make full operation of the equipments and obtain performance for cost-down.

1. Confirm as following items before GPIB-CONTROLLER is used

1-1 The attached Accessories for GPIB-CONTROLLER

(a) The body of GPIB-CONTROLLER Interface.

(b) CD copied with DATA.

1-2 It is necessary for cable to link. The GPIB-CABLE used for connection of Multiple-GPIB-INSTRUMENT is not included in the accessories of GPIB-CONTROLLER. Please purchase it from other professional manufacturer.

2. How to Connect PC to GPIB-CONTROLLER

2-1 PC with its accessories must support **USB** interface.

2-2 The combination of PC operating system and driver should be compatible.

GPIB-CONTROLLER and the PC operation system

The operating systems (1)- (9) listed as below are all applicable to the GPIB-CONTROLLER. It is necessary to confirm that:

(A) PC with its accessories has to support the **USB** interface.

(B) Operating system used by **PC** has to support **USB** interface.

(C) Driver installed in **PC** is required to match the operating system and GPIB-CONTROLLER connecting with **PC**. In case, the operating system used with **PC** is one of operating systems stated as below, it is applicable for GPIB-CONTROLLER.

(1) **Microsoft Windows 98**

(2) **Windows 2000**

(3) **Windows XP**

(4) **Windows SE**

(5) **Windows CE50/CE60**

(6) **Window Vista**

(7) Windows 7
(8) Windows 8
(9) MAC OS-9
(10) MAC OS-X
(11) Linux 2.40/3.xx

* The usb side of GPIB-CONTROLLER for PC utilize Usb_Default-ID (Vendor-ID:10C4 Product-ID:EA60) supplied by the manufacturer of the **IC-CP2102 (SILICON LABORATORIES)**, the driver which take use for PC to be installed the operating systems as listed above will be also supplied **SILICON LABORATORIES**, and has been burned in CD attached as accessories of GPIB-CONTROLLER.

* If there are an announcement of new operating system except lists as above and also its new driver put out by **SILICON LABORATORIES,** the information of new driver for the **IC-CP2102** would be taken, please refer to its website and try to use the **Key-Word** for **Web-Searching,** such as : **`cp2102 driver download`**

* Restriction: because the Vendor-ID or Product-ID for each of usb-interface in a PC must be different therefore:

(1)if other usb-interface with PC also take use for the Usb-Default-ID (Vendor-ID:10C4 Product-ID:EA60) supplied by **SILICON LABORATORIES** the other usb-interface would be moved out and not work with GPIB-CONTROLLER at the same time.

(2) Each one of PC is able to link with a GPIB-CONTROLLER only. Total **gpib** primary-address used by multi-instruments connected with GPIB-CONTROLLER must not be more than 30 pieces of primary-address, in general, an instrument needs a primary address only, if it is necessary for **gpib** to control a lot of **Digital-IO-PIN**, in order to reduce the consumption of primary-address by utilizing secondary-address. Each primary-address is able to control 30 secondary-address, and a secondary-address may control 8 pieces of **Digital-IO-PIN** at least through the GPIO interface, in a word **a primary-address is able to control 240 (240 = 8*30) pieces of Digital-IO-PIN**.

2-3 Driver install

GPIB-CONTROLLER driver installation

The **USB** plug of GPIB-CONTROLLER is directly linked to the **USB** socket of **PC**, in case additional **USB-Extension-Cable** or **Multiple-USB-Extension-Socket** are used, it will make **USB-Bus-Power-Voltage** dropped down, once Bus-Power-voltage is decreasing to less than **4.8** volt, interface could not operate usually. So avoid dropping over **0.2** volt for voltage-value-on-cable as possible, **(**suggest that the length of cable is shorter than that required, and the diameter of cable wire is as bigger as possible**)**.

2-3-1 Choose the installation of the driver:

Download the GPIB-CONTROLLER Driver

The driver setup in compliance with the operating system used by **PC**.
For example:
E:\Driver\CP210x VCP Win XPS2K3Vista 7\ CP210x_VCP_Win_XP_S2K3_Vista_7.exe.

2-3-2 Confirm the result of installing driver: Download KIGirax.exe

(1) Restart PC, and check items in order as below:
Start->Set(S)->Control Panel(C)->System->Hard ware->Device manager ->Port(COM and LPT)->Silicon Labs Cp210x USB to UART Bridge(COMn).
indicated as procedure above, the driver is already installed successfully.

(2) Other items

(a) PC is restarted after the driver set-up already and will appear the strings of "...COMn" , 'n' which is the last letter of `COMn` to stand for one of number from 1 to 256 and will be regarded as ID No. of New Communication Port to be set up.

(b) In case, the driver (stated as item 2-3-1) was already installed in PC, because other application programs also needed to use this driver, this step (item 3-1) can be omitted.

(c) The instruction `*idn?` (Please refer to item **5-4-2)** can be used to confirm the correct connection between **PC** and GPIB-CONTROLLER.

3. GPIB-CONTROLLER and Instrument Linking

3-1 GPIB-CONTROLLER connect to **PC** and single GPIB-INSTRUMENT

3-2 GPIB-CONTROLLER connect to **PC** and multiple GPIB-INSTRUMENTS
note: in order to minimize stress on the connector mountings, the quantity and weight of stacking cable connector blocks should be careful.

4. Choose the tool programs to design the application programs

Tool programs can be used to design the application program as following:

(1) **VB**(Visual Basic)
(2) **VC++**(Visual C++), VC#(Visual C#)
(3) **BORLAND C++**
(4) **Latview** (Because Latview support controls for RS232-COMMUNICATION)
(5) Other programming tools which support RS232-Serial-Port-Communication can be taken.

Summary:

There are five steps for linking process of the **PC**, drivers and GPIB-CONTROLLER summarized as below:

STEP	Description	Detail referred to item
1	**PC** have **USB** interface	2-1
2	Operating system used with **PC as requirement** notified	2-2
3	The **USB** plug of GPIB-CONTROLLER must be directly linked to the **USB** socket of **PC** before installing driver	2-3
4	**PC**, GPIB-CONTROLLER and **Instrument** Linking	3
5	The instructions of『**IDN?**』and『**FindListen?**』used to test the linking of **PC** to GPIB-CONTROLLER with instrument whether operating smoothly or not	2-3-c, 5.4.1
6	Application program designed with instruction of GPIB-CONTROLLER	4, 5

5.Instruction for GPIB-CONTROLLER

The list of abbreviation---common words.

Abbreviation	Description	Rem
HexToBin	Action taken to translate data-byte of **dStr** from Hex format into Binary format	
T-Block-Data	**Block-Data** given from **PC** will be sent to instrument by GPIB-CONTROLLER	
T-Block-Message	**Block-Message** given from **PC** will be sent to instrument by GPIB-CONTROLLER	

IDAT	It is Internal-Device-Address-Table for GPIB-CONTROLLER, With the `**Findlisten**` instruction, the addresses of all the instruments connected with GPIB-CONTROLLER through **GPIB-CABLE** will be registered in **IDAT.**	
iaB	It is internal-array-buffer of GPIB-CONTROLLER. Maximum bytes of buffer size to be programmed are 10238 bytes.	
Ist	IEEE488.1 message of individual-status-bit such as: PPR1, PPR2, PPR3, PPR4, PPR5, PPR6, PPR7, PPR8	

5.1 Instruments utilized for testing

Instruments Model-No listed below are used as the basic standard test equipments for all of the examples as taken below item 5.2.
If the other instruments Model-No are utilized, please refer to the User Manual for those instruments.

	Instrument Model No.	Makers
1	34410a	Agilent Technologies
2	HEWLETT-PACKARD, 6611C	Agilent Technologies
3	DSOX2012A	Agilent Technologies
4	E5071B	Agilent Technologies
5	HEWLETT PACKARD, 8752A	Agilent Technologies
6	HEWLETT PACKARD, 8753C	Agilent Technologies
7	HEWLETT PACKARD, 8753D	Agilent Technologies
8	HEWLETT-PACKARD, 54520A	Agilent Technologies
9	E4418A	Agilent Technologies
10	HEWLETT-PACKARD, 54820A	Agilent Technologies
11	HP8591EM	Agilent Technologies
12	33220A	Agilent Technologies
13	E5515C	Agilent Technologies
14	HP81101A	Agilent Technologies
15	HP6623A	Agilent Technologies
16	Hewlett-Packard, E4402B	Agilent Technologies
17	HEWLETT-PACKARD E3632A	Agilent Technologies
18	HEWLETT-PACKARD 34401A	Agilent Technologies
19	HEWLETT-PACKARD 83620A	Agilent Technologies
20	HEWLETT-PACKARD 8720D	Agilent Technologies
21	HEWLETT-PACKARD 8648D	Agilent Technologies
22	HEWLETT-PACKARD E3631A	Agilent Technologies
23	R3131	ADVANTEST
24	R3162	ADVANTEST
25	R3273	ADVANTEST
26	MT9810A	ANRITSU
27	MODEL 2000	KEITHLEY INSTRUMENTS INC.
28	2031	MARCONI INSTRUMENTS
29	NRVD	ROHDE&SCHWARZ
30	SME03	ROHDE&SCHWARZ
31	SMT06	ROHDE&SCHWARZ
32	TDS 220	TEKTRONIX
33	AFG3102	TEKTRONIX
34	PPT-1830	GOOD WILL
35	4235	WAYNE KERR

5.2 Procotol rule for command

(a) Each of instruments connected to the same GPIB-CONTROLLER must be configured to have different addresses for **GPIB** of instrument.
(b) The **parameter** of each instruction must be less than **228 bytes**.
(c) The length of each **Command** must be less than **1024 bytes**.
(d) The **parameter** pieces of a **Command** must not be more than **40** pieces.
(e) The maximum length of programming for **iaB** is **10238 bytes**.
(f) The communication-protocol between **PC** and GPIB-CONTROLLER is **"COM?:128000,N,8,1"**.
(g) A lot of block data can be read from **An** by various instructions as

below:
***ARBITRARY?** or **RdARBITRARY?** or **AgiETB?** or **RdAgiETB?** or **IEEEB?** or **RdIEEEB?**
Instructions plus Key-Symbol '', Please refer to Item 5-4-4,5-4-5, 5-4-6 and the example of model DXO-X2012A for page 22, As well as the example of Model 8753D for page 20.
*The instructions AgiETB or RdAgiETB or IEEEB or RdIEEEB added with Key-Symbol `<` are given from PC to GPIB-CONTROLLER. And then PC read data through GPIB-CONTROLLER from output-buffer of instrument except Block-Head (such as instruction IEEEB< 9 SYST:SET? of model DSOX2012A) details please refer to Example 8753D and example DSOX2012A
*Instructions SEND … with Key-Symbol '\', '|' or ':' are different in the function for transmission of dStr described as below:

	dStr[1]	dStr[2]… dStr[n-2]	dStr[n-1]	dStr[n]
\	HexToBin on T-Block-Data			
\	T-Block-Data	HexToBin on T-Block-Data (dStr[2])+EOI		
\	T-Block-Data+EOI		T-Block-Data	HexToBin on T-Block-Data+EOI
|	T-Block-Data			
|	T-Block-Data+EOI	T-Block-Data		
:	HexToBin on T-Block-Data			
:	T-Block-Data	HexToBin on T-Block-Data (dStr[2])		
:	T-Block-Data+EOI		T-Block-Data	HexToBin on T-Block-Data

(h) If including many dStr are used in an instruction, in which some of dStr are added with Key-Symbol `?` and other are not, i.e. some of dStr may have query-instruction and other have not, in this situation there are different functions between Key-Symbol `=` and `?` examples taken as below (instruction of HP8591EM) for description:
(1) the function of
Send= 18 'IP;''ID?;''CF 300MHZ;SP 2MHZ;RB 100KHZ;''CF?''rb?'
are same as
Send ? 18 'IP;''ID?;''CF 300MHZ;SP 2MHZ;RB 100KHZ;''CF?''rb?''ID?'
given to An and then data read automatically from An, in the same way as `CF?` and `rb?` given to An, data also read automatically from An, all of data read from An will be responded back to PC.
(2) Send= 18 'IP;''CF 300MHZ;SP 2MHZ;RB 100KHZ;''CF''rb'.There are no action of reading data from An, owing to dStr without Key-Symbol `?`.
(3) Send? 18 'IP;''CF 300MHZ;SP 2MHZ;RB 100KHZ;''CF''rb', have the same function as
Send? 18 'IP;''CF 300MHZ;SP 2MHZ;RB 100KHZ;''CF''rb?',
'rb' or 'rb?' given to An from PC and then data from instrument read automatically, it is meaning that there are one more actions taken of reading data from An when instruction is used for Key-Symbol `?`
(i) If both Ascii-format-data and the data of Hex format are included in an instruction, Key-Symbol '\' can be used in this situation, the data of Hex format will be automatically translated into the data of binary format which will be sent to instrument An or AAn. For example (Instruction of the HP8591EM and33220A):
SEND\ 18 'IB''7383961626…';
SEND\ 1 'DATA:DAC VOLATILE, #216''07FF0600040002000000FE00FA00F801';
Description:

Asc(D),Asc(A)…Asc(6),7,255,6…etc. of 39 bytes will be transmitted to the instrument of Address-1(Function Generator 33220A) through GPIB-CONTROLLER.

(j) If Key-Symbol '>' is not used in instructions, the data of dStr would be transmitted to instruments with normal speed, which are properly for all instruments to work. Instruction with Key-Symbol '>' are able to make the speed-up when to transmit the data of dStr to instrument, which is suitable for transmitting a lot of large -block data at the same time, it is a good selection for transmitting large-Block-Data, but the condition of instrument working speed must be taken into consideration, there are some of instruments for old model, such as :'HEWLETT PACKARD,8752A', which will be unable to use the Key-Symbol '>',because of working speed.

(k)
Instruction is add with Key-Symbol '!'
If Key-Symbol '!' is used with instruction, contents of iaB will be deleted at first, and then the normal function of instruction executed.

(l) The number of primary address for GPIB of instrument must be one of number from 1 to 30,If there are the number of secondary address should be also one of number from 1 to 30, the address set-up of primary and secondary will depend on the functions of instrument as below,
(1) If instrument is with function for T and L of IEEE488.1, instruction for the primary address will be accepted and secondary address will be discarded by the instrument.
(2) If instrument is with function for TE and LE of IEEE488.1, instruction for both both primary and secondary address will be accepted by the instrument.

GPIB Address format
Example for address for GPIB of instrument **set-up as below.**

Number	Address of GPIB		
	Data-of-PIO	Secondary	Primary
417	0	4	17
1551204	155	12	4
9	0	0	9

5.3 The Structure of GPIB-CONTROLLER`s Command

Following are the basic structure of the GPIB-CONTROLLER **Command** discribed with **BNF**(Backus-Normal Form).

5.3.01:

Command ::= Instruction CrLf | Instruction CMD_EXT CrLf

Description:
The format which is accepted by GPIB-CONTROLLER is the **Command**, all of the data transmitted to GPIB-CONTROLLER From **PC** must be in accordance with the regulation of command.

CMD_Ext	::=	Semicolon Instruction CMD_Ext | Lamda
Instruction	::=	Key-Word Key-Symbol_Ext PARAM_EXT | Lamda

Description:
Instruction will be described in details as item 5-4-x below.

KySym_Ext	::=	Key-Symbol KySym_ Ext | Lamda
PARAM_EXT	::=	PARAMETER PARAM_EXT | Lamda
CrLf	::=	0d | 0a | 0d+0a | 0a+0d
An_Ext	::=	An An_Ext | Lamda
dStr_Ext	::=	dStrdStr_Ext | Lamda
Semicolon	::=	;
Lamda	::=	nothing (NULL)

MS ::= Unit as time-delay measured by 1 millisecond(about)
RQS ::= Signal of IEEE488.1 (Request service)
EOI ::= Signal of IEEE488.1 (End of identity)
EOS ::= 0a | (Combination of ascii without excess of two piece) | 0d+0a | Lamda

Description:
EOS(End of String) is one of IEEE488 Data which is part of **RDT** ,and instruction (EOSO, RdEOS, SetRDT) is able to make **EOS** change.

DABE ::= EOI
RDT ::= EOS EOI | DABE

Description:
RDT (Response Data Terminator) is a kind of IEEE488-Data which is placed at the end of the **Block-Data** transmitted to GPIB-CONTROLLER from the output-buffer of instrument

PEOS ::= 0a | 0d+0a | EOI | Combination without excess of eleven pieces of ASCII | Lamda
PMT ::= PEOS EOI | DABE

Block-Data ::= Stand for the combination with one or more ASCII

Escaped sequence rule for the input of GPIB-CONTROLLER Block-Data
(Supported by Version 2.1 and further only)

Data	**Data represented in Block-Data**
nnn(ascii value)	\nnn (nnn is decimal and <= 255)
“	\”
‘	\’
\	\\
0d (hex)	\r
0a (hex)	\n
09 (hex)	\t
07 (hex)	\a
08 (hex)	\b
0c (hex)	\f
0b (hex)	\v
06 (hex)	\k
15 (hex)	\u
03 (hex)	\c
04 (hex)	\d
05 (hex)	\e
0e (hex)	\o
0f (hex)	\i
16 (hex)	\y
1a (hex)	\z
18 (hex)	\x
17 (hex)	\w
01 (hex)	\h
1b (hex)	\s
10 (hex)	\p
11 (hex)	\q
12 (hex)	\j
13 (hex)	\l
19 (hex)	\m
1d (hex)	\g

Block-Message ::= Block-Data PMT

Description:

GPIB-CONTROLLER received the **Command** from **PC** and pick **Block-Data** out from **Command**, the **Block-Data** added with `PMT` become Block-Message which are actual data transmitted from GPIB-CONTROLLER to instrument

5.3.02:

```
Key-Word ::=  "FindListen"  | "IDN"     | "SPOLL"       | "PP"       |
              "SEND"        | "EOSO"    | "TRANSCEIV"   | "AgiETB"   |
              "ARBITRARY"   | "IEEEB"   | "IEEEidn"     | "TestSys"  |
              "AgiETBW"     | "READ"    | "RdEOS"       |"RdAgiETB"  |
              "RdARBITRARY" | "RdIEEEB" | "ARBITRARYW"  | "IEEEBW"   |
              "MsgREN"      | "MsgTRG"  | "MsgSDC"      | "MsgDC"    |
              "MsgIFC"      | "RESET"   | "AryAdd"      | "ArySet"   |
              "SetPMT"      | "SetRDT"  | "PIO"
```

Description:

GPIB-CONTROLLER will decide to take any action in accordance with the **Key-Word**, every letter of **Key-Word**s is able to use the Capital or small letter at it`s option. There are no influence on the function of the **Key-Word**, for example:

SEND (Cap.), idn (Small), FindListen(Cap+small)

5.3.03:

```
Key-Symbol ::= "?" | "@" | "#" | "$" | "+" | "-" | "!" | "&" | "*" | "~" |
               "^" | "|" | "%" | ">" | "<" | ":" | "\" | "/" | Lamda
```

Description:

The command is given to GPIB-CONTROLLER from **PC**. some of additional action will be taken or not in accordance with **Key-Symbol** in **Command**. Each kind of **Key-Symbol** is how to make the combination with **Key-Word**, which will be described in item of syntax (5.4.x) and instruction (5.4.x) in details.

5.3.04:

```
PARAMETER   ::=  An | AAn | dStr | dS0..dSn | aL0..aLn | aQ0..aQn |
                 DlyW | DlyR | DlyB | BLEN | Lamda
An          ::=  Number (single of instrument address which must be one of
                          number from 1 to 30)
AAn         ::=  An  An_Ext  |  Lamda
```

Description:

AAn represent the combination with null **An** ormultiple **An**. **Command** is given to GPIB-CONTROLLER from **PC**, if **Command** do not designate the **PARAMETER** of Instrument address, GPIB-CONTROLLER would transmit the Block-Data in **Command** to all of the instruments addresses connected with GPIB-CONTROLLER, i.e. **AA**n equal to all of the instruments addresses connected with GPIB-CONTROLLER

```
aL0..aLn    ::= Combination of null or One more addresses of listener
                instrument(An)
aQ0..aQn    ::= Combination of null or One more instrument addresses
                whose RQS signal is activated.
```

Description:

Command is given to GPIB-CONTROLLER from **PC**, GPIB-CONTROLLER will decide to Communicate with instrument in accordance with the **An** within **Command**

```
dStr        ::= Block-Data enclosed with single quotation mark in head and
                rear of Block-Data (i.e. `…Block-Data…` )
```

Description:

Command is given to GPIB-CONTROLLER from **PC**, the **Block-Data** within the **Command** are transmitted to instrument from **PC** through GPIB-CONTROLLER

```
dS0..dSn    ::=  dStr_Ext  |  Lamda
```

Description: Null or one more **dStr** combined

```
BLEN  ::= Lamda |
         Number
         (Number of byte of data saved in output-buffer of instrument)
GPIB-CONTROLLER instruction parameter DlyR,DlyW,DlyB
DlyW  ::= Lamda |
         Number
         (
          There are time delay occurred between the time after Command
          given to GPIB-CONTROLLER from PC and the time Before Block-Data
          transmitted to instrument from GPIB-CONTROLLER. Unit of time
          delay is one millisecond
         )
DlyR  ::= Lamda |
         Number
         (
          There are time delay occurred between the time After instruction
          given already from PC to instrumentthrough GPIB-CONTROLLER and
          the time before data read from the output-buffer of instrument,
          unit of timedelay is also one millisecond
         )
```

Description:

Delay time is that waiting time of GPIB-CONTROLLER occurred between the time after GPIB-CONTROLLER received **Command** from PC, and the time before data transmitted to instrument or data read from instrument, the length of time will be decided by **DlyW** or **DlyR**

```
DlyB  ::= Lamda  |
         Number
         (
          It is the coefficient of GPIB-BUS-POLLING time, it is meaning
          that action will be taken of reading/writing data from/to
          instrument by PC through GPIB-CONTROLLER and of polling
          continuously to instruments. The value is 3000or 7000, default
          value 3000 is used without instruction of `Testsys`, value 7000
          is used with instruction of `Testsys`
         )
```

Description:

1. Except **Key-Word,** the parameter **of An, dStr, Key-Symbol, DlyW… etc** Are used only when to require from the instruction.
2. The order or sequence among **Key-Word**, **Key-Symbol** and **Parameter** must be in accordance with syntax stated above.
3. Each instruction and syntax will be described in details as item 5-4-x below.

There are the lists of instruction structures prepared in accordance with **BNF** as below: (**DlyR** and **DlyW** and **DlyB** are omitted):

```
Instruction   ::=  IDN?
Instruction   ::=  FindListen  AAn | FindListen?  AAn | FindListen#  AAn
Instruction   ::=  SPOLL    AAn | SPOLL?  AAn | SPOLL+  AAn | SPOLL#  AAn |
                   SPOLL+?  AAn | SPOLL+# AAn
Instruction   ::=  PP  AAn dS0..dSn  |  PP?  |  PP\  |  PP~
Instruction   ::=
 Send  AAn dS0**dSn   | Send?  AAn dS0**dSn  | Send# AAn dS0**dSn   | Send?  AAn       |
 Send-  AAn dS0**dSn  | Send-?  AAn dS0**dSn | Send-# AAn dS0**dSn  | Send-?  AAn      |
 Send=  AAn dS0**dSn  | Send* AAn dS0**dSn   | Send| AAn dS0**dSn   | Send  dS0**dSn   |
 Send>  AAn dS0**dSn  | Send^ AAn dS0**dSn   | Send^= AAn dS0**dSn  | Send?            |
 Send- AAn dS0**dSn   | Send+  AAn dS0**dSn  | Send$ AAn dS0**dSn   | Send*#           |
 Send\  AAn dS0**dSn  | Send@ AAn dS0**dSn   | Send=  AAn dS0**dSn  | Send# dS0**dSn    |
 Send& AAn dS0**dSn   | Send~  AAn dS0**dSn  | Send=# AAn dS0**dSn  | Send># dS0**dSn   |
 Send/# AAn dS0**dSn  | Send@# AAn dS0**dSn  | Send$# AAn dS0**dSn  | Send\ dS0**dSn    |
 Send+  AAn dS0**dSn  | Send+?  AAn dS0**dSn | Send+# AAn dS0**dSn  | Send+?  AAn       |
 Send>!?  dS0**dSn    | Send>!# dS0**dSn     | Send>?  dS0**dSn     | Send\# dS0**dSn   |
 Send\? An dS0**dSn   | Send: An dS0**dSn
Instruction   ::=
 EOSO EOC AAn  dS0**dSn   | EOSO? EOC  AAn dS0**dSn   | EOSO# EOC AAn dS0**dSn    |
 EOSO- EOC AAn dS0**dSn   | EOSO-? EOC AAn dS0**dSn   | EOSO-# EOC AAn dS0**dSn   |
 EOSO+? EOC AAn           | EOSO+? EOC AAn            | EOSO+# EOC AAn dS0**dSn   |
 EOSO+ EOC AAn dS0**dSn   | EOSO+? EOC AAn dS0**dSn   | EOSO\ EOC dS0**dSn        |
 EOSO= EOC  AAn dS0**dSn  | EOSO* EOC AAn dS0**dSn    | EOSO? EOC  AAn            |
 EOSO| EOC AAn dS0**dSn   | EOSO> EOC  AAn dS0**dSn   | EOSO^ EOC AAn dS0**dSn    |
 EOSO^= EOC AAn dS0**dSn  | EOSO- EOC AAn dS0**dSn    | EOSO+ EOC  AAn dS0**dSn   |
 EOSO$ EOC AAn dS0**dSn   | EOSO\ EOC  AAn dS0**dSn   | EOSO@ EOC AAn dS0**dSn    |
 EOSO= EOC AAn dS0**dSn   | EOSO& EOC AAn dS0**dSn    | EOSO~ EOC AAn dS0**dSn    |
 EOSO=# EOC AAn dS0**dSn  | EOSO/# EOC AAn dS0**dSn   | EOSO@# EOC AAn dS0**dSn   |
 EOSO$# EOC AAn dS0**dSn  | EOSO EOC  dS0**dSn        | EOSO# EOC dS0**dSn        |
 EOSO>? EOC dS0**dSn      | EOSO># EOC dS0**dSn       | EOSO>!? EOC dS0**dSn      |
 EOSO>!# EOC dS0**dSn     | EOSO? EOC                 | EOSO*# EOC                |
 EOSO\# EOC dS0**dSn
Instruction   ::=
 IEEEidn? AAn   | IEEEidn#  AAn | IEEEidn$ AAn | IEEEidn$# AAn | IEEEidn?- AAn |
 IEEEidn#- AAn | IEEEidn$- AAn | IEEEidn?     |IEEEidn#       | IEEEidn$      |
 IEEEidn>$
Instruction   ::=
 TestSys?  AAn | TestSys# AAn  | TestSys$ AAn  | TestSys$# AAn | TestSys#   | TestSys$ |
 TestSys?- AAn | TestSys#- AAn | TestSys$- AAn | TestSys?      | TestSys $#
Instruction   ::=
 ARBITRARY?  BLEN  An  dS0..dSn   | ARBITRARY@#  BLEN  An  dS0..dSn   |
 ARBITRARY$# BLEN  An  dS0..dSn   | ARBITRARY!@ BLEN   An             |
 ARBITRARY^#  BLEN  An  dS0..dSn  | ARBITRARY!$# BLEN  An  dS0..dSn   |
 ARBITRARY!# BLEN  An  dS0..dSn   | ARBITRARY!@# BLEN  An  dS0..dSn   |
 ARBITRARY@ BLEN   An             | ARBITRARY!? BLEN An               |
 ARBITRARY^$# BLEN   An           | ARBITRARY^? BLEN An               |
 ARBITRARY!? BLEN   An            | ARBITRARY!# BLEN An
Instruction   ::=
 AgiETB?  An  dS0..dSn    | AgiETB>^? An  dS0..dSn   | AgiETB#  An  dS0..dSn    |
 AgiETB^# An dS0..dSn     | AgiETB!?  An  dS0..dSn   | AgiETB!# An dS0..dSn     |
 AgiETB>^$#  An  dS0..dSn | AgiETB^$# An dS0..dSn    | AgiETB!$#  An  dS0..dSn  |
 AgiETB^$# An dS0..dSn    | AgiETB>^_#  An  dS0..dSn | AgiETB^_  An dS0..dSn    |
 AgiETB<#  An  dS0..dSn   | AgiETB<  An dS0..dSn     | AgiETB_# An  dS0..dSn    |
 AgiETB_  An              | AgiETB!$#  An            | AgiETB!<  An             |
 AgiETB!_  An
Instruction   ::=
 IEEEB?  An  dS0..dSn     | IEEEB>^? An  dS0..dSn    | IEEEB#  An  dS0..dSn    |
 IEEEB^# An dS0..dSn      | IEEEB!?  An  dS0..dSn    | IEEEB!# An dS0..dSn     |
 IEEEB>^$  An  dS0..dSn   | IEEEB^$ An dS0..dSn      | IEEEB$#  An  dS0..dSn   |
 IEEEB^$# An dS0..dSn     | IEEEB>^_#  An  dS0..dSn  | IEEEB^_  An dS0..dSn    |
 IEEEB<#  An  dS0..dSn    | IEEEB<  An dS0..dSn      | IEEEB_# An  dS0..dSn    |
 IEEEB_  An               | IEEEB!$#  An             | IEEEB!<  An dS0..dSn    |
 IEEEB!_  An dS0..dSn     | IEEEB!<  An
Instruction   ::=
 TransCeiv?  AAn dS0..dSn  | TransCeiv# AAn dS0..dSn   | TransCeiv$ AAn dS0..dSn    |
 TransCeiv$# AAn dS0..dSn  | TransCeiv^# AAn dS0..dSn  | TransCeiv!# AAn dS0..dSn   |
 TransCeiv^$# AAn dS0..dSn | TransCeiv!$# AAn dS0..dSn | TransCeiv>? AAn dS0..dSn   |
 TransCeiv>#  AAn dS0..dSn | TransCeiv^# AAn dS0..dSn  | TransCeiv!# AAn dS0..dSn   |
 TransCeiv~# AAn dS0..dSn  | TransCeiv~ AAn dS0..dSn
Instruction   ::=
 Read? AAn dS0..dSn    | Read!? AAn   | Read# AAn dS0..dSn    | Read!# AAn   |
 Read?# AAn dS0..dSn   | Read!?# AAn  | Read@# AAn dS0..dSn   | Read!@# AAn  |
 Read@ AAn dS0..dSn    | Read!@  AAn  | Read!? AAn dS0..dSn   | Read!? AAn   |
 Read!# AAn dS0..dSn   | Read!# AAn   | Read!?# AAn dS0..dSn | Read!?# AAn  |
 Read!@# AAn dS0..dSn | Read!@# AAn  | Read!@ AAn dS0..dSn   | Read!@ AAn   |
 Read!?   dS0..dSn     | Read!?       | Read!#   dS0..dSn     | Read!#       |
 Read!?#  dS0..dSn     | Read!?#      | Read!@#  dS0..dSn     | Read!@#      |
 Read!@   dS0..dSn     | Read!@
Instruction   ::=
 RdEOS? EOC AAn dS0..dSn    | RdEOS!? EOC AAn            | RdEOS # EOC AAn dS0..dSn  |
 RdEOS!# EOC AAn            | RdEOS?# EOC AAn dS0..dSn   | RdEOS!?# EOC AAn          |
 RdEOS @# EOC AAn dS0..dSn  | RdEOS!@# EOC AAn           | RdEOS @ EOC AAn dS0..dSn  |
 RdEOS!@  EOC AAn           | RdEOS!? EOC AAn dS0..dSn   | RdEOS!? EOC AAn           |
 RdEOS!# EOC AAn dS0..dSn   | RdEOS!# EOC AAn            | RdEOS!?# EOC AAn dS0..dSn |
 RdEOS!?# EOC AAn           | RdEOS!@# EOC AAn dS0..dSn  | RdEOS!@# EOC AAn          |
 RdEOS!@ EOC AAn dS0..dSn   | RdEOS!@ EOC AAn            | RdEOS!? EOC dS0..dSn      |
 RdEOS!? EOC                | RdEOS!# EOC dS0..dSn       | RdEOS!# EOC               |
 RdEOS!?# EOC dS0..dSn      | RdEOS!?# EOC               | RdEOS!@# EOC dS0..dSn     |
 RdEOS!@# EOC               | RdEOS!@ EOC dS0..dSn       | RdEOS!@ EOC
Instruction   ::=
 RdAgiETB? An dS0..dSn    | RdAgiETB!? An    | RdAgiETB # An dS0..dSn    | RdAgiETB!# An  |
 RdAgiETB?# An dS0..dSn   | RdAgiETB!?# An   | RdAgiETB @# An dS0..dSn   | RdAgiETB!@# An |
 RdAgiETB @ An dS0..dSn   | RdAgiETB!@  An   | RdAgiETB!? An dS0..dSn    | RdAgiETB!? An  |
 RdAgiETB!# An dS0..dSn   | RdAgiETB!# An    | RdAgiETB!?# An dS0..dSn   | RdAgiETB!?# An |
 RdAgiETB!@# An dS0..dSn | RdAgiETB!@# An   | RdAgiETB!@ An dS0..dSn    | RdAgiETB!@ An  |
 RdAgiETB_ An  dS0..dSn   | RdAgiETB_ An     | RdAgiETB!_ An  dS0..dSn   | RdAgiETB!_ An
Instruction   ::=
 RdIEEEB? An dS0..dSn     | RdIEEEB!? An     | RdIEEEB# An dS0..dSn      | RdIEEEB!# An   |
 RdIEEEB?# An dS0..dSn    | RdIEEEB!?# An    | RdIEEEB@# An dS0..dSn     | RdIEEEB!@# An  |
 RdIEEEB@ An dS0..dSn     | RdIEEEB!@  An    | RdIEEEB!? An dS0..dSn     | RdIEEEB!? An   |
 RdIEEEB!# An dS0..dSn    | RdIEEEB!# An     | RdIEEEB!?# An dS0..dSn    | RdIEEEB!?# An  |
 RdIEEEB!@# An dS0..dSn   | RdIEEEB!@# An    | RdIEEEB!@ An dS0..dSn     | RdIEEEB!@ An   |
 RdIEEEB_ An  dS0..dSn    | RdIEEEB _ An     | RdIEEEB!_ An  dS0..dSn    | RdIEEEB!_ An
Instruction   ::=
 RdARBITRARY? BLEN An dS0..dSn      | RdARBITRARY!? BLEN An               |
 RdARBITRARY # BLEN An dS0..dSn     | RdARBITRARY?# BLEN An dS0..dSn      |
 RdARBITRARY!# BLEN An              | RdARBITRARY @# BLEN An dS0..dSn     |
 RdARBITRARY @ BLEN An dS0..dSn     | RdARBITRARY!?# BLEN An              |
 RdARBITRARY!# BLEN An dS0..dSn     | RdARBITRARY!@# BLEN An              |
 RdARBITRARY!?# BLEN An dS0..dSn    | RdARBITRARY!@# BLEN An dS0..dSn     |
 RdARBITRARY!@  BLEN An             | RdARBITRARY!@ BLEN An dS0..dSn      |
 RdARBITRARY!? BLEN An              | RdARBITRARY!# BLEN An               |
 RdARBITRARY!?# BLEN An             | RdARBITRARY!@# BLEN An              |
 RdARBITRARY!@ BLEN An              | RdARBITRARY!? BLEN An dS0..dSn
Instruction   ::=
 ARBITRARYW AAn dS0..dSn     | ARBITRARYW>@ AAn dS0..dSn  | ARBITRARYW # AAn dS0..dSn   |
 ARBITRARYW! AAn dS0..dSn    | ARBITRARYW  dS0..dSn       | ARBITRARYW > AAn dS0..dSn   |
 ARBITRARYW^ AAn dS0..dSn    | ARBITRARYW- AAn dS0..dSn   | ARBITRARYW# AAn dS0..dSn    |
 ARBITRARYW>!# dS0..dSn      | ARBITRARYW@# AAn dS0..dSn  | ARBITRARYW@ AAn dS0..dSn    |
 ARBITRARYW>! AAn dS0..dSn   | ARBITRARYW># dS0..dSn      | ARBITRARYW >|# dS0..dSn     |
 ARBITRARYWI^# dS0..dSn     | ARBITRARYWI# dS0..dSn      | ARBITRARYWI! dS0..dSn        |
Instruction   ::=
 AgiETBW AAn dS0..dSn     | AgiETBW?  AAn dS0..dSn   | AgiETBW # AAn dS0..dSn   |
 AgiETBWI AAn dS0..dSn    | AgiETBW  dS0..dSn        | AgiETBW > AAn dS0..dSn   |
 AgiETBW- AAn dS0..dSn    | AgiETBW # AAn dS0..dSn   | AgiETBWI! dS0..dSn       |
 AgiETBW >!# dS0..dSn     | AgiETBW @# AAn dS0..dSn  | AgiETBW @ AAn dS0..dSn   |
 AgiETBW>!? AAn dS0..dSn  | AgiETBW ># dS0..dSn      | AgiETBW >I# dS0..dSn     |
 AgiETBWI^# dS0..dSn      | AgiETBWI# dS0..dSn       | AgiETBW ># AAn dS0..dSn
Instruction   ::=
 IEEEBW AAn dS0..dSn      | IEEEBW?  AAn dS0..dSn   | IEEEBW# AAn dS0..dSn   |
 IEEEBWI AAn dS0..dSn     | IEEEBW  dS0..dSn        | IEEEBW> AAn dS0..dSn   |
 IEEEBW- AAn dS0..dSn     | IEEEBW# AAn dS0..dSn    | IEEEBWI! dS0..dSn      |
 IEEEBW>!# dS0..dSn       | IEEEBW@# AAn dS0..dSn   | IEEEBW@ AAn dS0..dSn   |
 IEEEBW>!? AAn dS0..dSn   | IEEEBW># dS0..dSn       | IEEEBW>I# dS0..dSn     |
 IEEEBWI^# dS0..dSn       | IEEEBWI# dS0..dSn       | IEEEBW># AAn dS0..dSn
Instruction  ::=
 MsgREN- AAn   | MsgREN+ AAn | MsgREN AAn  | MsgREN-   | MsgREN+ | MsgREN   |
 MsgTRG  AAn   | MsgTRG      | MsgSDC AAn  | MsgSDC    | MsgDC   | MsgIFC
Instruction  ::=
 AryAdd  dStr  | AryAdd?  dStr   | AryAdd?     | ArySet dStr    |
 ArySet? dStr  | ArySet@  dStr   | AryAdd_     | AryAdd$        |
 ArySet$ dStr  | ArySet?         | ArySet_     | ArySet$        |
 AryAdd$ dStr  | AryAdd$
Instruction  ::=
 SetPMT EOC dStr    | SetPMT? EOC dStr    | SetPMT? EOC   | SetPMT EOC   |
 SetPMT dStr        | SetPMT? dStr        | SetPMT?
Instruction  ::=
 SetRDT EOC dStr    | SetRDT? EOC dStr    | SetRDT? EOC  | SetRDT EOC   |
 SetRDT dStr        | SetRDT? dStr        | SetRDT?
Instruction  ::=
 PIO+ AAn   | PIO? AAn   | PIO?+ AAn dStr  | PIO# AAn  | PIO AAn dStr
```

5.4 Description of Instructions

5.4.1 Instruction--FindListen

(a) An or AAn found from polling **GPIB-BUS** through GPIB-CONTROLLER and **GPIB-cable** will be registered to the IDAT, If instruction added with **Key-Symbol** '?' is used, addresses of An or AAn will be sent to PC.

(b) The connection of GPIB-CONTROLLER with PC and all instruments must be set up already before starting to use GPIB-CONTROLLER.

(c) The design of application program.

*Firmware Versions 2.1 (or previous) :
It is necessary to save numbers of AAn in IDAT of GPIB-CONTROLLER by Sending `**findlisten**` or `**RESET**`instruction at first to **GPIB-CONTROLLER.**

*Firmware Versions 2.2 (or further) :
While the USB connector of the GPIB-CONTROLLER and PC are linking (PlUG-OUT and then PLUG-IN), GPIB-CONTROLLER will execute the instruction of `Findlisten`to save numbers of AAn in the IDAT and open the connection of GPIB-CONTROLLER and AAn automatically.

Syntax:

Key-Word	Key-Symbol	Parameter
FindListen	?,#	AAn

```
Command                  ::= 'FINDLISTEN' Key-Symbol-FINDLISTEN AAn
Key-Symbol-FINDLISTEN ::= '?' | '#' | Lamda
```

AAn   ::= An AAn | Lamda

Exam: FindListen

Description:

1. PC will transmite instruction 'Findlisten' to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will excute Findlisten-Procedure of IEEE488 regard to GPIB-Address (1 to 30)
3. The results obtain from item-2 will be recorded to IDAT.

Exam: FindListen? 7 4 6 9

Rp: 07,04,09

Description:

1. PC will transmite instruction ' FindListen? 7 4 6 9' to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will excute Findlisten-Procedure of IEEE488 regard to GPIB-Address (7, 4, 6 and 9)
3. The results obtain from item-2 will be recorded to IDAT and transmitted back to PC.

Exam: Findlisten?

Rp : 0001,0101,0201,0301

Description:

1. PC will transmite instruction 'FindListen? 'to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will excute Findlisten-Procedure of IEEE488 regard to GPIB-Address (1 to 30)
3. The results “0001,0101,0201,0301” obtain from item-2 will be recorded to IDAT and transmitted back to PC.

   * The meaning of 0001 is Primary-Address = 1 and Secondary-Addres = 0
   * The meaning of 0101 is Primary-Address = 1 and Secondary-Addres = 1
   * The meaning of 0201 is Primary-Address = 1 and Secondary-Addres = 2
   * The meaning of 0301 is Primary-Address = 1 and Secondary-Addres = 3

Exam: Findlisten#

Rp : zz04zz07zz09

Description:

1. PC will transmite instruction 'FindListen? 'to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will excute Findlisten-Procedure of IEEE488 regard to GPIB-Address (1 to 30)
3. The results obtain from item-2 will be recorded to IDAT and transmitted in hex format back to PC.

   * The meaning of zz04 is Primary-Address = 4 and no Secondary-Addres
   * The meaning of zz07 is Primary-Address = 7 and no Secondary-Addres
   * The meaning of zz09 is Primary-Address = 9 and no Secondary-Addres

Exam: Findlisten# 1 0201

Rp : 00010101020103010201

Description:

1. PC will transmite instruction ' Findlisten# 1 0201 'to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will excute Findlisten-Procedure of IEEE488 regard to GPIB-Address (1 and 0201)
3. The results obtain from item-2 will be recorded to IDAT and transmitted in hex format back to PC.

*The meaning of 0001 is Primary-Address = 1 and Secondary-Addres = 0
*The meaning of 0101 is Primary-Address = 1 and Secondary-Addres = 1
*The meaning of 0201 is Primary-Address = 1 and Secondary-Addres = 2
*The meaning of 0301 is Primary-Address = 1 and Secondary-Addres = 3
*The meaning of 0201 is Primary-Address = 1 and Secondary-Addres = 2

5.4.2 Instruction--Idn

GPIB-CONTROLLER identification data read, and transmitted to **PC**.

Syntax:

Key-Word	Key-Symbol
IDN	?

```
Command              ::= 'IDN' Key-Symbol-IDN  dStr-IDN
Key-Symbol-IDN       ::= '?'
dStr-IDN             ::= dStr | Lamda
```

Exam:idn?
Rp: KI Work-Office, KI-GC1201, 10C4-EA60s00000100 (E-mail: kitec@phkaku.com) (Website: http://www.phkaku.com/), Version = 2.2
Description:
The KI-GC1201 is MODEL-NO for GPIB-CONTROLLER.
The 10C4 is Vender-ID and EA60 is Product-ID for the interface USB of GPIB-CONTROLLER.

Exam:idn? 'this is a message for good reading'
Rp: this is a message for good reading
Description:
1. PC transmite instruction
   "idn? 'this is a message for good reading'"
   to GPIB-CONTROLLER.
2. GPIB-CONTROLLER 傳送 'this is a message for good reading' back to PC.

5.4.3 Instruction--Status-Byte

(a) The process ofIEEE488 Serial-Poll executed to **AAn** ,signal SRQ regarded (with `+` or `-`) or disregarded (without `+` and `-`).
(b) The result of previous polling above **item 5.4.3 (a)** and transmitted to **PC**
(c) Function as following item **(1)** and item **(2)** is same as to **Std488.2-1987** page 164 common controller Protocols:
Instruction SPOLL with or without Key-Symbol `+` or `-`
(1) FINDRQS (equal to SPOLL with `+` or `-` regard signal SRQ)
(2) ALLSPOLL
(equal to SPOLL without `+` and `-` disregarded signal SRQ)..

Separate description of instruction:

SPOLL
The process of IEEE488-Serial-Polling executed to instrument AAn
PP
The process of IEEE488-Parallel-Polling executed to instrument AAn

Syntax:

Key-Word	Key-Symbol	Parameter				
SPOLL	?, +, -, #	**AAn**	dS0..dSn	DlyR	DlyW	DlyB
PP	?,\,-,#	**AAn**	dS0..dSn	DlyR	DlyW	DlyB

```
Command              ::= 'PP' Key-Symbol-PP AAn DSOToDSn
Key-Symbol-PP        ::= '?' | '\' | '-' | '#' | Lamda
```

```
Command             ::= 'SPOLL' Key-Symbol-SPOLL AAn DSOToDSn
Key-Symbol-SPOLL    ::= '?' | '+'  | '-' | '#' | '&' | '_' | '<'| Lamda
AAn                 ::= An AAn | Lamda
DSOToDSn            ::= DSOToDSn_R DlyRWB | ',' DlyRWB | Lamda
DSOToDSn_R          ::= DStr DSOToDSn_R | Lamda
DlyRWB              ::= DlyR DlyR_R
DlyR_R              ::= DlyW DlyW_R | Lamda
DlyW_R              ::= DlyB | Lamda
```

Description:

INSTRUCTION	ACTIONS
SPOLL?	The process of IEEE488-Serial-Poll executed to **AAn** from **PC** through the GPIB-CONTROLLER, signal SRQ is disregarded, result read from **AAn** will be sent to **PC**.
SPOLL-?	The process of IEEE488-Serial-Poll executed to **AAn** through the GPIB-CONTROLLER, signal SRQ is regarded. result read from **AAn** will be sent to **PC**.
SPOLL? 2 7	The process of IEEE488-Serial-Poll executed to **A2** and **A7** from **PC** through the GPIB-CONTROLLER, signal SRQ is disregarded, result read from **A2** and **A7** will be sent to **PC**.
SPOLL+	The process of IEEE488-Serial-Poll executed to **AAn** from **PC** through the GPIB-CONTROLLER, signal SRQ is regarded, result read from **AAn** will be sent to **PC**.
SPOLL+ 2 7	The process of IEEE488-Serial-Poll executed to **A2** and **A7** from **PC** through the GPIB-CONTROLLER, signal SRQ is regarded, result read from **A2** and **A7** will be sent to **PC**.
SPOLL# 2 7	The process of IEEE488-Serial-Poll executed to **A2** and **A7** from **PC** through GPIB-CONTROLLER, signal SRQ is disregarded, result read from **A2** and **A7** will be sent to **PC** with **Hex** format.
SPOLL#	The process of IEEE488-Serial-Poll executed to **AAn** from **PC** through the GPIB-CONTROLLER, signal SRQ is disregarded, result read from **AAn** will be sent to **PC** with **Hex** format
PP\	Signal of IEEE488 **PPU** message sent to **AAn**, from **PC** through the GPIB-CONTROLLER, configuration of PP within **AAn** will be deleted
PP- 6 5	Signal of IEEE488 **PPD** message sent to **A6** and **A5** from **PC** through the GPIB-CONTROLLER, configuration of **PP** for **A6** and **A5** will be deleted
PP 6 3 5 2 '13141516';	The signal of IEEE488 **PPC** and **PPE** message is sent to **A6, A3, A5** and **A2** from **PC** through the GPIB-CONTROLLER, and the **A6** will be configured to (13):PPR3 sense 1. the **A3** will be configured to (14):PPR4 sense 1 the **A5** will be configured to (15):PPR5 sense 1. the **A2** will be configured to (16):PPR6 sense 1.
PP?	The process of IEEE488-Parallel-Poll executed to instrument **AAn** from **PC** through the GPIB-CONTROLLER and the result from instrument will be sent to **PC**.

Exam: Send 7 '*CLS;*ESE 1;*SRE 96';Send 7 '*OPC';SPOLL+
Rp: 7:96
Description:
1. PC give instruction '*CLS;*ESE 1;*SRE 96'and '*OPC'to A7 through GPIB-CONTROLLER.
2. The function of IEEE488 Serial Polled action (regarded SRQ signal) to AAn will be executed by PC through GPIB-CONTROLLER.
3. the data obtained from instruments (AAn) for execution of

item 2 will be transmitted back to PC through GPIB-CONTROLLER.

Exam: Send 7 '*CLS;*ESE 1;*SRE 96';Send 7 '*OPC';SPOLL+#

Rp: 0760

Description:

1. PC give instruction '*CLS;*ESE 1;*SRE 96'and '*OPC'to A7 through GPIB-CONTROLLER.
2. The IEEE488 Serial Polled action (regarded SRQ signal) to AAn will be executed by PC through GPIB-CONTROLLER.
3. The data obtained from instruments (AAn) for execution of item 2 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:SPOLL 7 9 6 4 2

Rp: 32,161,00,30,00

Description:

1. The function of the IEEE488 Serial Polled toA7,A9,A6,A4 and A2 (disregarded SRQ signal) will be executed by PC through GPIB-CONTROLLER.
2. the data obtained from instruments A7,A9,A6,A4 and A2 for taking ation of item 1 will be transmitted back to PC through GPIB-CONTROLLER.

Exam:SPOLL# 7 9 6 4

Rp: 20A10048

Description:

1. The function of IEEE488 Serial Polled to A7,A9,A6,A4 and A2 (disregarded SRQ signal) will be executed by PC through GPIB-CONTROLLER.
2. The data obtained from instruments A7,A9,A6,A4 and A2 for execution of item 1 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:SPOLL+?

Rp: 07:32,09:161

Description:

1. The function of IEEE488 Serial Polled to AAn will be executed (regarded SRQ signal) by PC through GPIB-CONTROLLER.
2. The data obtained from instruments AAn for execution of item 1 will be transmitted back to PC through GPIB-CONTROLLER.

Exam:SPOLL#

Rp: 0400072009A1

Description:

1. The function of IEEE488 Serial Polled to AAn will be executed by PC (disregarded SRQ signal) through GPIB-CONTROLLER.
2. The data obtained from instruments AAn for execution of item 1 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP 28 '*ESE 1''*PRE 32''03';PP? '*CLS'

Rp: 04

Description:

1. PC give instruction '*ESE 1''*PRE 32'to A28 through GPIB-CONTROLLER.
2. Instrument A28 will be configured to the mode of PPR3-sense-0 (03)through GPIB-CONTROLLER by PC.
3. PC give instruction '*CLS'to AAn through GPIB-CONTROLLER
4. The function of IEEE488 Parellel Polled will be executed by PC through GPIB-CONTROLLER.
5. the data obtained from instruments (AAn) for taking action

    of item 4 will be transmitted back to PC through GPIB-CONTROLLER.
    (because ist value is '0' and PPR3 is sense-0, the data
     obtained from item 4 will be '04')

Exam:PP 28 '*ESE 1''*PRE 32''13';PP? '*CLS'

Rp: 00

Description:

1. PC give instruction '*ESE 1''*PRE 32'to A28 through GPIB-CONTROLLER.
2. Instrument A28 will be configured to the mode of PPR3-sense-1 (13) through GPIB-CONTROLLER by PC.
3. PC give instruction '*CLS'to AAn through GPIB-CONTROLLER.
4. The function of IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER.
5. The data obtained from instruments A28 for execution of item 2 will be transmitted back to PC through GPIB-CONTROLLER. (because ist value is '0' and PPR3 is sense-1, the data obtained from item 4 will be '00')

Exam:PP 28 '*CLS''*ESE 1''*PRE 32''15';PP# 28 '*OPC'

Rp: 10

Description:

1. PC give instruction '*ESE 1''*PRE 32'to A28 through GPIB-CONTROLLER.
2. Instrument A28 will be configured to the mode of PPR5-sense-1 (15) through GPIB-CONTROLLER by PC.
3. PC give instruction '*OPC'to AAn through GPIB-CONTROLLER.
4. The function of IEEE488 Parellel Polled will be executed by PC through GPIB-CONTROLLER.
5. The data obtained from instruments A28 for taking action of item 4 will be transmitted back to PC through GPIB-CONTROLLER. (because ist value is '1' and PPR5 is sense-1, the data obtained from item 2 will be '10')

Exam:PP 28 '*CLS''*ESE 1''*PRE 32''15';PP#

Rp: 00

Description:

1. PC give instruction '*ESE 1''*PRE 32'to A28 through GPIB-CONTROLLER.
2. Instrument A28 will be configured to the mode of PPR5-sense-1 (15)through GPIB-CONTROLLER by PC.
3. The function of IEEE488 Parellel Polled will be executed by PC through GPIB-CONTROLLER
4. The data obtained from instruments (AAn) for taking action of item 3 will be transmitted back to PC with Hex format through GPIB-CONTROLLER. (because ist value is '0' and PPR5 is sense-1, the data obtained from item 2 will be '00')

Exam:PP 28 '*CLS''*ESE 1''*PRE 32''05';PP#

Rp: 10

Description:

1. PC give instruction '*CLS''*ESE 1''*PRE 32'to A28 through GPIB-CONTROLLER.
2. Instrument A28 will be configured to the mode of PPR5-sense-0 (05)through GPIB-CONTROLLER by PC.
3. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER.
4. The data obtained from instruments (AAn) for taking action

of item 3 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.
(because ist value is '0' and PPR5 is sense-0, the data obtained from item 2 will be '10')

Exam:PP# 5 'SRQ 2''11'50
Rp: 01
Description:

1. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
2. Instrument A5 will be configured to the mode of PPR1 and sense-1 (11) through GPIB-CONTROLLER by PC.
3. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
4. The data obtained from instruments (AAn) for taking action of item 3 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP# 5 'SRQ 2''12'50
Rp: 02
Description:

1. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
2. Instrument A5 will be configured to the mode of PPR1 and sense-1 (12) through GPIB-CONTROLLER by PC.
3. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
4. The data obtained from instruments (AAn) for taking action of item 3 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP# 5 'SRQ 2''15'50
Rp: 10
Description:

1. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
2. Instrument A5 will be configured to the mode of PPR1 and sense-1 (15) through GPIB-CONTROLLER by PC.
3. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
4. The data obtained from instruments (AAn) for taking action of item 3 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP# 5 'SRQ 2''18'50
Rp: 80
Description:

1. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
2. Instrument A5 will be configured to the mode of PPR1 and sense-1 (18) through GPIB-CONTROLLER by PC.
3. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
4. The data obtained from instruments (AAn) for taking action of item 3 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP 28 '*OPC''12';PP 5 'SRQ 2''18';PP#
Rp: 82

Description:
1. PC give instruction '*OPC'to A28 through GPIB-CONTROLLER.
2. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
3. Instrument A28 will be configured to the mode of PPR2 and sense-1 (12) through GPIB-CONTROLLER by PC.
4. Instrument A5 will be configured to the mode of PPR8 and sense-1 (18) through GPIB-CONTROLLER by PC.
5. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
6. The data obtained from instruments (AAn) for taking action of item 5 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP 28 '*OPC''12';PP 5 'SRQ 2''17';PP#
Rp: 42
Description:
1. PC give instruction '*OPC'to A28 through GPIB-CONTROLLER.
2. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
3. Instrument A28 will be configured to the mode of PPR2 and sense-1 (12) through GPIB-CONTROLLER by PC.
4. Instrument A5 will be configured to the mode of PPR7 and sense-1 (17) through GPIB-CONTROLLER by PC.
5. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
6. The data obtained from instruments (AAn) for taking action of item 5 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP 28 '*OPC''11';PP 5 'SRQ 2''18';PP#
Rp: 81
Description:
1. PC give instruction '*OPC'to A28 through GPIB-CONTROLLER.
2. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
3. Instrument A28 will be configured to the mode of PPR1 and sense-1 (11) through GPIB-CONTROLLER by PC.
4. Instrument A5 will be configured to the mode of PPR8 and sense-1 (18) through GPIB-CONTROLLER by PC.
5. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
6. The data obtained from instruments (AAn) for taking action of item 5 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:PP 28 '*OPC''11';PP 5 'SRQ 2''13';PP#
Rp: 05
Description:
1. PC give instruction '*OPC'to A28 through GPIB-CONTROLLER.
2. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
3. Instrument A28 will be configured to the mode of PPR1 and sense-1 (11) through GPIB-CONTROLLER by PC.
4. Instrument A5 will be configured to the mode of PPR3 and sense-1 (13) through GPIB-CONTROLLER by PC.
5. The IEEE488 Parellel Polled action will be executed by PC

through GPIB-CONTROLLER
6. The data obtained from instruments (AAn) for taking action of item 5 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:SEND 28 '*OPC';SEND 5 'SRQ 2';PP# '1118'
Rp: 81
Description:
1. PC give instruction '*OPC'to A28 through GPIB-CONTROLLER.
2. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
3. Instrument A28 will be configured to the mode of PPR1 and sense-1 (11) through GPIB-CONTROLLER by PC.
4. Instrument A5 will be configured to the mode of PPR8 and sense-1 (18) through GPIB-CONTROLLER by PC.
5. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
6. The data obtained from instruments (AAn) for taking action of item 5 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

Exam:SEND 28 '*OPC';SEND 5 'SRQ 2';PP# '1112'
Rp: 03
Description:
1. PC give instruction '*OPC'to A28 through GPIB-CONTROLLER.
2. PC give instruction 'SRQ 2'to A5 through GPIB-CONTROLLER.
3. Instrument A28 will be configured to the mode of PPR1 and sense-1 (11) through GPIB-CONTROLLER by PC.
4. Instrument A5 will be configured to the mode of PPR2 and sense-1 (12) through GPIB-CONTROLLER by PC.
5. The IEEE488 Parellel Polled action will be executed by PC through GPIB-CONTROLLER
6. The data obtained from instruments (AAn) for taking action of item 5 will be transmitted back to PC with Hex format through GPIB-CONTROLLER.

It is necessary for **PP** to work correctly, and so instrument must have the function for supporting parallel-poll-response.

5.4.4 Instruction--Transmit-and-Receive

There are 3 functions in this form of instructions listed as below.

(1) Function1: **dStr** sent to instrument from **PC** through GPIB-CONTROLLER.
for example: SEND 3 ‘vset 1 5.7’

(2) Function2: Data read by **PC** from output-buffer of instrument through GPIB-CONTROLLER will be transmitted back to **PC** or **iaB** of GPIB-CONTROLLER.
for example: SEND? 7

(3) Function3: there have both functions as above description of Function1 and Function2,Function1 executed first.
for example: SEND? 7 ‘meas?’

(A) Key-Symbol `|`
SEND added with KeySymbol’|’
Because maximum length of block data **dStr** restricted to 228 bytes, in case there are large-**Block-Data** more than 228 bytes to be sent, the function of **Key-Symbol `|`** can be used for sending these data to instrument with several times under the same instruction.
Exam :

```
SENDl 4 'DATA:DAC VOLATILE, 2047,1536,1024,512,0,-512,-1536,-2047,'
SENDl 4 '0,-512,-1536,-2047,-512,-1536,-2047,1047,1236,1124,402,0,'
SENDl 4 '-512,-1136,-1547,0,-712,-1236,-1447,-112,'
SEND  4 '-1236,-1047,947,136,1624,4627,-312,-447,-192,-1536,-1147'
```

Description:

The data of 40 byte or more will be transmitted to the instruction A4 through GPIB-CONTROLLER by PC
(instruction of HEWLETT-PACKARD,33220A Function Generator)

The procedure of work described as below:

1. Pc give the Command
   "SENDl 4 'DATA:DAC VOLATILE, 2047,1536,1024,512,0,-512,-1536,-2047,'"
   and '0d' and '0a' to GPIB-CONTROLLER
2. The instruction of 'DATA:DAC VOLATILE, 2047,1536,1024,512,0,-512,
   -1536,-2047,' will be transmitted to instrument A4 by GPIB-CONTROLLER.
3. PC give command
   "SENDl 4 '0,-512,-1536,-2047,-512,-1536,-2047,1047,1236,1124,402,0,'"
   and '0d' and '0a' to GPIB-CONTROLLER
4. Data '0,-512,-1536,-2047,-512,-1536,-2047,1047,1236,1124,402,0,'
   will be transmitted to instrument A4 by GPIB-CONTROLLER.
5. PC give command
   "SENDl 4 '-512,-1136,-1547,0,-712,-1236,-1447,-112,' "
   and '0d' and '0a'
   to GPIB-CONTROLLER
6. Data '-512,-1136,-1547,0,-712,-1236,-1447,-112,'
   will be transmitted to instrument A4 by GPIB-CONTROLLER.
7. PC give command
   "SEND 4 '-1236,-1047,947,136,1624,4627,-312,-447,-192,-1536,-1147'"
   and '0d' and '0a' to GPIB-CONTROLLER
8. Data '-1236,-1047,947,136,1624,4627,-312,-447,-192,-1536,-1147'
   will be added with a signal of EOI by GPIB-CONTROLLER
9. Data and EOI obtained from item-8 will be transmitted to instrument
   A4 by GPIB-CONTROLLER.

* The command "Send" mentioned above the data of item 1,3,5 is added With Key-Symbol 'l' in the rear, it will mean that these data are not wholly transmitted to the instrument through GPIB-CONTROLLER, there are some data left in PC without transmission yet, therefore the Handshak between GPIB-CONTROLLER and instrument A4 are not stopped after item2,4,6 are completed already.

* The command "Send" of the final item 7 without adding Key-Symbol 'l' will mean that the data in PC are already transmitted to GPIB-CONTROLLER completely, these data through GPIB-CONTROLLER will be automatically added with EOI signal and transmitted to instrument A4, the action of Handshak between GPIB-CONTROLLER and instrument A4 will be stopped by EOI signal.

Exam:

SEND added with KeySymbol'^'

GPIB-CONTROLLER send SIN-ROM-Encoded data of wave to Agilent 33220A
The SIN-ROM-Encoded data convert to Hex format and then transmitted to instrument A4 (Agilent 33220A) through GPIB-CONTROLLER by PC.

```
  0       ,490    ,980    ,1467   ,1950   ,2429   ,2902   ,3368
 ,3826    ,4275   ,4713   ,5141   ,5555   ,5956   ,6343   ,6715
 ,7071    ,7409   ,7730   ,8032   ,8314   ,8577   ,8819   ,9039
 ,9238    ,9415   ,9569   ,9700   ,9807   ,9891   ,9951   ,9987
```

```
      ,9999   ,9987   ,9951   ,9891   ,9807   ,9700   ,9569   ,9415
      ,9238   ,9039   ,8819   ,8577   ,8314   ,8032   ,7730   ,7409
      ,7071   ,6715   ,6343   ,5956   ,5555   ,5141   ,4713   ,4275
      ,3826   ,3368   ,2902   ,2429   ,1950   ,1467   ,980    ,490
      ,0      ,-490   ,-980   ,-1467  ,-1950  ,-2429  ,-2902  ,-3368
      ,-3826  ,-4275  ,-4713  ,-5141  ,-5555  ,-5956  ,-6343  ,-6715
      ,-7071  ,-7409  ,-7730  ,-8032  ,-8314  ,-8577  ,-8819  ,-9039
      ,-9238  ,-9415  ,-9569  ,-9700  ,-9807  ,-9891  ,-9951  ,-9987
      ,-9999  ,-9987  ,-9951  ,-9891  ,-9807  ,-9700  ,-9569  ,-9415
      ,-9238  ,-9039  ,-8819  ,-8577  ,-8314  ,-8032  ,-7730  ,-7409
      ,-7071  ,-6715  ,-6343  ,-5956  ,-5555  ,-5141  ,-4713  ,-4275
      ,-3826  ,-3368  ,-2902  ,-2429  ,-1950  ,-1467  ,-980   ,-490
```

Code represented with format of Visual C++ as below:
(hComm is handle to the communication port)

```
WriteFile
(hComm,
 "SEND| 4\'DATA:DAC VOLATILE, #3256\';SEND|^4\'000001ea03d405bb079e097d0b56"
 "0d280ef210b31269141515b3174418c71a3b1b9f1cf11e321f60207a21812273234f24162"
 "4c7256125e4264f26a326df2703270f270326df26a3264f25e4256124c72416234f22\'"
 "\r\n"
 ,217,&nByteWrite,NULL
);
WriteFile
(hComm,
 "SEND|^4\'732181207a1f601e321cf11b9f1a3b18c7174415b31415126910b30ef20d280b"
 "56097d079e05bb03d401ea0000fe16fc2cfa45f862f683f4aaf2d8f10eef4ded97ebebea4"
 "de8bce739e5c5e461e30fe1cee0a0df86de7fdd8ddcb1dbeadb39da9fda1cd9b1d95d\'"
 "\r\n"
 ,217,&nByteWrite,NULL
);
WriteFile
(hComm,
 "SEND ^4\'d921d8fdd8f1d8fdd921d95dd9b1da1cda9fdb39dbeadcb1dd8dde7fdf86e0a0"
 "e1cee30fe461e5c5e739e8bcea4debebed97ef4df10ef2d8f4aaf683f862fa45fc2cfe16\'"
 "\r\n"
 ,147,&nByteWrite,NULL
);
```

Exam :

Transmit the data as following to instrument A4 through GPIB-CONTROLLER
DATA:DAC VOLATILE, 999,1000,1001,1002,1003,1004,1005,1006,1007,1008,
1009,1010,1011,1012,1013,1014,1015

Code represented with format of Visual C++ as below:

```
WriteFile
(hComm,
 "SEND| 4 \'DATA:DAC VOLATILE,\';SEND  4 \' 999, 1000, 1001, 1002, 1003, 10"
 "04, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015\'"
 "\r\n"
 ,143,&nByteWrite,NULL
);
```

Exam:(Instruction of instrument 33120A Function-Generator)

```
 SEND\ 1 'DATA:DAC VOLATILE, #216''07FF0600040002000000FE00FA00F801'
Description:
 1. PC give the command
     "Send\ 1 'DATA:DAC VOLATILE, #216''07...00F801'" and '0d' and '0a'
    to GPIB-CONTROLLER.
    Code represented with format of Visual C++ as below:
    nByteWrite=wsprintf
    (codebuf,
     "SEND\\ 1 \'DATA:DAC VOLATILE, #216\'\'07FF0600040002000000FE00FA00F801"
     "\'\r\n"
    );
    WriteFile
    (hComm,
     codebuf,nByteWrite,&nByteWrite,NULL
    );
 2. The data of '07FF06...01' will be interpreted as the format
    binary by GPIB-CONTROLLER
 3. The data of 'DATA:DAC VOLATILE, #216' and binary data obtained from
    item-2 will be transmitted to instrument A18 by GPIB-CONTROLLER.
```

(B) Key-Symbol `^`
SEND added with KeySymbol'^'
With Key-Symbol `^`,if the data is binary format which can be written into Hex format and saved in **dStr**. The **Key-Symbol `^`**added to the end of **Key-Word** will make GPIB-CONTROLLER translate the data of **dStr** into binary format before sending data to instrument, and then the binary format data will be sent to instrument.

Exam:SEND^? 7 '4D4541533A41433F'
Rp: 1.17373430E-06
Description:
1 The data 'MEAS:AC?' which is going to be transmitted will be written in Hex format '4D4541533A41433F'.
2 The data changed into Hex-Format will become the contents as for parameter dStr of instruction.
3 The instruction Key-Word 'SEND' is added with KeySymbol '^' and '?' in the rear,
  Such as:
    SEND^? 7 '4D4541533A41433F'
4 PC give the command
  "SEND^? 7 '4D4541533A41433F'" and '0d' and '0a'
  to GPIB-CONTROLLER.
5 Hex format of data '4D4541533A41433F' will be interpreted to binary format of 'MEAS:AC?' by GPIB-CONTROLLER automatically.
6 Data 'MEAS:AC?' obtained from Step-5 will be transmitted to instrument A7 by GPIB-CONTROLLER.
7 The output-buffer data of instrument A7 will be read by GPIB-CONTROLLER.
8 GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from Step-7 .
9 the data left from Step-8 after deleting will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

(C) Difference between instruction-with-% and Instruction-without-%

SEND added with KeySymbol'%'

(1) the instruction without adding '%':

PC give the instruction to GPIB-CONTROLLER that parameter of secondary address for the instruction is zero. GPIB-CONTROLLER will disregard the IEEE488-Message secondary address of SA0, it means that GPIB-CONTROLLER will not send the signal SA0 to the instrument.

Example :

SEND? 7 'MEAS:DC?'

a. PC give the instruction "SEND? 7 'MEAS:DC?'" to GPIB-CONTROLLER.

b. GPIB-CONTROLLER transmite IEEE488-Message to instrument as following:

ATN1 UNL TA0 LA7 ATN0 'MEAS:DC?' ATN1 LA0 TA7....

No signal SA0 are included.

(2) the instruction with adding '%':

PC give the instruction to GPIB-CONTROLLER that parameter of secondary address for the instruction is zero. GPIB-CONTROLLER will transmite the IEEE488-Message secondary address of SA0 to instrument,

Example :

SEND? 7 'MEAS:DC?'

a. PC give the instruction "SEND? 7 'MEAS:DC?'" to GPIB-CONTROLLER.

b. GPIB-CONTROLLER transmite IEEE488-Message to instrument as following:

ATN1 UNL TA0 LA7 SA0 ATN0 'MEAS:DC?' ATN1 LA0 TA7 SA0....

two signal of SA0 is included.

(D) character \`?\`If only a character \`?\` is included in a dStr of instruction, this **dStr with \`?\`**would not be sent to instrument, but data which is already saved in output-buffer of instrument will be read through GPIB-CONTROLLER and sent to **PC**.

Exam:SEND 7'*idn?';SEND? 7 '?''MEAS:DC?'

Rp: Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09;-1.17373430E-06

Exam:SEND 7 'MEAS:AC?''?'':CONF:VOLT:DC;:SAMP:COUN 1'':READ?''?'

Rp: +5.43147446E-06;+2.53149556E-08

Exam:SEND 9'*idn?';TRANSCEIV? 7 7 9 7 ':SAMP:COUN 2'':READ?''?''MEAS:DC?'

Rp: +2.07209052E-05,+2.32005757E-05;AGILENT TECHNOLOGIES,DSO-X2012A, MY52132806,02.10.2012022200;-1.17373430E-06

(E) Key-Symbol '$'

SEND added with KeySymbol'$'

Exam: Send$ '*idn?;'

```
Rp: 04:HEWLETT-PACKARD,6611C,0,A.01.03 "
    07:Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
    09:AGILENT TECHNOLOGIES,DSO-X 2012A,MY52132806,02.10.2012022200
    17:Agilent Technologies,E5071B,MY42404423,A.06.50
```

Description:

PC give the instruction AAn through GPIB-CONTROLLER, the Key-Symbol '$' added on the rear of Key-Word will the have function as below:

1. PC read the data of the output-buffer of instrument AAn through GPIB-CONTROLLER.
2. the number of each Gpib-address for each instrument An will be added in the front of the data which are read from output-buffer of each instrument.
3. 0d+0a will be added at the rear of the data which are read from output buffer of each instrument An.
4. All of the address value, the data, and the 0d+0a obtained

wholly from item 2 and 3 (above mentioned) will be sent
back to PC through

(F) Key-Symbol '*'

SEND added with KeySymbol'*'

Exam: Send* 22 'TDF M''TRA?'

Rp: 5625,6636,6161,3446,1226,1264,1136,1294,1134,1087,1309,1239,1182…

Description:

1. PC give the Command
   "Send* 22 'TDF M''TRA?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. Instruction 'TDF M' will be transmitted to A22 by GPIB-CONTROLLER.
3. Instruction 'TRA?' will be transmitted to A22 by GPIB-CONTROLLER.
4. The data in output-buffer of instrument A22 will be read by GPIB-CONTROLLER.
5. The data obtained from 'item 4' will be transmitted back to PC by GPIB-CONTROLLER.

(G) Key-Symbol '='

SEND added with KeySymbol'='

```
Exam:Send= 18 'IP;''KEYCLR;''ID?;''CF 300MHZ;SP 2MHZ;RB 100KHZ;''CF?'
              'rb?''RL?''RLPOS?''SER?''REV?''SETDATE?''SETTIME?'
              'SQLCH?''SS?''SRCTK?''SRCALC?''SRCPSTP?''SRCPWR?'
              'SRCPSWP?''SRCNORM?''SRCPOFS?''MEM?''AMB?''AMBPL?'
              'AMPLEN?''ANLGPLUS?';
```

```
Rp:HP8591EM;300000000;100000;0;8;976;960709;120621;141459;0;
   100000000;0;INT;10.00;-10.00;0;OFF;0;261414;OFF;OFF;-1;OFF
```

Description:

1. PC give the command
   "Send= 18 'IP;''KE....ANLGPLUS?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. Instruction 'IP;','KEYCLR;' and 'ID?;' will be transmitted to A18 by GPIB-CONTROLLER.
3. The data in output-buffer of instrument A18 will be read by GPIB-CONTROLLER.
4. The data are obtained from 'item 3', if data are inclusive of 0d+0a, 0d+0a will be replaced by ';' and then transmitted back to PC.
5. Instruction 'CF 300MHZ;SP 2MHZ;RB 100KHZ;' and 'CF?' will be transmitted to A18 by GPIB-CONTROLLER.
6. GPIB-CONTROLLER read the data in output-buffer of instrument A18
7. The data are obtained from 'item 6', if data are inclusive of 0d+0a, 0d+0a will be replaced by ';' and then transmitted back to PC.
8. Instruction 'rb?' will be transmitted to A18 by GPIB-CONTROLLER.
9. The data in output-buffer of instrument A18 will be read by GPIB-CONTROLLER.
10. The data are obtained from 'item 9', if data are inclusive of 0d+0a, 0d+0a will be replaced by ';' and then transmitted back to PC.
    ...
    ...

ii. Instruction 'ANLGPLUS?' will be transmitted to A18 by GPIB-CONTROLLER.

jj. The data in output-buffer of instrument A18 will be read by

GPIB-CONTROLLER.

kk. If data are included with 0d or 0a (EOS) and obtained from item jj mentioned above. the 0d or 0a will be only deleted by GPIB-CONTROLLER from data.

ll. The data left after deleting are obtained from item kk, these data will be added again with 0d+0a (EOS) at the rear by GPIB-CONTROLLER and sent back to PC .

(H) Key-Symbol '\'

SEND added with KeySymbol'\'

if Ascii-format-data and Binary-data of Hex format are included in the instruction, under the situation, Key-Symbol '|' is available, examples drscribed as below:

Exam:(Instruction of instrument HP8591EM/EMC ANALYZER)

```
Send\ 18 'IB''2000000031323334353637383961626364656667686966A6B
              6C6D6E6F707172737475767774F747576774F747576774F74
              7576774F747576774F747576774F';
```

Description:

1. PC give the command
   "Send\ 18 'IB''20...774F'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. Parameter data of '20...774F' in the command will be interpreted to binary format by GPIB-CONTROLLER
3. The instruction of 'IB' and binary data obtained from item-2 will be transmitted to instrument A18 by GPIB-CONTROLLER.

(I) Key-Symbol '#'

SEND added with KeySymbol'#'

Exam: Send# 7 'MEAS:VOLT:AC?'

Rp: 2B312E38363839383333534452D30330A

Description:

1. PC give the command
   "Send# 7 'MEAS:VOLT:AC?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. Instruction 'MEAS:VOLT:AC?' will be transmitted to instrument A7 by GPIB-CONTROLLER;
3. GPIB-CONTROLLER will read the data in output-buffer of instrument A7 and stop reading until EOI signal is received by GPIB-CONTROLLER
4. The data obtained from item 3 above mentioned will be interpreted to Hex format, and added with 0d+0a at the rear, and then transmitted back to PC by GPIB-CONTROLLER.

(J) Key-Symbol '?'

SEND added with KeySymbol'?'

Exam: Send? 'MEAS:VOLT:AC?'

Rp: +1.86898354E-03

description:

1. PC transmite command
   "Send? 7 'MEAS:VOLT:AC?'"和 '0d' 和 '0a'
   to GPIB-CONTROLLER
2. GPIB-CONTROLLER transmiteinstruction 'MEAS:VOLT:AC?'
   to instrument A7
3. GPIB-CONTROLLER read data from the output-buffer of A7 until the signal of EOI is received.
4. GPIB-CONTROLLER transmite data obtained from item 3 and ‘0d’ and ‘0a’ back to PC.

Separate Description of instructions:

SEND

(a) **dS0..dSn** sent to **An** or **AAn**

(b) The data of **IEEE488-ARBITRARY-ASCII** format will be read from **An** or **AAn** Through GPIB-CONTROLLER by PC, if **Key-Symbol** `*` is not used in instruction, **EOS** only will be deleted from data added with EOS and replaced by `;` or **0d+0a**

(c) SEND instruction with **Key-Symbol '&'** will have the same function as the instruction TRANSCEIV.

(d) If the instruction of SEND have **Key-Symbol '~'**, the function would be the same as TRANSCEIV of instruction, but after the last block-data of the instruction completely transmitted to instrument from **PC** through GPIB-CONTROLLER, no matter which include the character **`?`** whether in the last block data or not, GPIB-CONTROLLER will read automatically the data from output buffer of instrument and send them back to **PC**.

(e) SEND instruction with **Key-Symbol '+'**:
It will make the dStr of instruction transformed into the block data type of **IEEE488-DEFINITE-ARBITRARY-BLOCK** and then transmite those data to instrument through GPIB-CONTROLLER.
for example:
Instruction:
send?+ 4 'abcdefghijklmnopqrstuvwxyz1234567890'
Description:
The **IEEE488-DEFINITE-ARBITRARY-BLOCK** data dStr of '#236abcdefghijklmnopqrstuvwxyz1234567890' will be transmitted to the instrument of address-4, and the'#236' in dStr is the BLOCK-HEAD of dStr.

(f) SEND instruction with **Key-Symbol '-'**:
It will make the dStr of instruction transformed into the block data type of **Agilent-8753ET-Form 1,2,3,5-BLOCK** and then transmite those data to instrument through GPIB-CONTROLLER.
for example:
Instruction:
send?- 4 'abcdefghijklmnopqrstuvwxyz1234567890'
Description:
The **Agilent-8753ET-Form 1,2,3,5-BLOCK** data dStr of '#A$abcdefghijklmnopqrstuvwxyz1234567890' will be transmitted to the instrument of address-4, and the '#A$' in dStr is the BLOCK-HEAD of dStr.

AgiETB

(a) Parameter **dS0..dSn** in the instruction sent to **An** through GPIB-CONTROLLER by **PC**

(b) Data of **Agilent-8753ET-Form 1,2,3,5-BLOCK** format are read from An through the GPIB-CONTROLLER by PC. The number (bytes) of data read will be detected by the GPIB-CONTROLLER according to the block-head of data automatically.

Exam: AgiETB? 16 'FORM1;''OUTPDATA;'

```
Rp: #A.....B...$....."....`...........N.d.........X......1.....X...P…
```

Description:

1. PC give the command
   "AgiETB? 16 'FORM1;''OUTPDATA;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'FORM1;' will be transmitted to A16 by GPIB-CONTROLLER

3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in the output-buffer of instrument A16 and stop reading until EOI signal is received.
5. The data of 1210-Byte obtained from item 4 will be transmitted to PC by GPIB-CONTROLLER.

ARBITRARY

(a) Parameter **dS0..dSn** in the instruction sent to **An** through GPIB-CONTROLLER by **PC**

(b) The result of any format of data read from **An** will be sent to **PC**, number of reading data will be decided by Parameter **BLEN**.

(c) in various format of data, Parameter **BLEN** is able to decide on reading data from one or more times to complete.

Exam:ARBITRARY? 1024 16 'FORM1''OUTPDATA;'

Rp: #A...p.P...$.....$....\...........P.d.........N...........n. ….

Description:

1. PC give the command
   "ARBITRARY? 1024 16 'FORM1''OUTPDATA;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'FORM3' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the 1024-byte of data in output-buffer of instrument A16 .
5. GPIB-CONTROLLER will transmite the data obtained from item 4 back to PC.

IEEEB

(a) Parameter **dS0..dSn** in the instruction sent to **An** through GPIB-CONTROLLER by **PC**.

(b) The data of **IEEE488-DEFINITE-LENGTH-ARBITRARY-BLOCK** format are read from output buffer of instrument An through GPIB-CONTROLLER by **PC**, the number of data to stop reading automatically will be detected by GPIB-CONTROLLER in accordance with the block-head

Exam:IEEEB 9 'SYST:SET?;';IEEEB? 9

Rp:#800076354.setup prod="InfiniiVision" ver="02.10.2012022200"…

Description:

1. PC give the command
   "IEEEB 9 'SYST:SET?;';IEEEB? 9" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction 'SYST:SET?;' will be transmitted to A9 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in output-buffer of instrument A9 when total byte number of receiving data are coincedental with the regulation for block-head of data obtained from instrument, GPIB-CONTROLLER will stop reading data automatically.
4. The 76364 bytes of data obtained from item-3 will be transmitted back to PC by GPIB-CONTROLLER,
   (10 bytes of BLOCK-HEAD '#800076354' will be included in the 76364 bytes of data)

IEEEidn

(a) IEEE488.2 `*idn?` sent to **An** or **AAn**

(b) The data of IEEE488-ARBITRARY-ASCII format are read from **An** or **AAn** through GPIB-CONTROLLER by **PC**, only **EOS** will be deleted from the data added with **EOS** and replaced with character `;` or **0d+0a**, and then sent back to **PC**

TestSys

(a) IEEE488.2 `*tst?` sent to **An** or **AAn**, from which the format of **IEEE488-ARBITRARY-ASCII** will be read.

(b) Value of **DlyB** set-up to 7000 automatically. if the length for **GPIB-RW-POLLING-TIME** are needed to increase or decrease, value Of **DlyB** within instruction would be re-set.
for example:
TestSys? 100 100 9000; "DlyB would be re-set to 9000

EOSO

Same function as previous 'send' but EOS is input-parameter in addition.

TRANSCEIV

Any one of Parameter at **dS0…dSn** (such as **dStr0**) is sent to one of instruments among **AAn** (such as **A0**), if the end of **dStr0** is added with character `**?**`,GPIB-CONTROLLER would automatically read data in output buffer of instrument and send them to **PC**, reference as below:

Block data at dS0..dS	Instrument addresses at AAn
dStr0	A0
…	•••
dStrN	AN

In a word, addresses of the parameter for sending block data must be corresponded to the addresses of received-instrument.

Exam:TRANSCEIV? 9 7 '*IDN?' 'MEAS:AC?'
Rp: AGILENT TECHNOLOGIES,DSO-X 2012A,MY52132806,02.10.2012022200.;
+1.55823202E-03.
NOTE: `*IDN?` corresponded to A9 (Rp is '…DSO-X 2012A…')
'MEAS:AC?' corresponded to A7 (Rp is '+1.55823202E-03')

Syntax:

Key-Word	Key-Symbol	Param	Param	Param	Param	Param	Param
SEND	~,&,=,!,\,%,|,^,-,+,?,$,#,*,>		AAn	dS0..dSn	DlyR	DlyW	DlyB
SEND	~,&,=,!,\,%,|,^,-,+,?,$,#,*,>		AAn		DlyR	DlyW	DlyB
EOSO	~,&,=,!,\,%,|,^,-,+,?,$,#,*,>	EOS	AAn	dS0..dSn	DlyR	DlyW	DlyB
EOSO	~,&,=,!,\,%,|,^,-,+,?,$,#,*,>	EOS	AAn		DlyR	DlyW	DlyB
ARBITRARY	!,>,|,^,-,?,$,#	BLEN	An	dS0..dSn	DlyR	DlyW	DlyB
AgiETB	!,<,_,>,|,^,?,#		An	dS0..dSn	DlyR	DlyW	DlyB
IEEEB	!,<,_,>,|,^,?,#		An	dS0..dSn	DlyR	DlyW	DlyB
IEEEidn	>,?,$,#		AAn		DlyR	DlyW	DlyB
TestSys	>,?,$,#		AAn		DlyR	DlyW	DlyB
TRANSCEIV	!,^,-,+,?,#,*,>		AAn	dS0..dSn	DlyR	DlyW	DlyB

```
Command                ::= 'arbitrary' Key-Symbol-arbitrary BLEN An DS0ToDSn
 Description:
  BLEN is the number of bytes to be transmitted from instrument to PC.
Key-Symbol-arbitrary ::= '!' | ' |' | '^' | '-' | '?' | '#' | Lamda
Command                ::= 'AgiETB' Key-Symbol-AgiETB An DS0ToDSn
Key-Symbol-AgiETB      ::= '!' | ' |' | '^' | '-' | '?' | '#' | Lamda
Command                ::= 'IEEEB' Key-Symbol-IEEEB An DS0ToDSn
Key-Symbol-IEEEB       ::= '!' | ' |' | '^' | '-' | '?' | '#' | Lamda
```

```
Command              ::= 'IEEEidn' Key-Symbol-IEEEidn AAn
Key-Symbol-IEEEidn   ::= '!' | '$' | '?' | '#' | Lamda
Command              ::= 'TestSys' Key-Symbol-TestSys AAn
Key-Symbol-TestSys   ::= '!' | '$' | '?' | '#' | Lamda
Command              ::= 'TransCeiv' Key-Symbol-TestSys AAn DS0ToDSn
AAn                  ::= An AAn | Lamda
DS0ToDSn             ::= DS0ToDSn_R DlyRWB | ',' DlyRWB | Lamda
DS0ToDSn_R           ::= DStr DS0ToDSn_R | Lamda
DlyRWB               ::= DlyR DlyR_R
DlyR_R               ::= DlyW DlyW_R | Lamda
DlyW_R               ::= DlyB | Lamda
For example:
 SEND? '*cls''meas?'
  AAn      = (NULL)
  DS0ToDSn = '*cls''meas?'
  DlyRWB   = (NULL)
 SEND? 7 '*cls''meas?' 450
  AAn      = 7
  DS0ToDSn = '*cls''meas?'
  DlyRWB   = 450
 SEND 7 9 '*cls'
  AAn      = 7 9
  DS0ToDSn = '*cls'
  DlyRWB   = 450
 SEND? 7,200
  AAn      = 7
  DS0ToDSn = Lamda(NULL)
  DlyRWB   = 200
```

Examples for instructions description:

Examples of Model ADVANTEST, R3131:

Exam:SEND? 19 '*IDN?'

Rp: ADVANTEST,R3131,22286039,B02

Description:

1. PC give the command "SEND? 19 '*IDN?'" and '0d' and '0a' to GPIB-CONTROLLER.
2. Data '*IDN?' will be transmitted to A19 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A19 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with data obtained from item-3
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted to PC by GPIB-CONTROLLER.

Examples of Model HEWLETT PACKARD,8720D:

Exam:SEND? 16 '*IDN?'

Rp: HEWLETT PACKARD,8720D,0,6.06

Description:

1. PC give the command "SEND? 16 '*IDN?'" and '0d' and '0a' to GPIB-CONTROLLER.

2. Data '*IDN?' will be transmitted to instrument A16 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A16 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Examples of Model **TEKTRONIX,AFG3102**:

```
 Exam:SEND? 3 '*IDN?'
  Rp: TEKTRONIX,AFG3102,C011603,SCPI:99.0 FV:1.2.1
```

Examples of Model **Hewlett-Packard, 8648D**:

```
 Exam:SEND? 6 '*IDN?'
  Rp: Hewlett-Packard, 8648D, 3847M00189, B.04.09
```

Examples of Model **HEWLETT-PACKARD,E3631A**:

```
 Exam:SEND? 7 '*IDN?'
  Rp: HEWLETT-PACKARD,E3631A,0,2.1-5.0-1.0
```

Examples of Model **WAYNE KERR,4235**:

```
 Exam:SEND? 6 '*IDN?'
  Rp: WAYNE KERR,4235,0,1.85
```

Examples of Model **ROHDE&SCHWARZ,NRVD**:

```
 Exam:SEND? 24 '*IDN?'
  Rp: ROHDE&SCHWARZ,NRVD, 100035.002,V1.52  V1.40
```

Examples of Model **ADVANTEST,R3162**:

```
 Exam:SEND? 10 '*IDN?'
  Rp: ADVANTEST,R3162,120301463,F05
```

Examples of Model **HEWLETT-PACKARD,83620A**:

```
 Exam:SEND? 19 '*IDN?'
  Rp: HEWLETT-PACKARD,83620A,3420A02158,05 APR 94
```

Examples of Model **MARCONI-INSTRUMENTS-2031**:

```
 Exam:SEND? 19 '*IDN?'
  Rp: MARCONI INSTRUMENTS,2031,119851047,9.02
```

Examples of Model **SME03**:

```
 Exam:SEND? 28 '*IDN?'
  Rp: Rohde&Schwarz,SME03,835328/017,4.11
```

Examples of Model **SMT06**:

```
 Exam:SEND? 28 '*IDN?'
  Rp: Rohde&Schwarz,SMT06,830723/003,4.11
 Exam:SEND? 28 'SOUR:FREQ 100E6''SOUR:POW:LIM 16 dBm''SOUR:FREQ?'
  Rp: 99999990.0
 Exam:SEND? 28 'SOUR:FREQ 200E6''SOUR:'SOUR:POW:LIM 10 dBm''SOUR:FREQ?'
  Rp: 200000010.0
```

Examples of Model **E3632A**:

```
 Exam:FINDLISTEN?
  Rp: 05
```

Description:

1. PC give the command
   "FINDLISTEN? " and '0d' and '0a'
   to GPIB-CONTROLLER.
2. GPIB-CONTROLLER execute ieee488.1 Findlisten procedure to gpib-bus.
3. The data obtained from item (2) will be added with (0d+0a) in the rear by GPIB-CONTROLLER.

4. The data obtained from item (3) will be transmitted back to PC by GPIB-CONTROLLER.

Exam:SEND? 5 '*IDN?'

Rp: HEWLETT-PACKARD,E3632A,0,1.4-5.0-1.0

Description:

1. PC give the command
   "SEND? 5 '*IDN?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. Data '*IDN?' will be transmitted to A5 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A5 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete thee EOS (End of String) added with the data obtained from item-3
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam:SEND? 5 'output?'

Rp: 0

Description:

1. PC give the command
   "SEND? 5 'output?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. Data 'output?' will be transmitted to A5 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A5 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with characters of 0d+0a at the rear and then transmitted to PC by GPIB-CONTROLLER.

Examples of Model 34401a:

Exam:FINDLISTEN?

Rp: 07

Description:

1. PC give the command
   "FINDLISTEN? 7" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. GPIB-CONTROLLER execute ieee488.1 Findlisten procedure to A7 of gpib-bus.
3. The data obtained from item (2) will be added with (0d+0a) in the rear by GPIB-CONTROLLER.
4. The data obtained from item (3) will be transmitted back to PC by GPIB-CONTROLLER.

Exam: Send>? 7 'MEAS:DC?'

Rp: -3.21000047E-05

Description:

1. PC give the command
   "Send? 7 'MEAS:DC?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'MEAS:DC?' will be transmitted to A7 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7 and stop reading until EOI signal is

received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam:SEND? 7 '*IDN?'

Rp: HEWLETT-PACKARD,34401A,0,7-5-2

Description:

1. PC give the command
   "Send? 7 '*IDN?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of '*IDN?' will be transmitted to A7 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam: Send>? 7 'MEAS:DC?;'

Rp: -3.21000047E-05

Description:

1. PC give the command
   "Send>? 7 'MEAS:DC?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'MEAS:DC?' will be transmitted to A7 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Examples of Model 34410a:

Exam:SEND? 7 '*IDN?;'

Rp: Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09

Description:

1. PC give the command
   "SEND? 7 '*IDN?;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of '*IDN?;' will be transmitted to A7 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back toPC by GPIB-CONTROLLER.

Exam:ARBITRARY? 3 7 '*idn?;'

Rp: Agi

Description:

1. PC give the command
   data "ARBITRARY? 3 7 '*idn?;'" and '0d' and '0a'

      to GPIB-CONTROLLER.
   2. The instruction of '*idn?;' will be transmitted to A7 by
      GPIB-CONTROLLER
   3. GPIB-CONTROLLER will read 3 bytes of the data in output-buffer
      of instrument A7.
   4. The 3 bytes data obtained from item-3 will be transmitted
      back to PC by GPIB-CONTROLLER.

Exam:ARBITRARY? 59 7

Rp: lent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09.

Description:

1. PC give the command
   "ARBITRARY? 59 7" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will read 57 bytes of the data in
   output-buffer of instrument A7
   (not 59 bytes, because there are 57 bytes data only in
   output-buffer of instrument A7)
3. The 57 bytes data will be transmitted back to PC by
   GPIB-CONTROLLER

Exam: Send>? 7 'MEAS:DC?;'

Rp: -8.42030017E-05

Description:

1. PC give the command
   "Send>? 7 'MEAS:DC?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. GPIB-CONTROLLER will take a little of delay-time to
   transmite the instruction 'MEAS:DC?'to the instrument A7.
3. GPIB-CONTROLLER will read the data in output-buffer of
   instrument A7 and stop reading until EOI signal is
   received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added
   with the data obtained from item-3.
5. The data left from item-4 will be added with 0d+0a at the
   rear and then transmitted to PC by GPIB-CONTROLLER.

Exam: Send| 7 'MEAS'
      Send? 7 ':DC?'

Rp: -8.42030017E-05

Description:

1. PC give the command
   "Send| 7 'MEAS'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'MEAS' will be transmitted to A7 by
   GPIB-CONTROLLER
3. PC give the command
   "Send? 7 ':DC?;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
4. The instruction of ':DC?;' will be transmitted to A7 by
   GPIB-CONTROLLER
5. GPIB-CONTROLLER will read the data in output-buffer of
   instrument A7 and stop reading until EOI signal is
   received.
6. GPIB-CONTROLLER will delete the EOS (End of String) added
   with the data obtained from item-5.
7. The data left from item-6 will be added with 0d+0a at the
   rear and then transmitted to PC by GPIB-CONTROLLER.

Exam: Send^? 7'4D4541533A44433F3B'
 Rp: -8.42030017E-05
 Description:
 1. PC give the command
 "Send^? 7 '4D4541533A44433F3B'" and '0d' and '0a'
 to GPIB-CONTROLLER.
 2. Hex format of data '4D4541533A44433F3B' will be interpreted
 to binary format of 'MEAS:DC?;' by GPIB-CONTROLLER
 automatically.
 3. Data 'MEAS:AC?' obtained from Step-2 will be transmitted
 to instrument A7 by GPIB-CONTROLLER.
 4. GPIB-CONTROLLER will read the output-buffer data of
 instrument A7 .
 5. GPIB-CONTROLLER will delete the EOS (End of String) added
 with the data obtained from item-4.
 6. The data left from item-5 will be added with 0d+0a at the rear
 and then transmitted back to PC by GPIB-CONTROLLER.

Exam: Send? 7 '*CLS;''MEAS:DC?;'
 Rp: -1.40907418E-04
 Description:
 1. PC give the command
 "Send? 7 '*CLS;''MEAS:DC?;'" and '0d' and '0a'
 to GPIB-CONTROLLER.
 2. The instruction of '*CLS;' will be transmitted to A7 by
 GPIB-CONTROLLER
 3. The instruction of 'MEAS:DC?;' will be transmitted to A7 by
 GPIB-CONTROLLER
 4. GPIB-CONTROLLER will read the data in the output-buffer
 of instrument A7 and stop reading until EOI signal is received.
 5. GPIB-CONTROLLER will delete the EOS (End of String) added
 with the data obtained from item-4.
 6. The data left from item-5 will be added with 0d+0a at the rear
 and then transmitted back to PC by GPIB-CONTROLLER.

Exam: Send? 7 '*IDN?'
 Rp: Agilent Technologies,E5071B,MY42404423,A.06.50

Exam: Send# 7 'MEAS:VOLT:AC?'
 Rp:2B312E38363839383335344452D30330A
 Description:
 1. PC give the command
 "Send# 7 'MEAS:VOLT:AC?'" and '0d' and '0a'
 to GPIB-CONTROLLER.
 2. The instruction of 'MEAS:VOLT:AC?' will be transmitted to A7 by
 GPIB-CONTROLLER
 3. GPIB-CONTROLLER will read the data in the output-buffer
 of instrument A7 and stop reading until EOI signal is received.
 4. GPIB-CONTROLLER will delete the EOS (End of String) added
 with the data obtained from item-3.
 5. The data left from item-4 will be interpreted to the Hex format.
 6. The data of the Hex format obtained from item 5 will be
 transmitted back to PC.

Exam: Send? 7 4 '*idn?;'
 Rp: Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09;HEWLETT-PACKARD,6611C,0,A.01.03
 Description:
 1. PC give the command

"Send? 7 4 '*idn?;'" and '0d' and '0a'
to GPIB-CONTROLLER.

2. The instruction of '*IDN?;' will be transmitted to instrument A7 and A4 simultaneously by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7.
4. GPIB-CONTROLLER will make that the EOS (End of String) put in the center of the data obtained from item 3 will be replaced with character of ';', and EOS put in the rear of the data obtained from item 3 will be replaced with character of '0d+0a'.
5. The data obtained from item 4 will be transmitted back to PC.
6. GPIB-CONTROLLER will read the data in the output-buffer of instrument A4 and stop reading until EOI signal is received.
7. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-6.
8. The data left from item-7 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam:SEND? 7 ':CONF:VOLT:DC 10,0.1;:TRIG:COUN 1000''INIT'':FETC?'
Rp: +5.77112684E-03,+5.10888123E-03,+4.42670484E-03,+3.71256154E ….-
Description:
1. PC give the command
   "SEND? 7 ':CONF:VOLT:DC 10,0.1;:TRIG:COUN 500''INIT'':FETC?'"
   and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of ':CONF:VOLT:DC 10,0.1;:TRIG:COUN 500' will be transmitted to A7 by GPIB-CONTROLLER
3. The instruction of 'INIT' will be transmitted to A7 by GPIB-CONTROLLER
4. The instruction of ':FETC?' will be transmitted to A7 by GPIB-CONTROLLER
5. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7 and stop reading until EOI signal is received.
6. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-5.
7. The data left from item-6 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.
   (500 Readings: total 16001 bytes)

Exam: Send 7'CONF:VOLT:DC' 'TRIG:SOUR BUS''INIT';MsgTRG 7;Send# 7'FETC?'
Rp: 2D322E3131313037383236452D30340A
Description:
1. PC give the command
   " Send 7'CONF:VOLT:DC' 'TRIG:SOUR BUS''INIT';MsgTRG 7;Send# 7'FETC?' "
   and '0d' and '0a'
   to GPIB-CONTROLLER by PC.
2. GPIB-CONTROLLER will transmite the instruction of 'CONF:VOLT:DC' to instrument A7
3. GPIB-CONTROLLER will transmite the instruction of 'TRIG:SOUR BUS' to instrument A7
4. The instruction 'INIT' will be transmitted to instrument A7 by GPIB-CONTROLLER
5. The signal of IEEE488.1-Message-GET will be transmitted to instrument A7 by GPIB-CONTROLLER
6. The instruction of ':FETC?' will be transmitted to instrument

    A7 by GPIB-CONTROLLER
 7. GPIB-CONTROLLER will read the data in the output-buffer
    of instrument A7 and stop reading until EOI signal is received .
 8. GPIB-CONTROLLER will delete the EOS (End of String) added
    with the data obtained from item-3.
 9. The result of data left from item-8 will be transformed to the
    Hex format.
 10. The Hex format of data obtained from item 9 and '0d' and '0a'
    will be transmitted back to PC.

Exam: Send 7'CONF:VOLT:DC' 'TRIG:SOUR BUS''INIT';MsgTRG 7;Send? 7'FETC?'

Rp: -2.11107826E-04

Description:

1. PC give the command
   " Send 7'CONF:VOLT:DC' 'TRIG:SOUR BUS''INIT';MsgTRG 7;Send? 7'FETC?'"
   and '0d' and '0a'
   to GPIB-CONTROLLER by PC.
2. GPIB-CONTROLLER will transmite the instruction of
   'CONF:VOLT:DC' to instrument A7
3. GPIB-CONTROLLER will transmite the instruction of
   'TRIG:SOUR BUS' to instrument A7
4. The instruction 'INIT' will be transmitted to instrument A7 by
   GPIB-CONTROLLER
5. The signal of IEEE488.1-Message-GET will be transmitted
   to instrument A7 by GPIB-CONTROLLER
6. The instruction of ':FETC?' will be transmitted to instrument
   A7 by GPIB-CONTROLLER
7. GPIB-CONTROLLER will read the data in the output-buffer
   of instrument A7 and stop reading until EOI signal is received .
8. GPIB-CONTROLLER will delete the EOS (End of String) added
   with the data obtained from item-3.
9. The result of the data left from item-8 and '0d' and '0a' will
   be transmitted back to PC.

Exam:IEEEidn$

```
Rp: 01:Agilent Technologies,33220A,MY44046179,2.07-2.06-22-2
    04:HEWLETT-PACKARD,6611C,0,A.01.03 "
    05:TEKTRONIX,TDS 220,0,CF:91.1CT FV:v2.03 TDS2CM:CMV:v1.04
    06:WAYNE KERR,4235,0,1.85
    07:Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
    08:ADVANTEST,R3162,110600533,F04
    09:AGILENT TECHNOLOGIES,DSO-X 2012A,MY52132806,02.10.2012022200
    10:ADVANTEST,R3273,110501576,G02
    11:HEWLETT-PACKARD,54520A,0          ,00.13,01.10,02.10,01.01
    12:ANRITSU,MT9810A,0,V1.0,V2.0
    13:Agilent Technologies,E4418A,,A1.09.01
    14: Agilent Technologies,8960 Series 10 E5515C,GB45360238,A.08.14
    15:HEWLETT-PACKARD,54820A,US40380133,A.04.50
    17:Agilent Technologies,E5071B,MY42404423,A.06.50
    18:Hewlett-Packard, E4402B, US39441101, A.07.05;+0
    20:ROHDE&SCHWARZ,NRVD, 100036.002,V1.52  V1.40
    28:Rohde&Schwarz,SMT06,830723/003,4.11
```

Description:

1. PC give the command
   "IEEEidn$" and '0d' and '0a'
   to GPIB-CONTROLLERC.
2. The instruction of '*IDN?;' will be transmitted simultaneously

to the whole instruments of AAn which are connected with GPIB-CONTROLLER through gpib-cable
for example:
(A1,A4,A5,A6,A7,A8,A9,A10,A11,A12, A13,A14,A15,A17,A18,A20,A28)
GPIB-CONTROLLER transmite the instruction of '*IDN?;' simultaneously to all instruments" it mean that:
the instruction of '*IDN?;' is sent to all instruments with broadcasting only a time by GPIB-CONTROLLER, all instruments will receive the instruction simultaneously.

3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A1 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will make the EOS (End of String) added with the data obtained from item-3 to be replaced with character of '0d+0a'
5. the data obtained from item-4 will be transmitted back to PC by GPIB-CONTROLLER.
6. GPIB-CONTROLLER will read the data in the output-buffer of instrument A4 and stop reading until EOI signal is received .
7. GPIB-CONTROLLER will make the the EOS (End of String) added with the data obtained from item-6 to be replaced with character of '0d+0a' .
8. the data obtained from item-7 will be transmitted back to PC by GPIB-CONTROLLER.
9. GPIB-CONTROLLER will read the data in the output-buffer of instrument A5 and stop reading until EOI signal is received.
10. GPIB-CONTROLLER will make the EOS (End of String) added with the data obtained from item-9 to be replace with character of '0d+0a'.
11. the data obtained from item-10 will be transmitted back to PC by GPIB-CONTROLLER.

....
....

ii. GPIB-CONTROLLER will read the data in the output-buffer of instrument A28 and stop reading until EOI signal is received.
jj. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-ii.
kk. The data left from item-jj will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Examples of Model E5515C:

Exam: Send? 14 '*IDN?;'

Rp: Agilent Technologies,8960 Series 10 E5515C,GB45360238,A.08.14

Description:

1. PC give the command
   "SEND? 14 '*IDN?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of '*IDN?' will be transmitted to A14 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A14 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added

with the data obtained from item-3.
5. The data left from item-4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam: Send= 14 'Afg:Freq?''AFG:VOLT?''CALL:BAND?''SYST:DATE?''SYST:UTC?'
Rp: +1.00000000E+003;+0.00000000E+000;PGSM;+2012,+6,+25;+14,+33,+22

Exam: Send= 14 'Afg:Freq?''AFG:VOLT?''CALL:BAND?''SYST:DATE?''SYST:UTC?''syst:tzon?'
Rp: +1.00000000E+003;+0.00000000E+000;PGSM;+2012,+6,+25;+14,+34,+47;+0,+0

Exam: Send= 14 'CALL:STAT:TRAF?''CALL:STAT:TRAF:CELL1:LEV:SEL?'
'CALL:STAT:TRAF:CELL1:LEV?''CALL:STAT:TRAF:CELL1?'
Rp: 9.91E37;9.91E37;9.91E37;9.91E37

Exam: Send= 14 'CALL:ORIG:DONE?''CALL:ORIG:OPC?''CALL:OPER:MODE?''CALL:PAG:STAT?'
Rp: 1;1;CALL;0

Exam: Send= 14 'CALL:STAT:SYNC?''CALL:STAT:SYNC:LEV?''CALL:STAT:SYNC:LEV:SEL?'
Rp: +8.00000000E+003;+8.00000000E+003;+8.00000000E+003

Description:

1. PC give the command
 "Send= 14 'CALL:STAT:SYNC?''CALL:STAT:SYNC:LEV?''CALL:STAT:SYNC:LEV:SEL?'"
 and '0d' and '0a'
 to GPIB-CONTROLLER.
2. The instruction of 'CALL:STAT:SYNC?' will be transmitted to A14 by GPIB-CONTROLLER
3. The data in the output-buffer of instrument A14 will be read by GPIB-CONTROLLER.
4. GPIB-CONTROLLER will make that the EOS (End of String) put in the center of the data obtained from item 3 will be replaced with the character of '; and the EOS put in the rear of the data obtained from item 3 will be replaced with the character of '0d+0a'.
5. Data obtained from item 4 will be transmitted to PC by GPIB-CONTROLLER.
6. The instruction of 'CALL:STAT:SYNC:LEV?' will be transmitted to A14 by GPIB-CONTROLLER
7. The data in the output-buffer of instrument A14 will be read by GPIB-CONTROLLER.
8. GPIB-CONTROLLER will make that the EOS (End of String) put in the center of the data obtained from item 7 will be replaced with the character of ';', and the EOS put in the rear of the data obtained from item 3 will be replaced with the character of '0d+0a'.
9. Data obtained from item 4 will be transmitted to PC by GPIB-CONTROLLER.
10. The instruction of 'CALL:STAT:SYNC:LEV:SEL?'" will be transmitted to A14 by GPIB-CONTROLLER
11. The data in the output-buffer of instrument A14 will be read by GPIB-CONTROLLER.
12. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-11.
13. The data left from item-12 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam: Send= 14 'CALL:STAT:MSP?''CALL:STAT:MS:ANAL:TXL?''CALL:STAT:LOOP?'
Rp: NORM;9.91E37;0

Examples of Model 33220A:

Exam: Send? 1 '*IDN?;'
Rp: Agilent Technologies,33220A,MY44046179,2.07-2.06-22-2

Exam:SEND? 1 'OUTP:LOAD 50''APPL:SIN 5000,5''FM:INT:FUNC SIN'
'FM:INT:FREQ 500''FM:DEV 250''DATA:CAT?'
Rp: +1.00000000E+003;+0.00000000E+000;PGSM;+2012,+6,+25;+14,+34,+47;+0,+0

Description:
  1. PC give the command
     "SEND? 1 'OUTP:LOAD 50''APPL:SIN 5000,5''FM:INT:FUNC SIN'
              'FM:INT:FREQ 500''FM:DEV 250''DATA:CAT?'"
     and '0d' and '0a'
     to GPIB-CONTROLLER.
  2. The instruction of 'OUTP:LOAD 50' will be transmitted to A14 by
     GPIB-CONTROLLER
  3. The instruction of 'APPL:SIN 5000,5' will be transmitted to A14 by
     GPIB-CONTROLLER
  4. The instruction of 'FM:INT:FUNC SIN' will be transmitted to A14 by
     GPIB-CONTROLLER
  5. The instruction of 'FM:INT:FREQ 500' will be transmitted to A14 by
     GPIB-CONTROLLER
  6. The instruction of 'FM:DEV 250' will be transmitted to A14 by
     GPIB-CONTROLLER
  7. The instruction of 'DATA:CAT?' will be transmitted to A14 by
     GPIB-CONTROLLER
  8. The data in the output-buffer of instrument A1 will
     be read by GPIB-CONTROLLER.
  9. GPIB-CONTROLLER will delete the EOS (End of String) added
     with the data obtained from item-8.
  10. The data left from item-9 will be added with 0d+0a at the rear
     and then transmitted back to PC by GPIB-CONTROLLER.

Exam:SEND= 1 'DATA:DAC VOLATILE, 2047,1536,1024,512,0,-512,-1536,-2047'

Exam:SEND= 1 'DATA:DAC VOLATILE, #216''07FF0600040002000000FE00FA00F801'

Exam:SEND= 1 'DISP:TEXT \'33120a Func. Gen.\'''SYSTem:BEEPer'

Exam:SEND= 1 'FUNC:USER?''SWE:TIME?''SWE:STAT?''TRIG:SOUR?''SYST:ERR?'
 Rp: EXP_RISE;+1.0000000000000E+00;0;IMM;+0,"No error"

Exam: Transmit 40 or more value to A4 as below:
     (InstructiOn of Heweett-Packard 33220a)
```
SENDl 4 'DATA:DAC VOLATILE,2047,1536,1024,512,0,-512,-1536,-2047,0,-512, 0,'
SENDl 4 '-1536,-2047,-512,-1536,-2047,1047,1236,1124,402,0,-512,-1136-1547,0,'
SEND  4 '712,-1236,-1447,-112, -1236,-1047,947,136,1624,462, 7,-312,-447192,0'
```
Description:
  Data of 40 byte or more will be transmitted to the instrument A4 through
  GPIB-CONTROLLER by PC
   (instruction of HEWLETT-PACKARD,33220A Function Generator)
  The process of work described as below:
  1. PC give the Command
     "SENDl 4 'DATA:DAC VOLATILE, 2047,1536,1024,512,0,-512,-1536,-2047,'"
     and '0d' and '0a'
     to GPIB-CONTROLLER
  2. The instruction of 'DATA:DAC VOLATILE, 2047,1536,1024,512,0,-512,-1536,
                       -2047,'
     will be transmitted to instrument A4 by GPIB-CONTROLLER.
  3. PC give the command
     "SENDl 4 '0,-512,-1536,-2047,-512,-1536,-2047,1047,1236,1124,402,0,' "
     and '0d' and '0a'
     to GPIB-CONTROLLER
  4. The data of '0,-512,-1536,-2047,-512,-1536,-2047,1047,1236,1124,402,0,'
     will be transmitted to instrument A4 by GPIB-CONTROLLER.
  5. PC give the command

"SEND| 4 '-512,-1136,-1547,0,-712,-1236,-1447,-112,'"
and '0d' and '0a'
to GPIB-CONTROLLER by PC

6. The data of '-512,-1136,-1547,0,-712,-1236,-1447,-112,' will be transmitted to instrument A4 by GPIB-CONTROLLER.
7. PC give the command
   "SEND 4 '-1236,-1047,947,136,1624,4627,-312,-447,-192,-1536,-1147'"
   and '0d' and '0a'
   to GPIB-CONTROLLER
8. Data '-1236,-1047,947,136,1624,4627,-312,-447,-192,-1536,-1147' in the rear will be added with a signal of EOI by GPIB-CONTROLLER
9. Data and EOI obtained from item-8 will be transmitted to instrument A4 by GPIB-CONTROLLER.

* The command "Send 4 ..." for item 1,3 and 5 mentioned above will be added with Key-Symbol of '|' in the rear, this means that the data are not wholly sent from PC yet.
The data for item 2,4 and 6 are transmitted to the instrument A4 through GPIB-CONTROLLER, and the handshake action between GPIB-CONTROLLER and instrument will not be stopped.

* The command "Send 4 ..." for item 7 mentioned above will not be added with Key-Symbol of '|' in the rear, this means that the data are completely sent from PC and the data item 8 are transmitted to instrument A4 through GPIB-CONTROLLER, and the handshake action between GPIB-CONTROLLER and instrument will be stopped by the signal of EOI.

Examples of Model HP6623A:

Exam:SEND? 5 'PON 1''DSP 1''UNMASK 1,255''OCP 1,1''OVRST 1''OVSET 1,5' 'SRQ 1''ID?'

Rp: HP6623A

Exam:SEND= 5 'VSET 1,3''VSET? 1''VOUT? 1'

Rp: 2.999; 3.001

Examples of Model E4402B:

Exam: Send? 1 '*IDN?;'

Rp: Hewlett-Packard, E4402B, US39441101, A.07.05;+0

Exam: Send= 18 'DISP:WIND:TRAC:Y:SCAL:PDIV 5''SENS:FEED AREF''sens:freq:cent?'

Rp: +5.00000000E+007

Description:

1. PC give the command
   "Send= 18 'DISP:WIND:TRAC:Y:SCAL:PDIV 5'
   'SENS:FEED AREF''sens:freq:cent?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'DISP:WIND:TRAC:Y:SCAL:PDIV 5' will be transmitted to instrument A18 by GPIB-CONTROLLER
3. The instruction of 'SENS:FEED AREF'will be transmitted to instrument A18 by GPIB-CONTROLLER
4. The instruction of 'sens:freq:cent?' will be transmitted to instrument A18 by GPIB-CONTROLLER
5. GPIB-CONTROLLER will read the data in the output-buffer of instrument A18 and stop reading until EOI signal is received.
6. GPIB-CONTROLLER will delete the EOS (End of String)

added with the data obtained from item-5.
7. The data left from item-6 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

Exam:SEND= 18 'sens:freq:cent?''CALC1:MARK1:X?''CALC1:MARK1:Y?'
Rp: +5.00000000E+007;+5.00000000000E+006;+0.00000E+00

Exam:SEND= 18 ':STAT:QUES:CAL:COND?''SYST:COMM:GPIB:SELF:ADDR?''SENS:SWE:POIN?'
Rp: +0;+18;+401

Exam: Send* 18 ':TRAC:DATA? TRACE1'
Rp: -6.09970E+01,-6.09970E+01,-6.09970E+01,-6.09970E+01,-6.09970E+01, …

Description:
1. PC give the command
   "Send* 18 ':TRAC:DATA? TRACE1'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of ':TRAC:DATA? TRACE1' will be transmitted to A18 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A18 and stop reading until EOI signal is received.
4. Data obtained from item 3 will be transmitted back to PC by GPIB-CONTROLLER.

Examples of Model HP8591EM:

Exam: Send= 18 'ID?;''SER?;''REV?;''SETDATE?;''SETTIME?;'
Rp: HP8591EM;976;960709;120621;140932

Exam: Send* 22 'TDF M;''TRA?;'
Rp: 5625,6636,6161,3446,1226,1264,1136,1294,1134,1087,1309,1239, …

Description:
1. PC give the command
   "Send* 22 'TDF M;''TRA?;'" and '0d' and '0a'
   to GPIB-CONTROLLER .
2. The instruction of 'TDF M;'will be transmitted to A22 by GPIB-CONTROLLER
3. The instruction of 'TRA?;'will be transmitted to A22 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data of 2007 byte in the output-buffer of instrument A22 and stop reading until EOI signal is received.
5. Data obtained from item 4 will be transmitted back to PC by GPIB-CONTROLLER.

Exam: Send= 18 'IP;''KEYCLR;''ID?;''CF 300MHZ;SP 2MHZ;RB100KHZ;''CF?''rb?''RL?'
'RLPOS?''SER?''REV?''SETDATE?''SETTIME?''SQLCH?''SS?''SRCTK?'
'SRCALC?''SRCPSTP?''SRCPWR?''SRCPSWP?''SRCNORM?''SRCPOFS?''MEM?'
'AMB?''AMBPL?''AMPLEN?''ANLGPLUS?'
Rp: HP8591EM;300000000;100000;0;8;976;960709;120621;141459;0; 100000000; 0;
INT;10.00;-10.00;0;OFF;0;261414;OFF;OFF;-1;OFF

Exam: Findlisten? 16 17
Rp: 16,17

Examples of Model 8753D:

Exam:SEND? 16 '*IDN?'
Rp: HEWLETT PACKARD,8753D,0,6.14

Exam:SEND 16 'PRES'

Description:
1. PC give the command
   "SEND 16 'PRES'" and '0d' and '0a'
   to GPIB-CONTROLLER.

2. The instruction of 'PRES' will be transmitted to A16 by GPIB-CONTROLLER

Exam: AgiETB_ 16 'FORM2;''OUTPDATA;'

Rp: 4,1607

Description:

1. PC give the command "AgiETB_ 16 'FORM2;''OUTPDATA;'" and '0d' and '0a' to GPIB-CONTROLLER.
2. The instruction of 'FORM2;' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. The byte number of data-head and data-body in the output-buffer of instrument A16 will be read by GPIB-CONTROLLER.
5. The data obtained from item 4 will be added with 0d+0a at the rear by GPIB-CONTROLLER.
6. The data and characters of od+0a obtained from item 5 will be transmitted to PC by GPIB-CONTROLLER.

Exam: AgiETB_ 16 'FORM1;''OUTPDATA;'

Rp: 4,1206

Exam: AgiETB_ 16 'FORM3;''OUTPDATA;'

Rp: 4,3216

Exam: AgiETB? 16 'FORM1;''OUTPDATA;'

Rp: #A.....B...$....."....`..........N.d...…

Description:

1. PC give the command "AgiETB? 16 'FORM1;''OUTPDATA;'" and '0d' and '0a' to GPIB-CONTROLLER.
2. The instruction of 'FORM1;' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in the output-buffer of instrument A16 and stop reading until EOI signal is received.
5. The data of 1210-Byte obtained from item 4 will be transmitted to PC by GPIB-CONTROLLER.

Exam: AgiETB< 16 'FORM1;''OUTPDATA;'

Rp: ...D..."........\..........V.t........T..........p.....N. …

Description:

1. PC give the command "AgiETB< 16 'FORM1;''OUTPDATA;'" and '0d' and '0a' to GPIB-CONTROLLER.
2. The instruction of 'FORM1;' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. The Data-Body in the output-buffer of instrument A16 will be read by GPIB-CONTROLLER.
5. GPIB-CONTROLLER will transmite the Data-Body of 1206-Bytes obtained from item 4 back to PC.

Exam: Send* 16 'FORM2;''OUTPDATA;'

Rp: #A.H.........D.>.......>r .....>5@...|.=.....<.>....…

Description:

1. PC give the command
    "send* 16 'FORM2''OUTPDATA;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'FORM1;' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in the output-buffer of instrument A16 and stop reading until EOI signal is received.
5. GPIB-CONTROLLER will transmite data of 1611-Byte obtained from item 4 back to PC.

Exam: Send* 16 'FORM3;''OUTPDATA;'

```
 Rp: #A...........!............?.L............?.H........... …
```

 Description:

1. PC give the command
    "send* 16 'FORM3;''OUTPDATA;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'FORM3' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in the output-buffer of instrument A16 and stop reading until EOI signal is received.
5. GPIB-CONTROLLER will transmite the data of 3220-Byte obtained from item 4 back to PC.

Exam:ARBITRARY? 1024 16 'FORM1;''OUTPDATA;'

```
 Rp: #A...p.P...$.....$....\..........P.d........ ….
```

 Description:

1. PC give the command
    "ARBITRARY? 1024 16 'FORM1''OUTPDATA;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'FORM3' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the 1024-byte of data in output-buffer of instrument A16 .
5. GPIB-CONTROLLER will transmite the data obtained from item 4 back to PC.

Exam:send* 16 'FORM4;''OUTPDATA;'

```
 Rp: -4.472656000000000E-01,  -1.134766000000000E+00.  -1.377075000000 …
```

 Description:

1. PC give the command
    "ARBITRARY? 1024 16 'FORM1''OUTPDATA;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'FORM4' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in output-buffer of instrument A16 and stop reading until EOI signal is received.
5. GPIB-CONTROLLER will transmite the data obtained

        from item 4 back to PC.

Examples of Model 8752a:

Exam: Send? 16 'IDN?;'
 Rp: HEWLETT PACKARD,8752A,0,1.01

Exam:SEND* 16 'FORM1;''OUTPDATA;'
 Rp: #A.../BE...QA..... ....v 2...C ..... p.... #]..e e..2 >.... d... …
  Description:
   1. PC give the command
       "SEND* 16 'FORM1''OUTPDATA;'" and '0d' and '0a'
      to GPIB-CONTROLLER.
   2. The instruction of 'FORM1' will be transmitted to A16 by
      GPIB-CONTROLLER
   3. The instruction of 'OUTPDATA;' will be transmitted to A16
      by GPIB-CONTROLLER
   4. GPIB-CONTROLLER will read the data in output-buffer
      of instrument A16 and stop reading until EOI signal is
      received.
   5. GPIB-CONTROLLER will transmite the data obtained
      from item 4 back to PC.

Exam: AgiETB? 16 'FORM1;''OUTPDATA;'
 Rp: #A...'B4..._B..... w.... ...? ...c `.... .c..g C... I...k H. …
  Description:
   1. PC give the command
       "AgiETB? 16 'FORM1''OUTPDATA;'" and '0d' and '0a'
      to GPIB-CONTROLLER.
   2. The instruction of 'FORM1' will be transmitted to A16 by
      GPIB-CONTROLLER
   3. The instruction of 'OUTPDATA;' will be transmitted to A16
      by GPIB-CONTROLLER
   4. GPIB-CONTROLLER will read the data in output-buffer
      of instrument A16 and stop reading until EOI signal is
      received.
   5. GPIB-CONTROLLER will transmite the data of 1210 bytes
      obtained from item 4 back to PC.

Exam:SEND* 16 'FORM4;''OUTPDATA;'
 Rp: 1.034363000000000E+00, 0.037231000000000E-01. 1.031250000000000E+00, …
  Description:
   1. PC give the command
       "SEND* 16 'FORM4''OUTPDATA;'" and '0d' and '0a'
      to GPIB-CONTROLLER.
   2. The instruction of 'FORM1' will be transmitted to A16 by
      GPIB-CONTROLLER
   3. The instruction of 'OUTPDATA;' will be transmitted to A16 by
      GPIB-CONTROLLER
   4. GPIB-CONTROLLER will read the data in output-buffer
      of instrument A16 and stop reading until EOI signal is
      received.
   5. GPIB-CONTROLLER will transmite the data of 10054 bytes obtained
      from item 4 back to PC.

Exam: AgiETB? 16 'FORM4;''OUTPDATA;'
 Rp: 1.035217000000000E+00,   0.023804000000000E-01. …
  Description:
   1. PC give the command
       "AgiETB? 16 'FORM4''OUTPDATA;'" and '0d' and '0a'
      to GPIB-CONTROLLER.

2. The instruction of 'FORM1' will be transmitted to A16 by GPIB-CONTROLLER
3. The instruction of 'OUTPDATA;' will be transmitted to A16 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data of 10054 bytes in output-buffer of instrument A16.
5. GPIB-CONTROLLER will transmite the data of 10054 bytes obtained from item 4 back to PC.

Examples of Model E5071B:

```
 Exam:Findlisten? 17
  Rp: 17
 Exam:SEND 17 '  *IDN?'
  Rp:  Agilent Technologies,E5071B,MY42404423,A.06.50
 Exam:Send 18 'VARDEF D_ADDRESS,0;''CLRDSP;''PUPA 100,180;TEXT %Measurement%'
              'MOVE D_ADDRESS,DA;''PUPA 100,100;TEXT%Signal found%;'
 Exam:SEND 17 '*RST'
 Exam:SEND* 17 ':FORM:DATA ASC'':CALC1:DATA:SDAT?'
  Rp: +1.00489401832E+000,+1.14720556199E-002,+1.00375867860E+000, …
```

Description:
1. PC give the command
   "SEND* 17 ':FORM:DATA ASC'':CALC1:DATA:SDAT?'"
   and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of ':FORM:DATA ASC' will be transmitted to A17 by GPIB-CONTROLLER
3. The instruction of ':CALC1:DATA:SDAT?' will be transmitted to A17 by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in output-buffer of instrument A16 and stop reading until EOI signal is received.
5. GPIB-CONTROLLER will transmite data of 8085 bytes obtained from item 4 back to PC.

```
 Exam:Send$ '*idn?;'
  Rp: 04:HEWLETT-PACKARD,6611C,0,A.01.03 "
      07:Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
      09:AGILENT TECHNOLOGIES,DSO-X 2012A,MY52132806,02.10.2012022200
      17:Agilent Technologies,E5071B,MY42404423,A.06.50
```

Description:
PC give the instruction of '*idn?'to instruments AAn through GPIB-CONTROLLER, the KeyWord of 'Send' is added with a Key-Symbol of '$' of which function described as below:
1. PC will read output-buffer data of instruments AAn through GPIB-CONTROLLER.
2. The value of the each GPIB-Address of instrument An will be added with the data in the front which is read from output-buffer of each instrument through GPIB-CONTROLLER.
3. The character of 0d+0a will be added at the rear of the data which are read from the output buffer of each instrument An.
4. All of the address value, data, and character of 0d+0a obtained from item 2 and 3 (above mentioned) will be transmitted back to PC through GPIB-CONTROLLER.

Examples of Model 6611C:

```
 Exam:SEND? '*idn?'
  Rp: HEWLETT-PACKARD,6611C,0,A.01.03;Agilent Technologies,MY47013754,
      34410A,2.35-2.35-0.09-46-09;AGILENT TECHNOLOGIES,DSO-X2012A,
      MY52132806,02.10.2012022200
   Description:
     1.  PC give the command
          "SEND? '*idn?'" and '0d' and '0a'
         to GPIB-CONTROLLER.
     2.  The instruction  of '*idn?;' will be transmitted simultaneously
         to all instruments of AAn which are connected with
         GPIB-CONTROLLER through gpib-cable ,
         for example:
             A4, A7, A9, A11
         GPIB-CONTROLLER transmite the instruction of '*IDN?;'
          simultaneously to instruments" it mean that:
         the instruction of '*IDN?;' is sent to instruments with
         broadcasting only a time by GPIB-CONTROLLER,
         all instruments  will receive the instruction simultaneously.
     3.  GPIB-CONTROLLER will read the data in the output-buffer
         of instrument A4 and stop reading until EOI signal is
         received.
     4.  GPIB-CONTROLLER will make the EOS (End of String) added
         with the data obtained from item-3 to be replaced with the
         character of ';'.
     5.  the data obtained from item-4 will be transmitted back to PC by
         GPIB-CONTROLLER.
     6.  GPIB-CONTROLLER will read the data in the output-buffer
         of instrument A7 and stop reading until EOI signal is
         received.
     7.  GPIB-CONTROLLER will make the EOS (End of String) added
         with the data obtained from item-3 to be replaced with the
         character of ';'.
     8.  the data obtained from item-7 will be transmitted to PC by
         GPIB-CONTROLLER.
     9.  GPIB-CONTROLLER will read the data in the output-buffer
         of instrument A9 and stop reading until EOI signal is
         received.
     10. GPIB-CONTROLLER will make the EOS (End of String) added
         with the data obtained from item-3 to be replaced with the
         character of ';'.
     11. GPIB-CONTROLLER will read the data in the output-buffer
         of instrument A11 and stop reading until EOI signal is
         received.
     12. The data in the output-buffer of instrument A11 will be read
         until EOI signal is received by GPIB-CONTROLLER.
     13. GPIB-CONTROLLER will make the EOS (End of String) added
         with the data obtained from item-3 to be replaced with the
         character of '0d+0a'.
     14. the data obtained from item-13 will be transmitted to PC by
         GPIB-CONTROLLER.
 Exam:SEND= 4 '*cls''outp on''volt 4''curr 0.1''meas:volt?''meas:curr?''*idn?;'
  Rp: 3.99897E+0;-3.00838E-4;HEWLETT-PACKARD,6611C,0,A.01.03
 Exam:send? 4 '*cls''outp on''volt 4''curr 0.1''curr:prot:stat on''meas:volt?'
               'meas:curr?';send? 4 '*idn?;'
  Rp: 3.99882E+0;-3.06621E-4;HEWLETT-PACKARD,6611C,0,A.01.03
```

Examples of Model DSO-X 2012A:

```
 Exam:IEEEIDN? 9
  Rp: AGILENT TECHNOLOGIES,DSO-X 2012A,MY52132806,02.10.2012022200
```

  Description:
    1. PC give the command
        "IEEEIDN? 9" and '0d' and '0a'
       to GPIB-CONTROLLER.
    2. The instruction of '*idn?' will be transmitted to A9 by
       GPIB-CONTROLLER
    3. GPIB-CONTROLLER will read the data in the output-buffer
       of instrument A9 and stop reading until EOI signal is
       received.
    4. GPIB-CONTROLLER will delete the EOS (End of String)
       added with the data obtained from item-3.
    5. The data left from item-4 will be
       added with character of 0d+0a at the rear and then transmitted
       back to PC by GPIB-CONTROLLER.

Exam:ARBITRARY? 100 9 'SYST:SET?;'
 Rp: #800076354<setup prod="InfiniiVision" ver="02.10.2012022200"…
  Description:
    1. PC give the command
        data "ARBITRARY? 100 9 'SYST:SET?'" and '0d' and '0a'
       to GPIB-CONTROLLER.
    2. The instruction of 'SYST:SET?' will be transmitted to A9 by
       GPIB-CONTROLLER
    3. GPIB-CONTROLLER will read the foremost 100 bytes of
       data in output-buffer of instrument A9.
    4. The 100 bytes of data obtained from item-3 will be transmitted
       back to PC by GPIB-CONTROLLER.

Exam:ARBITRARY? 10240 9
 Rp: rol_section>…
  Description:
    1. PC give the command
        data "ARBITRARY? 10240 9" and '0d' and '0a'
       to GPIB-CONTROLLER.
    2. GPIB-CONTROLLER will read the next 10240 bytes of data
       in output-buffer of instrument A9.
    3. The 10240 bytes data obtained from item-2 will be transmitted
       back to PC by GPIB-CONTROLLER.

Exam:ARBITRARY? 66024 9
 Rp: #800076354<setup prod="InfiniiVision" ver="02.10.2012022200" srver="2.0">
  Description:
    1. PC give the command
        data "ARBITRARY? 66024 9" and '0d' and '0a'
       to GPIB-CONTROLLER.
    2. GPIB-CONTROLLER will read the next 66024 bytes of data
       in output-buffer of instrument A9.
    3. The 66024 bytes data obtained from item-2 will be transmitted
       back to PC by GPIB-CONTROLLER.

Exam:ARBITRARY? 76354 'SYST:SET?;'
  Rp: #800076354<setup prod="InfiniiVision" ver="02.10.2012022200" srver="2.0…
   Description:
     1. PC give the command
         data "ARBITRARY? 76364 9 'SYST:SET?;'" and '0d' and '0a'
        to GPIB-CONTROLLER.
     2. The instruction of 'SYST:SET?;' will be transmitted to A9 by
        GPIB-CONTROLLER
     3. GPIB-CONTROLLER will read the foremost 76364 bytes of

data in output-buffer of instrument A9.
4. The 76364 bytes of data obtained from item-3 will be transmitted back to PC by GPIB-CONTROLLER. .

Exam:SEND* 9 'SYST:SET?;'
Rp: #800076354<setup prod="InfiniiVision" ver="02.10.2012022200" srver="2.0…
Description:
1. PC give the command
   data "SEND* 9 'SYST:SET?;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'SYST:SET?;' will be transmitted to A9 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in output-buffer of instrument A7 and stop reading until EOI signal is received.
4. The 76364 bytes of data (10 bytes of blockhead is included) obtained from item-3 will be transmitted back to PC by GPIB-CONTROLLER.

Exam:IEEEB? 9 'SYST:SET?;'
Rp: #800076354<setup prod="InfiniiVision" ver="02.10.2012022200" srver="2.0…
Description:
1. PC give the command
   "IEEEB 9 'SYST:SET?;';IEEEB? 9a" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction 'SYST:SET?;' will be transmitted to A9 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in output-buffer of instrument A9 when total byte number of receiving data are coincedental with the regulation for block-head of data obtained from instrument, GPIB-CONTROLLER will stop reading data automatically.
4. The 76364 bytes of data obtained from item-3 will be transmitted back to PC by GPIB-CONTROLLER,
   (10 bytes of BLOCK-HEAD '#800076354' will be included in the 76364 bytes of data)

Exam:IEEEB_ 9 'SYST:SET?;'
Rp: 10,76354.
Description:
1. PC give the command
   " IEEEB_ 9 'SYST:SET?;'1000a" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'SYST:SET?;' will be transmitted to A9 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the number of block-Head count and Block-Body count of the Block-Data from output buffer of instrument A9.
4. GPIB-CONTROLLER will transmite the data obtained from item-3 will be transmitted back to PC.

Exam:IEEEB< 9 'SYST:SET?'
Rp: <setup prod="InfiniiVision" ver="02.10.2012022200" srver="2.0"> …
Description:
1. PC give the command
   "IEEEB< 9 'SYST:SET?;'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of 'SYST:SET?;' will be transmitted to A9 by GPIB-CONTROLLER

3. GPIB-CONTROLLER will read the data from output buffer of instrument A9 when total byte number of receiving data are coincedental with the regulation of Block-Head of Block-Data obtained from instrument A9, GPIB-CONTROLLER will stop reading data automacically.
4. The 76354 bytes of Block-Data obtained from item-3 will be transmitted back to PC by GPIB-CONTROLLER.
   (10 bytes of BLOCK-HEAD '#800076354' will not be included in the 76354 bytes of Block-Data)

Exam: TestSys?
 Rp: 04:0;07:0;09:0
  Description:
   '0' is sent from A4,A7 and A9 to PC, it is meaning that instrument A4,A7 and A9 is ok.

Exam: TestSys#
 Rp: 043007300930
  Description:
   The character of '30' is sent from A4,A7 and A9 to PC, it is meaning that instrument A4,A7 and A9 is ok. (Hex '30' = Binary '0')

1. PC give the command
    "TestSys#" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of `*tst?` will be transmitted to instrument A4, A7 and A9 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A4 and stop reading until EOI signal is received.
4. The the data obtained from item-3 will be interpreted to Hex-Format, and then transmitted to PC by GPIB-CONTROLLER.
5. GPIB-CONTROLLER will read the data in the output-buffer of instrument A7 and stop reading until EOI signal is received .
6. The the data obtained from item-5 will be interpreted to Hex-Format, and then transmitted to PC by GPIB-CONTROLLER.
7. GPIB-CONTROLLER will read the data in the output-buffer of instrument A9 and stop reading until EOI signal is received.
8. The the data obtained from item-7 will be interpreted to Hex-Format added with characters of 0d+0a at the rear and then transmitted to PC by GPIB-CONTROLLER.

Examples of Model HP81101A:

Exam:SEND? 10 '*IDN?'
 Rp: HEWLETT-PACKARD,HP81101A,DE41B01746,REV 01.11.00

Exam:SEND= 10 ':SYST:PRES;''SYST:VERS?''SYST:WARN:COUN?''SYST:WARN:STR?'
            'SYST:WARN:BUFF?''SOUR:FREQ?'
 Rp: 1992.0;+0;"";+3456;+1.0000E+06

Exam:SEND* 10 ':STAT:PRES''SYST:SET?'
 Rp: #4648681101A..............2..................7'........5.7...
  Description:

1. PC give the command
    "SEND* 10 ':STAT:PRES''SYST:SET?'" and '0d' and '0a'
   to GPIB-CONTROLLER.
2. The instruction of ':STAT:PRES' will be transmitted to A10 by GPIB-CONTROLLER
3. The instruction of 'SYST:SET?;' will be transmitted to A10

by GPIB-CONTROLLER
4. GPIB-CONTROLLER will read the data in output-buffer of instrument A10 and stop reading until EOI signal is received.
5. GPIB-CONTROLLER will transmite the 6493 bytes of data obtained from item-4 back to PC.

```
Exam:SEND= 10 ':ARM:LEV?'':ARM:SEQ:FREQ?'':ARM:SEQ:SENS?'':ARM:SEQ:SLOP?'
              ':ARM:SEQ:SOUR?'':OUTP:NORM:STAT?;'
 Rp: +1.0000E+00;+1.0000E+05;EDGE;POS;IMM;0
```

Description:

1. PC give the command "SEND= 10 ':ARM:LEV?'...':OUTP:NORM:STAT?;'" and '0d' and '0a' to GPIB-CONTROLLER.
2. The instruction of ':ARM:LEV?' will be transmitted to A10 by GPIB-CONTROLLER
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument A10 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item-4 will be added with ';' at the rear and then transmitted back to PC by GPIB-CONTROLLER.

   ....

   ....

e. The instruction of ':OUTP:NORM:STAT?;' will be transmitted to A10 by GPIB-CONTROLLER
f. GPIB-CONTROLLER will read the data in the output-buffer of instrument A10 and stop reading until EOI signal is received.
g. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-f.
h. The data left from item-g will be added with characters of 0d+0a at the rear and then transmitted to PC by GPIB-CONTROLLER.

```
Exam:SEND 10 ':DISP:WIND:STAT OFF;'':DISP:WIND:STAT ON;'
Exam:Send= 10 ':SOUR:FREQ:AUTO ONCE;'':ARM:SOUR INT'':CURR:OFFS?'':CURR?'
              ':CURR:HIGH?'':CURR:LOW?'':CURR:LIM?'
  Rp: +0.0000E+00;+2.0000E-02;+1.0000E-02;-1.0000E-02;+1.0000E-02
Exam:SEND= 10 ':VOLT:OFFS?'':VOLT:HIGH?'':VOLT:LOW?'':VOLT:LIM?'
              ':VOLT:LIM:LOW?'':VOLT:LIM:STAT?'
  Rp: +0.0000E+00;+5.0000E-01;-5.0000E-01;+5.0000E-01;-5.0000E-01;0
```

5.4.5 Instructions--Receive-and-Transmit

There are 3 functions for this kinds of instructions:

Function1.

PC read data from instrument through GPIB-CONTROLLER and sent back to PC.

Exam: READ? 7

Function2.

PC send data to instrument through GPIB-CONTROLLER.

Exam: READ 7 'ISET 1 0.6'

Function3.

It is with both Function1 and Function2. Function1 will be executed at first

Exam:READ? 7 'VSET 1 2.7

Separate description of instructions:

READ	Text format data will be read from **An** or **AAn** and responded to **PC**
RdEOS	Same function as above 'READ', but EOS is decided by input-parameter instead of default **EOS**.
RdAgiETB	Agilent-8753ET-Form1,2,3,5 block data will be read from **An** and responded to **PC**
RdARBITRARY	The data of any format are read from **An** and sent to **PC**. the data read completely at a time or several times will be decided by parameter BLEN
RdIEEEB	**Block-Data** of format IEEE488-DEFINITE-LENGTH-ARBITRARY-BLOCK read from **An** and sent to **PC** through GPIB-CONTROLLER.

Syntax:

Key-Word	Key-Symbol	Param	Param	Param	Param	Param	Param
READ	@, ?,#	**AAn**	dS0..dSn		DlyR	DlyW	DlyB
RdEOS	@, ?,#	EOS	**AAn**	dS0..dSn	DlyR	DlyW	DlyB
RdAgiETB	@, ?,#	**An**			DlyR	DlyW	DlyB
RdARBITRARY	@, ?,#	BLEN	An		DlyR	DlyW	DlyB
RdIEEEB	@, ?,#	**An**			DlyR	DlyW	DlyB

Description:

Exam: Send 7 'MEAS:DC?;';READ? 7
 Rp: -1.13459428E-04
  Description:
   1. Pc give the command
      "Send 7 'MEAS:DC?;';READ? 7; " and '0d' and '0a'
      to GPIB-CONTROLLER.
   2. GPIB-CONTROLLER will transmite Instruction 'MEAS:DC?;'
      to instrument A7.
   3. GPIB-CONTROLLER will read the data in output-buffer
      of instrument A7 and transmitted the data back to PC.

Exam: Send 7 'MEAS:DC?;';READ# 7
 Rp: 2d312e3135323939393135452d3034
  Description:
   1. PC give the command
      "Send 7 'MEAS:DC?;';READ? 7; " and '0d' and '0a'
      to GPIB-CONTROLLER.
   2. The instruction of 'MEAS:DC?;' will be transmitted to instrument
      A7 by GPIB-CONTROLLER.
   3. GPIB-CONTROLLER will read the data in output-buffer of
      instrument A7 and transform the data into Hex format which will
      be transmitted back to PC.

Exam: Send 4 7 '*IDN?;';READ? 7 4;
 Rp: Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09;HEWLETTPACKARD,6611C,0,A.01.03
  Description:
   1. PC give the command
      "Send 4 7 '*IDN?;';READ? 7 4;" and '0d' and '0a'
      to GPIB-CONTROLLER.
   2. The Instruction of '*IDN?;' will be transmitted to instrument
      A4 and A7 by GPIB-CONTROLLER.
   3. GPIB-CONTROLLER will read the data in output-buffer of
      instrument A7 and transmite the data back to PC.
   4. GPIB-CONTROLLER will read the data in output-buffer of
      instrument A4 and transmite the data back to PC..

Exam:SEND 7 'TRIG:SOUR BUS';Send 7 'INIT';MsgTRG 7;Send 7 'FETC?';Read? 7

Rp: -1.38483849E-04
Description:
1. PC give the command
"SEND 7 'TRIG:SOUR BUS';Send 7 'INIT';MsgTRG 7;Send 7 'FETC?'; Read? 7"
and '0d' and '0a'
to GPIB-CONTROLLER.
2. The instruction of 'TRIG:SOUR BUS' will be transmitted to instrument A7 by GPIB-CONTROLLER.
3. The instruction of 'INIT' will be transmitted to instrument A7 by GPIB-CONTROLLER.
4. The Message of IEEE488.1-GET will be transmitted to instrument A7 by GPIB-CONTROLLER.
5. GPIB-CONTROLLER will read the data in output-buffer of instrument A7 and transmite the data back to PC.

Exam:SEND 9 'DISP:LABL?';RdIEEEB? 9
Rp: #8000004181MHZ …
Description:
1. PC give the command
"SEND 9 'DISP:LABL?';RdIEEEB? 9" and '0d' and '0a'
to GPIB-CONTROLLER.
2. The instruction of 'DISP:LABL?' will be transmitted to instrument A9 by GPIB-CONTROLLER.
3. GPIB-CONTROLLER will read the Block-Data of IEEE488-DEFINITE-ARBITRARY-BLOCK in output-buffer of instrument A9 and transmite the Block-Data back to PC.

Exam:SEND 7 '*idn?'; SEND 7'MEAS:AC?';READ? 7
Rp: Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09+1.32408543E-03
Description:
1. PC give the command
"SEND 7 '*idn?';SEND 7'MEAS:AC?';READ? 7 "
and '0d' and '0a'
to GPIB-CONTROLLER.
2. The instruction of '*IDN?;' will be transmitted to instrument A7 by GPIB-CONTROLLER.
3. The instruction of 'MEAS:AC?' will be transmitted to instrument A7 by GPIB-CONTROLLER.
4. GPIB-CONTROLLER will read the data in output-buffer of instrument A7 and transmite the data back to PC.

5.4.6 Instructions-Write

(a) **Block-Data** in **dS0..dSn** or **dStr** or **iaB** sent to **An** or **AAn** through GPIB-CONTROLLER
(b) If the length of **Block-Data** is more than 228 bytes, **Key-Symbol** **'|'** can be used for **Block-Data** which is divided into many parts for sending them with several times
(c) The function of **Key-Symbol** **`^`** please refer to item **5-4-4 (e)**
(d) if **dStr** are used in the instruction given to GPIB-CONTROLLER from **PC** but **Key-Symbol** **`@`** are not used, the contents of the first **dStr** in instruction only would be sent to **AAn** through GPIB-CONTROLLER
(e) if **dStr** are not used in the instruction given to GPIB-CONTROLLER from **PC** but **Key-Symbol** **`@`** are used, the contents of **IDAT** would be sent to **AAn** through GPIB-CONTROLLER.

Separate description of instructions:
Various format of **Block-Data** sent to **AAn** from **PC** through **K-PRODUC**:
(a) **IEEEBW**: IEEE488-DEFINITE-LENGTH-ARBITRARY-BLOCK
(b) **AgiETBW**: Agilent-8753ET-Form1,2,3,5

(c) **ARBITRARYW**: Data of any format

Syntax:

Key-Word	Key-Symbol	Param		
AgiETBW	|,<,>,%,-,@,?,^,!,#	AAn	dStr	DlyW
IEEEBW	|,<,>,%,-,@,?,^,!,#	AAn	dStr	DlyW
ARBITRARYW	|,<,>,%,-,@,?,^,!,#	AAn	dStr	DlyW

Command	::= 'IEEEBW' Key-Symbol-IEEEBW An DS0ToDSnW
Key-Symbol-IEEEBW	::= '|' | < | > | - | @ | ? | ˆ | ! | #
Command	::= 'ARBITRARYW' Key-Symbol-ARBITRARYW BLEN An DS0ToDSnW

Description:

BLEN is the data byte number to be transmitted to PC.

Key-Symbol-ARBITRARYW	::= '|' | < | > | - | @ | ? | ˆ | ! | #
Command	::= 'AgiETBW' Key-Symbol-AgiETBW An DS0ToDSnW
Key-Symbol-AgiETBW	::= '|' | < | > | - | @ | ? | ˆ | ! | #明
AAn	::= An AAn | Lamda
DS0ToDSnW	::= DS0ToDSnW_R DlyW | ',' DlyW | Lamda
DS0ToDSn_R	::= dStr DS0ToDSn_R | Lamda

Exam: AgiETBW^ 9 '4D4541533A41433F'

Rp: 1.17373430E-06 5.4.7

Instructions--IEEE488-Message

Separate Description of instructions:

MsgREN

IEEE488.1-REN-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER.

MsgREN+

IEEE488.1-LLO-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER

MsgREN-

IEEE488.1-GTL-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER

MsgTRG

IEEE488.1-GET-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER

MsgSDC

IEEE488.1-SDC-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER

MsgDC

IEEE488.1-DC-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER

MsgIFC

IEEE488.1-IFC-MESSAGE sent to **AAn** from **PC** through GPIB-CONTROLLER

RESET

PC give the instruction to GPIB-CONTROLLER and indicate that the **IEEE488** message used for the regulation of reset-protocol In **IEEE488.2** will be sent to instrument by GPIB-CONTROLLER.
such as:

***RST, IFC, DCL, FINDLISTEN** etc.

Syntax:

Key-Word	Key-Symbol	Parameter			
MsgREN	-,+	AAn	,	DlyW	DlyB
MsgTRG		AAn	,	DlyW	DlyB
MsgSDC		AAn	,	DlyW	DlyB
MsgDC				DlyW	DlyB
MsgIFC				DlyW	DlyB
RESET				DlyW	DlyB

Example of various IEEE488.1 Message sent to instrument from PC through GPIB-CONTROLLER **as below:**

Exam: MsgDC.

Description:
  PC send IEEE488.1-DC to AAn through GPIB-CONTROLLER

Exam: MsgIFC
 Description:
  PC send IEEE488.1-IFC to AAn through GPIB-CONTROLLER

Exam: MsgSDC 7 4 9
 Description:
  PC send IEEE488.1-SDC to A7,A4 and A9 through GPIB-CONTROLLER

Exam: MsgREN- 7 4 9
 Description:
  PC send IEEE488.1-GTL to A7,A4 and A9 through GPIB-CONTROLLER

Exam: MsgREN 9 7 4
 Description:
  PC send IEEE488.1-REN to A9,A7 and A4 through GPIB-CONTROLLER

Exam: MsgREN+
 Description:
  PC send IEEE488.1-LLO to AAn through GPIB-CONTROLLER

Exam: MsgREN
 Description:
  PC send IEEE488.1-REN to AAn through GPIB-CONTROLLER

Exam: MsgTRG 9 7
 Description:
  PC send IEEE488.1-GET to A9 and A7 through GPIB-CONTROLLER

Exam:SEND 7 9 'CONF:VOLT:DC''TRIG:SOUR BUS''INIT'; MsgTRG 7 9;
    Send? 7 9'FETC?'

Rp: +1.84852040E-04;+1.84804040E-04

Description:

a. Instrument A7 and A9 are AGILENT-34410A
b. Instrument A7 and A9 are configured as state of DC-VOLTAGE-MEASUREMENT by the instrction of 'CONF:VOLT:DC'
c. The TRIGGER-SOURCE of Instrument A7 and A9 are configured as the signal of GPIB-BUS.
d. Instrument A7 and A9 are configured as State of waiting for trig by the instrction of 'INIT'
e. GPIB-CONTROLLER will transmite signal of trig to instrument A7 and A9 at the same time when receiving instruction of 'MsgTRG 7 11'from PC.
   Once A7 receive the signal of trig, Analog-to-Digital-Converting of A7 will be executed, and data obtained from Analog-to-Digital-Converting will be saved to the output buffer of A7.
   Once A9 receive the signal of trig, Analog-to-Digital-Converting of A9 will be executed,and data obtained from Analog-to-Digital-Converting will be saved to the output buffer of A9.
f. The instruction of "Send? 7 11'FETC?'" will make GPIB-CONTROLLER do actions:
   1. Read data from the output buffer of A7 and send them back to PC
   2. Read data from the output buffer of A9 and send them back to PC

5.4.8 Instruction--Internal-Array-Buf-IO

AryAdd Contents of **dStr** will be added to the end of **iaB** address.

ArySet **(a)** Instruction without **'@'**: Contents of **iaB** will be deleted and replaced by **Block-Data** in **dStr**.

(b) Instruction with **'@'**: **Block-Data** in **dStr** is Intel-Hex Format, The **data in iaB** would be replaced by the data in **dStr** if there are same address of the data in **dStr** and **in iaB**

Syntax:

Key-Word	Key-Symbol	Parameter

AryAdd	!,_,?,$,^ ('^' used, **HexToBin** will be execute)	dStr
ArySet	!,_,?,$,^	dStr,

Exam: ArySet? 'abc… etc.'
 Rp: abc… etc.
  Description:
    PC give the instruction with 'abc… etc.' to GPIB-CONTROLLER,
    `abc… etc.` " will be set-up in the iaB of GPIB-CONTROLLER,
    (the dStr contents of instruction) at first, and then contents
    of iaB will be transmitted to PC by GPIB-CONTROLLER

Exam: AryAdd? '12345678'
 Rp: abcdefg12345678
  Description:
   Instruction with '?', contents of iaB will be transmitted
   back to PC by GPIB-CONTROLLER.

Exam: ArySet_
 Rp: 15
  Description:
   Instruction with '_', the number of data bytes in iaB will be
   transmitted back to PC by GPIB-CONTROLLER.

Exam: ArySet!
 Description:
  Instruction with '!', the Contents of iaB in GPIB-CONTROLLER
  will be deleted.

Exam: ArySet^? '4D4541533A41433F4D4541533A41433F4D4541533A41433F4D4541533A41433F4D4541533A41433F'
  Rp: MEAS:AC?MEAS:AC?MEAS:AC?MEAS:AC?MEAS:AC?

Exam: ArySet@ ':080000003132333435363738 54';ArySet?;ArySet_
  Rp: 12345678;8

5.4.9 Instruction--Default-Terminator-Configuration

SetPMT

(1) default value of Program-Message-Terminator**(**PMT**)** configured
(2) default value of Program-Message-Terminator(PMT) queried
(3) default value of Program-Message-Terminator(PMT) both
configured and queried

SetRDT

(1) default value of Response Data Terminator(RDT)configured
(2) default value of Response Data Terminator(RDT)queried
(3) default value of Response Data Terminator(RDT)both
configured and queried

Syntax:

Key-Word	Key-Symbol	Parameter	Parameter
SetPMT	?	EOC	dStr
SetRDT	?	EOC	dStr

Exam: SetPMT '0d0a'
 Description:
   PC configure the PMT default value of GPIB-CONTROLLER to '0d0a'

Exam: SetPMT? '0a'
 Rp:0A
  Description:
   1. PC configure the PMT default value of GPIB-CONTROLLER to '0a'
   2. The PMT default value of GPIB-CONTROLLER is read back by PC.

Exam: SetPMT
  Description:
   PC configure the PMT default value of GPIB-CONTROLLER

to DABE (DABE is default Power-On-PMT value) by PC.

Exam: SetPMT ?

Rp:

Description:

The response to PC from GPIB-CONTROLLER is null because PMT is DABE which is default PMT value for powered-On of GPIB-CONTROLLER.

Exam: SetRDT '0a'

Description:

PC configure the RDT default value of GPIB-CONTROLLER to '0a'. **Exam:**SetRDT? '0d'

Exam: SetRDT? '0d0a'

Rp:0D 0A

Description:

1. PC configure the RDT as the default RDT value for GPIB-CONTROLLER 0d+0a.
2. The RDT default value of GPIB-CONTROLLER is read back by PC.

Exam: SetRDT

Description:

1. PC configure the RDT as the default RDT value for GPIB-CONTROLLER 0d+0a.

Exam: SetRDT ?

Rp:0D 0A

Description:

PC read back the RDT value of GPIB-CONTROLLER.

Exam: SetRDT? 22

Rp:16

Description:

1. PC configure the RDT value of GPIB-CONTROLLER as 22+DABE.
2. The PMT value of GPIB-CONTROLLER is read back by PC. (the 16 is Hex format of the 22)

5.4.10 Instruction --GpIO-Board-Configuration

PIO

1. GpIO-Board-Port-Output is configured.
2. GpIO-Board-Port-Output will be read and transmitted to PC.

Syntax:

Key-Word	KeySymbol	Paramter	Parameter
PIO	?,#,+	AAn	dStr

Description:

INSTRUCTIONS	ACTIONS
PIO+ 1230801	The GpIO-Board address primary:1 secondary:8 Output value will be configured to 123.
PIO? 801	The GpIO-Board address primary:1 secondary:8 Output value will be read and transmitted Back to PC
PIO?+ 210801	The GpIO-Board address primary:1 secondary:8 Output value will be configured as 21 first and then read and then transmitted back to PC
PIO# 801	The output value of GpIO-Board address primary:1 secondary:8 will be read and converted to Hex format and then transmitted back to PC

PIO 401 601 301 '3F27F2'	GpIO-Board address primary:1 secondary:4 will be configured to 3F GpIO-Board address primary:1 secondary:6 will be configured to 27 GpIO-Board address primary:1 secondary:3 will be configured to F2

Ps: GpIO-Board :
Such as: KI-GRUA01 ….

Exam:FINDLISTEN?
Rp: 24 25 26 27
Description:
The port GPIB-ADDRESS of GpIO-Board are 24, 25, 26 and 27.

Exam:PIO 26 27 '3AFF'
Description:
Configuring the I/O port of address 26 to be 3A (Hex) and address 27 to be FF (Hex).

Exam:PIO? 26 27
Rp: 58 255
Description:
The input level of I/O port for address 26 and 27 are '58' and '255'.

6.The KI-GB1201X,KI-GB1201R,KI-GC3201,KI-GC3201 and KI-GX3201 Extend-Function description.

6.1 To setup the baudrate of RS232 interface

a.The GPIB address for the setting of baudrate:
Primary : 30 and Secondary : 30

b.Instruction for setting baudrate :
baudrate=Baudrate-ID

Baudrate-ID	Baudrate	
0	2400	
1	9600	(default situation)
2	14400	
3	19200	
4	28800	
5	38400	
6	57600	
7	115200	
8	128000	

Exam:send 3030 'baudrate=2'
Rp:
Description:
The baudrate of RS232 interface will be configured as 14400.

Exam:send? 3030 'baudrate=4'
Rp: BAUDRATE:28800;
Description:
The baudrate of RS232 interface will be configured as 28800, and transmitted back to PC by GPIB-CONTROLLER.

Note:
After the baudreate has been set, the baudreate value will be saved in the fresh-ROM of CPU-8051, and it will not be destroyed when the power of CPU-8051 is off. As CPU-8051 is initialed,the value of baudrate in Fresh-ROM will be read and then based for the setting of the CPU-8051 baudrate.

c. Instruction for the querying of baudrate :
   Baudrate?
   Exam:send? 3030
    Rp: BAUDRATE:28800;
    Description:
     The baudrate value, 2880, of RS232 interface will be transmitted
     back to PC by GPIB-CONTROLLER.

6.2 Description of Data-Flow for the RS232 interface

Default-Data-Flow-Type:

**The data transmitted from the instrument to GPIB-CONTROLLER
through Rs232-Port :

a. Block-Data: The end of Block-Data must be with 0a, and.
   data must not exceed 748 bytes.

b. If IO-Buffer-Status of the GPIP-CONTROLLER is Ready-and-Lock-State,
   the GPIP-CONTROLLER will give up the data input from the Rs232-Port.
   However,if the IO-Buffer-Status of the GPIP-CONTROLLER is Free-State,
   the GPIP-CONTROLLER will save the data in IO-Buffer until the input
   data is \`0a\`. Then the IO-Buffer-Status will be set as the state of
   Ready-and-Lock-State .

c. PC read data form the IO-Buffer of GPIP-CONTROLLER.
   1. If IO-Buffer-Status is not Ready-and-Lock-State,
      GPIP-CONTROLLER will not transmite data to PC.
   2. If IO-Buffer-Status is Ready-and-Lock-State,GPIP-CONTROLLER
      will transmite all the data in IO-Buffer and \`0d\` and \`0a\`to
      PC.
   3. if all of the data in IO-Buffer are already read by PC, the
      IO-Buffer will be cleared and IO-Buffer-Status will be set
      as Free-State.

**The data transmitted from GPIB-CONTROLLER to the instrument
through Rs232-Port :

a.If data is not with the signal of EOI:
  GPIB-CONTROLLER will transmite the data to instrument through
  Rs232-Port.

b.The number of data are not limited.

c.If data is with the signal of EOI:
  GPIB-CONTROLLER will transmite the data \`0d\` and \`0a\`to
  instrument through Rs232-Port.
   Exam:SENDI 230 '*id'
    Description:
     It is not with the signal of EOI.
   Exam:SEND? 230 'n?'
    Rp: Rohde&Schwarz,SME03,835328/017,4.11
    Description:
     It is with the signal of EOI.

Code-Reader-Data-Flow-Type

**The data transmitted from Barcode-Reader to GPIB-CONTROLLER through Rs232-Port :

a. Block-Data:
   The end of Block-Data must be with 0a, and number of data must not exceed 748 bytes.

b. If IO-Buffer-Status of the GPIP-CONTROLLER is Ready-and-Lock-State, the GPIP-CONTROLLER will give up the data input from the Rs232-Port. However,if the IO-Buffer-Status of the GPIP-CONTROLLER is Free-State, the GPIP-CONTROLLER will save the data in IO-Buffer until the input data is `0a`. Then the IO-Buffer-Status will be set as the state of Ready-and-Lock-State .

c. PC read data form the IO-Buffer of GPIP-CONTROLLER.
   1. If IO-Buffer-Status is not Ready-and-Lock-State, GPIP-CONTROLLER will transmite `0d` and `0a` to PC.
   2. If IO-Buffer-Status is Ready-and-Lock-State,GPIP-CONTROLLER will transmite `0` and `0d` and `0a` and `0` and `0d` and `0a` and all the data in IO-Buffer and `0d` and `0a` to PC.
   3. if all of the data in IO-Buffer are already read by PC, the IO-Buffer will be cleared and IO-Buffer-Status will be set as Free-State.

**The data transmitted from GPIB-CONTROLLER to the Barcode-Reader through Rs232-Port :

The relation about EOI and Rs232-Port output data `0d` and `0a`

a.If data is not with the signal of EOI:
  GPIB-CONTROLLER will transmite the data to Barcode-Reader through Rs232-Port.

b.The number of data are not limitted.

c.If data is with the signal of EOI:
  GPIB-CONTROLLER will transmite the data `0d` and `0a`to Barcode-Reader through Rs232-Port.

  example:
```
  SENDI 230 ‘*id’
```
  Description:
  It is not with the signal of EOI.

  example:
```
  SEND? 230 ‘n?’
Rp: Rohde&Schwarz,SME03,835328/017,4.11
```
  Description:
  It is with the signal of EOI.

6.2.1 The configuration for Data-Flow Type

a.GPIB address for configuring the Data-Flow Type :
  Primary : 30 and Secondary : 30

b.Instruction for configuring the Data-Flow :
  input=0 or 1
  Exam:SEND? 3030 `input=0`
  Description :
  Configure the Data-Flow Type for RS232 interface as

Default-Data-Flow-Type.

Exam:send? 3030 'input=1'

Description :

Configure the Data-Flow Type for RS232 interface as Code-Reader-Data-Flow-Type.

6.3 The reading and writing for the interface of RS232

Exam: Send? 230 '*IDN?'

Rp: Rohde&Schwarz,SME03,833777/013,4.11

Description:

1. PC give the command "Send? 230 '*IDN?'" and '0d' and '0a' to GPIB-CONTROLLER.
2. The instruction of '*IDN?' will be transmitted to the instrument address Primary:30 and Secondary:2 by GPIB-CONTROLLER.
3. GPIB-CONTROLLER will read the data in the output-buffer of instrument address Primary:30 and Secondary:2 and stop reading until EOI signal is received.
4. GPIB-CONTROLLER will delete the EOS (End of String) added with the data obtained from item-3.
5. The data left from item 4 will be added with 0d+0a at the rear and then transmitted back to PC by GPIB-CONTROLLER.

6.4 The reading and writing for the interface of Digital I/O port

Exam:PIO 130 430 '3AFF'

Description:

1 PC give the command "PIO 130 430 '3AFF'" and '0d' and '0a' to GPIB-CONTROLLER

2 GPIB-CONTROLLER will configure the output port address Primary:30 and Secondary:1 as 3A (Hex)

3 GPIB-CONTROLLER will configure the output port with Address Primary:30 and Secondary:4 as FF (Hex)

Exam:PIO? 130

Rp: 58

Description:

1 PC give the command "PIO? 130" and '0d' and '0a' to GPIB-CONTROLLER

2. GPIB-CONTROLLER will transmite the output value '58'of the port with address Primary:30 and Secondary:1 back to PC

Exam:PIO# 130

Rp: 3A

Description:

1 PC give the command "PIO# 130" and '0d' and '0a' to GPIB-CONTROLLER

2. GPIB-CONTROLLER will read the output value of the port with address Primary:30 and Secondary:1
3. GPIB-CONTROLLER transform the value obtained from Item2 as the HEX-format 3A and transmite it back to PC

6.5 To control the digital input RI and DSR and output RTS and DTR

Exam:

```
EscapeCommFunction(hComm, SETRTS);
EscapeCommFunction(hComm, CLRRTS);
Value-buffer = GetCommModemStatus(hComm, MS_RING_ON);
Value-buffer = GetCommModemStatus(hComm, MS_DSR_ON);
```

6.6 Digital I/O port address Primary:30 Secondary:5

Exam:PIO 530 '3A'

Description:

1 PC give the command
 "PIO 530 '3A'" and '0d' and '0a'
 to GPIB-CONTROLLER
2. GPIB-CONTROLLER will configure the output value of the I/O port with address Primary:30 and Secondary:5 as '3A'(HEX)

Exam:PIO? 530

Rp: 37

Description:

1 PC give the command
 "PIO? 530" and '0d' and '0a'
 to GPIB-CONTROLLER
2. GPIB-CONTROLLER will read the I/O port output value '37' for the address of Primary:30 and Secondary:5 and transmite it back to PC.

GPIB UTILITY of GPIB-CONTROLLER with HP-6623A

```
001 -> findlisten? 5
    <- 05
002 -> PP# 5 'SRQ 2''11'50
    <- 01
003 -> PP# 5 'SRQ 2''12'50
    <- 02
004 -> PP# 5 'SRQ 2''13'50
    <- 04
005 -> PP# 5 'SRQ 2''14'50
    <- 08
006 -> PP# 5 'SRQ 2''15'50
    <- 10
007 -> PP# 5 'SRQ 2''16'50
    <- 20
008 -> PP# 5 'SRQ 2''17'50
    <- 40
009 -> PP# 5 'SRQ 2''18'50
    <- 80
010 -> SEND? 5 'PON 1''DSP 1''UNMASK 1,255''OCP 1,1''OVRST 1''SRQ 1''ID?'
    <- HP6623A
011 -> SEND? 5 'TEST?'
    <-   0
012 -> SEND= 5 'VSET? 1''ISET? 1''VOUT? 1''OCP? 1''OUT? 1''UNMASK? 1''DLY? 1'
    <-   5.002;  0.082;  5.004; -0.001;   5.01;  1;  1;255;  0.020
013 -> SEND= 5 'STS? 1''ASTS? 1''FAULT? 1''ERR?''SRQ?''PON?''DSP?''CMODE?'
    <-   1;  1;  1;  6;  1;  1;  1;  0
014 -> SEND= 5 'VSET 1,3''VSET? 1''VOUT? 1'
    <-   2.999;  3.001
015 -> SEND? 5 'VSET 1,4; ISET 1,2''OUT? 1'
    <-   1
016 -> SEND? 5 '"OVSET 1,3.5''OUT 1,1''OVRST 1''OUT? 1'
    <-   1
017 -> SEND? 5 'OCP 2,1''OCP? 2'
    <-   1
018 -> SEND5 'ISET 1,1''ISET 2,1.5''ISET 3,0.01'
    <-
019 -> SEND5 'VSET 1,10''VSET 2,7''VSET 3,1'
    <-
020 -> SEND? 5 'OCRST 2''STS? 2'
    <-   1
021 -> SEND? 5 'ASTS? 2'
    <-   1
022 -> SEND? 5 'UNMASK? 2'
    <-   0
023 -> SEND5 'VSET 2,6''VSET 1,3''VSET 3,6'
    <-
024 -> SEND? 5 'FAULT? 2'
```

```
    <-   0
025 -> SEND? 5 'SRQ 1''PON 1''PON?'
    <-   1
026 -> SEND5 'VSET 1,3''VSET 2,5''VSET 3,9'
    <-
027 -> SEND 5 'DSP \"OUTPUT 2 OK\"'
    <-
028 -> SEND? 5 'DLY 2,.08''SRQ 1''ERR?'
    <-   6
029 -> SPOLL? 5
    <- 113
030 -> SEND? 5 '"OVSET 2,4.5''OUT 2,1''OVRST 2''OUT? 2'
    <-   1
031 -> SEND5 'VSET 1,20''VSET 1,5'
    <-
032 -> SEND5 'VSET 1,5''VSET 1,3''VSET 2,4''VSET 3,15'
    <-
033 -> SEND5 'CLR'
    <-
034 -> SEND? 5 'OCP 3,1''OCP? 3'
    <-   0
035 -> SEND5 'VSET 1,12''VSET 2,5''VSET 3,4'
    <-
036 -> SEND? 5 'OCRST 3''STS? 3'
    <-   1
037 -> SEND? 5 'ASTS? 3''VSET 1,5'
    <-   1
```

GPIB UTILITY of GPIB-CONTROLLER with Agilent-34410A

```
001 -> Findlisten? 7
    <- 07
002 -> Send? 7 '*SRE?'
    <- +32
003 -> Send? 7 '*STB?'
    <- +0
004 -> Send? 7 '*IDN?'
    <- Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
005 -> Send 7 '*CLS'
    <-
006 -> Send? 7 '*IDN?'
    <- Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
007 -> SEND? 7 '*ESE?'
    <- +64
008 -> Send? 7 '*STB?'
    <- +0
009 -> SEND? 7 ':CONF:VOLT:DC 10,0.1;'':SAMP:COUN 4'':READ?'30
    <- -9.91556517E-03,-8.74864364E-03,-7.83706910E-03,-6.91345844E-03
010 -> SEND? 7 '*ESR?'
    <- +0
011 -> Send 7 '*PSC 1'
    <-
012 -> Send? 7 'MEAS:VOLT:DC?'
    <- +1.50896586E-04
013 -> Send? 7 '*PSC?'
    <- 1
014 -> Send? 7 'MEAS:VOLT:DC?'
    <- +1.48671368E-04
015 -> Send 7 '*ESE 33'
    <-
016 -> Send? 7 '*ESE?'
    <- +33
017 -> SEND? 7 '*PSC?'
    <- 1
018 -> Send? 7 '*SRE?'
    <- +32
019 -> Send 7 '*ESE 33'
    <-
020 -> Send? 7 'MEAS:VOLT:DC?'
    <- +1.07699886E-04
021 -> Send 7 '*ESE 96'
    <-
022 -> Send? 7 '*ESR?'
    <- +0
023 -> Send? 7 '*IDN?'50
    <- Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
024 -> Send? 7 'MEAS:VOLT?;'50 50
```

```
    <- +1.27973527E-04
025 -> Send 7 '*SRE 96'
    <-
026 -> Send 7 '*ESE 64'
    <-
027 -> Send? 7 '*IDN?'50 50
    <- Agilent Technologies,34410A,MY47013754,2.35-2.35-0.09-46-09
028 -> Send? 7 'MEAS:VOLT?'50 50
    <- +1.15146222E-04
```

GPIB UTILITY of GPIB-CONTROLLER with Agilent-DSO-X2012A

```
001 -> Findlisten? 9
     <- 09
 002 -> Send? 9 '*idn?'50
     <- AGILENT TECHNOLOGIES,DSO-X 2012A,MY52132806,02.10.2012022200
 003 -> SEND? 9 'EXT:BWL?'50
     <- 0
 004 -> SEND? 9 'MEAS:SHOW?'50
     <- 1
 005 -> SEND? 9 '*STB?'50
     <- +165
 006 -> SEND? 9 'ACQ:TYPE?'50
     <- NORM
 007 -> SEND? 9 'ACQ:MODE?'50
     <- RTIM
 008 -> SEND? 9 'AUT:FDEB?'50
     <- 0
 009 -> SEND? 9 'CHAN1:DISP?'50
     <- 1
 010 -> SEND? 9 'ACQ:COMP?'50
     <- 100
 011 -> SEND? 9 'CHAN1:IMP?'50
     <- ONEM
 012 -> Send? 9 '*OPT?'50
     <- 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,BW10,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 013 -> SEND? 9 '*ESE?'50
     <- +255
 014 -> SEND? 9 'MTE?'50
     <- +1795
 015 -> SEND? 9 '*OPC?'50
     <- 1
 016 -> SEND= 9 '*CLS''*OPC?'50
     <- 1
 017 -> SEND? 9 '*OPC?'50
     <- 1
 018 -> SPOLL# 9
     <-
 019 -> SEND? 9 'OPEE?'50
     <- +2216
 020 -> SEND? 9 'OVLR?'50
     <- +0
 021 -> SEND? 9 'TER?'50
     <- +1
 022 -> SEND? 9 'ACQ:POIN?'50
     <- 50000
```

GPIB UTILITY of GPIB-CONTROLLER with ROHDE&SCHWARZ-SME03

```
001 -> FINDLISTEN? 28
    <- 28
002 -> SEND? 28 'SOUR:POW:LEV -11''SOUR:POW:LEV?'1
    <- -11.00
003 -> SEND? 28 'SOUR:FREQ 11E6''SOUR:FREQ?'1
    <- 11000000.0
004 -> SEND? 28 '*IDN?'
    <- Rohde&Schwarz,SME03,833777/013,4.11
005 -> SEND= 28 '*ESE?''*ESR?''*PRE?''*SRE?''*STB?''*OPC?'
    <- 1;0;32;32;0;1
006 -> SPOLL? 28
    <- 00
007 -> SEND? 28 'SOUR:POW:LEV -15''SOUR:POW:LEV?'1
    <- -15.00
008 -> SEND? 28 'SOUR:FREQ 25E6''SOUR:FREQ?'1
    <- 25000000.0
009 -> SPOLL? 28 '*CLS''*ESE 1''*SRE 32''*OPC'
    <- 96
010 -> SPOLL? 28 '*CLS''*ESE 1''*SRE 32'
    <- 00
011 -> SPOLL? 28
    <- 00
012 -> SEND? 28 'SOUR:POW:LEV -18''SOUR:POW:LEV?'1
    <- -18.00
013 -> SEND? 28 'SOUR:FREQ 18E6''SOUR:FREQ?'1
    <- 18000000.0
014 -> SEND# 28 '*CLS''*ESE 1''*PRE 32''*SRE 32''*OPC''*ist?'
    <-
015 -> SEND# 28 '*CLS''*ESE 1''*PRE 32''*SRE 32''*ist?'
    <-
016 -> SEND? 28 'SOUR:POW:LEV -12''SOUR:POW:LEV?'1
    <- 31
017 -> SEND? 28 'SOUR:FREQ 12E6''SOUR:FREQ?'1
    <- -12.00
018 -> SEND? 28 '*IST?'
    <- 12000000.0
019 -> PP# 28 '*OPC'
    <- 1
020 -> SEND? 28 'SOUR:POW:LEV -10''SOUR:POW:LEV?'1
```

```
    <- -10.00
021 -> SEND? 28 'SOUR:FREQ 50E6''SOUR:FREQ?'1
    <- 50000000.0
022 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''11'
    <- 01
023 -> PP# 28 '*CLS''*ESE 1''*PRE 32''11'
    <- 00
024 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''01'
    <- 00
025 -> PP# 28 '*CLS''*ESE 1''*PRE 32''01'
    <- 01
026 -> SEND? 28 'SOUR:POW:LEV -10''SOUR:POW:LEV?'1
    <- -10.00
027 -> SEND? 28 'SOUR:FREQ 10E6''SOUR:FREQ?'1
    <- 10000000.0
028 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''12'
    <- 02
029 -> PP# 28 '*CLS''*ESE 1''*PRE 32''12'
    <- 00
030 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''02'
    <- 00
031 -> PP# 28 '*CLS''*ESE 1''*PRE 32''02'
    <- 02
032 -> SEND? 28 'SOUR:POW:LEV -13''SOUR:POW:LEV?'
    <- -13.00
033 -> SEND? 28 'SOUR:FREQ 13E6''SOUR:FREQ?'1
    <- 13000000.0
034 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''13'
    <- 04
035 -> PP# 28 '*CLS''*ESE 1''*PRE 32''13'
    <- 00
036 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''03'
    <- 00
037 -> PP# 28 '*CLS''*ESE 1''*PRE 32''03'
    <- 04
038 -> SEND? 28 'SOUR:POW:LEV -16''SOUR:POW:LEV?'1
    <- -16.00
039 -> SEND? 28 'SOUR:FREQ 16E6''SOUR:FREQ?'1
    <- 16000000.0
040 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''14'
    <- 08
```

```
041 -> PP# 28 '*CLS''*ESE 1''*PRE 32''14'
    <- 00
042 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''04'
    <- 00
043 -> PP# 28 '*CLS''*ESE 1''*PRE 32''04'
    <- 08
044 -> SEND? 28 'SOUR:POW:LEV -19''SOUR:POW:LEV?'1
    <- -19.00
045 -> SEND? 28 'SOUR:FREQ 19E6''SOUR:FREQ?'1
    <- 19000000.0
046 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''15'
    <- 10
047 -> PP# 28 '*CLS''*ESE 1''*PRE 32''15'
    <- 00
048 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''05'
    <- 00
049 -> PP# 28 '*CLS''*ESE 1''*PRE 32''05'
    <- 10
050 -> SEND? 28 'SOUR:POW:LEV -22''SOUR:POW:LEV?'1
    <- -22.00
051 -> SEND? 28 'SOUR:FREQ 22E6''SOUR:FREQ?'1
    <- 22000000.0
052 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''16'
    <- 20
053 -> PP# 28 '*CLS''*ESE 1''*PRE 32''16'
    <- 00
054 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''06'
    <- 00
055 -> PP# 28 '*CLS''*ESE 1''*PRE 32''06'
    <- 20
056 -> SEND? 28 'SOUR:POW:LEV -26''SOUR:POW:LEV?'1
    <- -26.00
057 -> SEND? 28 'SOUR:FREQ 26E6''SOUR:FREQ?'1
    <- 26000000.0
058 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''17'
    <- 40
059 -> PP# 28 '*CLS''*ESE 1''*PRE 32''17'
    <- 00
060 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''07'
    <- 00
061 -> PP# 28 '*CLS''*ESE 1''*PRE 32''07'
```

```
    <- 40
062 -> SEND? 28 'SOUR:POW:LEV -29''SOUR:POW:LEV?'1
    <- -29.00
063 -> SEND? 28 'SOUR:FREQ 29E6''SOUR:FREQ?'1
    <- 29000000.0
064 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''18'
    <- 80
065 -> PP# 28 '*CLS''*ESE 1''*PRE 32''18'
    <- 00
066 -> PP# 28 '*CLS''*ESE 1''*PRE 32''*OPC''08'
    <- 00
067 -> PP# 28 '*CLS''*ESE 1''*PRE 32''08'
    <- 80
068 -> SEND? 28 'SOUR:POW:LEV -32''SOUR:POW:LEV?'1
    <- -32.00
069 -> SEND? 28 'SOUR:FREQ 32E6''SOUR:FREQ?'1
    <- 32000000.0
070 -> PP- 28
    <-
071 -> SEND? 28 '*CLS''*ESE 1''*PRE 32''*SRE 32''*OPC?'
    <- 1
072 -> SEND? 28 'SOUR:FREQ 200E6''SOUR:FREQ?'
    <- 200000000.0
073 -> SEND? 28 'SOUR:POW:LEV -30''SOUR:POW:LEV?'1
    <- -30.00
074 -> SEND? 28 'SOUR:FREQ:OFFS?'1
    <- 0
075 -> SEND? 28 'SOUR:POW:LEV -77''SOUR:POW:LEV?'1
    <- -77.00
076 -> SEND? 28 'SOUR:FREQ 77E6''SOUR:FREQ?'1
    <- 77000000.0
077 -> SEND? 28 'SOUR:FREQ:STEP?'1
    <- 1000000.0
078 -> SEND? 28 'SOUR:POW:LEV -25''SOUR:POW:LEV?'1
    <- -25.00
079 -> SEND? 28 'SOUR:FREQ 175E6''SOUR:FREQ?'1
    <- 175000000.0
080 -> SEND? 28 'SOUR:POW:LIM?'1
    <- 16.00
081 -> SEND? 28 'SOUR:POW:LEV -88''SOUR:POW:LEV?'1
    <- -88.00
```

```
082 -> SEND? 28 'SOUR:FREQ 88E6''SOUR:FREQ?'1
    <- 88000000.0
083 -> SEND? 28 'SOUR:POW:ALC:BAND:AUTO?'1
    <- ON
084 -> SEND? 28 'SOUR:POW:LEV -66''SOUR:POW:LEV?'1
    <- -66.00
085 -> SEND? 28 'SOUR:FREQ 66E6''SOUR:FREQ?'1
    <- 66000000.0
086 -> SEND? 28 'SOUR:POW:ALC:BAND?'1
    <- 100000
087 -> SEND? 28 'SOUR:CORR?'1
    <- 0
088 -> SEND? 28 'SOUR:POW:LEV -44''SOUR:POW:LEV?'1
    <- -44.00
089 -> SEND? 28 'SOUR:FREQ 44E6''SOUR:FREQ?'1
    <- 44000000.0
090 -> SEND? 28 'OUTP:AMOD AUTO''SOUR:POW:STEP 1''*OPC?'1
    <- 1
091 -> SEND? 28 'SOUR:POW:LEV -55''SOUR:POW:LEV?'1
    <- -55.00
092 -> SEND? 28 'SOUR:FREQ 55E6''SOUR:FREQ?'1
    <- 55000000.0
093 -> SEND? 28 'SOUR:FREQ 99E6''SOUR:FREQ?'1
    <- 99000000.0
094 -> SEND? 28 ':SOUR:AM:EXT:COUP AC''*OPC?'1
    <- 1
095 -> SEND? 28 'SOUR:FREQ 100E6''SOUR:FREQ?'1
    <- 100000000.0
096 -> SEND? 28 'SOUR:POW:LEV -20''SOUR:POW:LEV?'1
    <- -20.00
097 -> SEND? 28 'SOUR:FREQ 200E6''SOUR:POW:LEV -10''SOUR:FREQ?'1
  <- 200000000.0
```

GPIB UTILITY of GPIB-CONTROLLER with KI-GRUA01-R2

```
001 -> Findlisten?
    <- 24,25,26,27
002 -> Findlisten# 24 25 26 27
    <- ZZ18ZZ19ZZ1AZZ1B
003 -> PIO? 26 27 'FF00'
    <- 255;00
004 -> SEND? 24 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
005 -> SEND? 25 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
006 -> SEND? 26 'FF'
    <- F
007 -> SEND? 27 '00'
    <- 0
008 -> SEND 25 '1234567890'
    <-
009 -> SPOLL? 25
    <- 80
010 -> SEND? 25
    <- 1234567890
011 -> SEND 24 '1234567890'
    <-
012 -> SPOLL? 24
    <- 80
013 -> SEND? 24
    <- 1234567890
014 -> SEND? 24 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
015 -> SEND 25 '1234567890'
    <-
016 -> SPOLL? 25
    <- 80
017 -> SEND? 25
    <- 1234567890
018 -> PIO? 27 26 '00FF'
    <- 00;255
019 -> SEND? 24 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
020 -> SEND? 25 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
021 -> SEND 25 '1234567890'
    <-
022 -> SPOLL? 25
    <- 80
023 -> SEND? 25
    <- 1234567890
024 -> SEND 24 '1234567890'
```

```
    <-
025 -> SPOLL? 24;
    <- 80
026 -> SEND? 24
    <- 1234567890
027 -> SEND? 24 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
028 -> SEND 24 '1234567890'
    <-
029 -> SPOLL? 24
    <- 80
030 -> SEND? 24
    <- 1234567890
```

GPIB UTILITY of GPIB-CONTROLLER with KI-GRUA01-R4

```
001 -> Findlisten? 2 3 4 5 6 7
    <- 02,03,04,05,06,07
002 -> Findlisten# 2 3 4 5 6 7
    <- ZZ02ZZ03ZZ04ZZ05ZZ06ZZ07
003 -> SEND? 4 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
004 -> SEND? 5 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
005 -> SEND? 6 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
006 -> SEND? 7 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
007 -> SEND? 5 '1234567890'
    <- 1234567890
008 -> SEND 6 '1234567890';SPOLL? 6;SEND? 6
    <- 80;1234567890
009 -> SEND? 6 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
010 -> SEND 7 '1234567890';SPOLL? 7;SEND? 7
    <- 80;1234567890
011 -> Findlisten# 4 5 6 7
    <- ZZ04ZZ05ZZ06ZZ07
012 -> SEND? 4 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
013 -> SEND? 5 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
014 -> SEND 5 '1234567890';SPOLL? 5;SEND? 5
    <- 80;1234567890
015 -> SEND 4 '1234567890';SPOLL? 4;SEND? 4
    <- 80;1234567890
016 -> SEND? 4 'ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890'
    <- ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
017 -> SEND? 4 '1234567890'
    <- 1234567890
018 -> PIO? 2 3 'FF00'
    <- 255;00
019 -> PIO? 2 3 '00FF'
    <- 00;255
```

GPIB UTILITY of GPIB-CONTROLLER with HP-6611C

```
001 -> Findlisten? 4
    <- 04
002 -> send? 4 '*idn?'
    <- HEWLETT-PACKARD,6611C,0,A.01.03
003 -> SEND= 4 'outp on''volt 4''curr 0.1''curr:prot:stat on''meas:volt?'
    <- 3.99837E+0
004 -> SPOLL? 4
    <- 00
005 -> SEND= 4 '*cls''meas:volt?'
    <- 3.99846E+0
006 -> send= 4 'curr 0.1''volt 5''prot:stat on''outp on''meas:volt?'
    <- 4.99865E+0
007 -> SEND 4 '*IDN?';READ? 4
    <- HEWLETT-PACKARD,6611C,0,A.01.03
008 -> SEND? 4 '*cls''meas:volt?'
    <- 4.99868E+0
009 -> send= 4 'outp on''volt 4''curr 0.1''curr:prot:stat on''meas:curr?'
    <- -1.41440E-4
010 -> send= 4 '*cls''meas:volt?';send? 4 'meas:curr?';
    <- 3.99847E+0;-1.40313E-4
011 -> send? 4 'volt?'
    <- 4.00000E+0
012 -> SPOLL# 4
    <- 00
013 -> Send= 4 '*IDN?'
    <- HEWLETT-PACKARD,6611C,0,A.01.03
014 -> Send? 4 '*IDN?'
    <- HEWLETT-PACKARD,6611C,0,A.01.03
015 -> Send 4 '*idn?';SEND? 4
    <- HEWLETT-PACKARD,6611C,0,A.01.03
```

Sample-APP for the GPIB application of GPIB-CONTROLLER

```
/****************************************************************
 Sub-program is used for the data from the output of
 communication port, these data will be transmitted
 to instrument through GPIB-CONTROLLER,in the
 same way, the text data in communication port
 responded from instrument will be read through
 GPIB-CONTROLLER and added with characters of
 0d+0a in the rear.
 hWnd:            The handle of main-window
 WrDataStr:       Address of buffer is used to save the
                  data which will be outpput from the
                  communication port.
 Wait_TicketTime: The time setup to read the input data
                  continuously from communication port,
                  unit is 1/1000 second
                  for example:
                   3000, waiting for data over 3 second, it
                  mean that bus is error. and the procedure
                  for WriteAndReadBus will be withdrawn.
 RdDataStr:       Address of buffer is used to save the data
                  which are received from communication port.
 return:          number bytes of data received from
                  communication port.
****************************************************************/
int CALLBACK WriteAndReadBus
(
 DWORD Wait_TicketTime,HANDLE hComm,
 char *WrDataStr,char *RdDataStr
)
{
 MSG Message;int i,n,nReceive;
 char buf[1024],rbuf[1024];DWORD dwTime,nBytesRead;
 /****************************************************************
  PC transmite the contents of WrDataStr to instrumene
  34410a through GPIB-CONTROLLER
 ****************************************************************/
 nReceive=0;
 if(WrDataStr)
 {
  wsprintf(rbuf,"%s",WrDataStr);
  ::WriteFile(hComm,rbuf,strlen(rbuf),&nBytesWrite,NULL);
 }
 dwTime=GetTickCount()+Wait_TicketTime;*RdDataStr=0;
 while(1)
 {
  if(GetTickCount()>=dwTime)
  {
   if(StopTest)
 {
  StopTest=0;
  PostMessage(hWnd,WM_SYSCOMMAND,SC_CLOSE,0);
 }
 return NULL;
}
/***************************************************************
 PC will execute the work for requirement with method of
 background-processing to maintain window operation.
***************************************************************/
if(::PeekMessage(&Message,NULL,0,0,PM_REMOVE))
{
 if
 (
  (
   Message.message==WM_NCLBUTTONDOWN &&
   Message.wParam==0x14
  )||
  (
   Message.message==WM_SYSCOMMAND &&
   Message.wParam==SC_CLOSE
  )||
  (
   Message.message==WM_KEYDOWN &&
   LOWORD(Message.wParam)==0x1b
  )
 )
 {
  StopTest=1;
 }
 else
 {
  ::TranslateMessage(&Message);::DispatchMessage(&Message);
  if(haccel!=NULL)
  {
   (::TranslateAccelerator(hWnd,haccel,&Message));
  }
 }
}
/***************************************************************
 Same situation as mentioned above, PC also take the method
 of background-processing to poll the communication port,
 the data will be responded to PC from instrument through
 GPIB-CONTROLLER, PC will save these data in the address of
 RdDataStr.
***************************************************************/
if(::GetCommMask(hComm,&dwEvent))
{
 ::ClearCommError(hComm,&dwError,&comstat);
 if
```

```
    (
     ::ReadFile(hComm,rbuf,comstat.cbInQue,&nBytesRead,NULL)
     &&nBytesRead
    )
    {
     rbuf[nBytesRead]=0;n=nBytesRead;
     wsprintf(buf,"%s",rbuf);
     for(i=0;;i++)
     {
      if(i>=nBytesRead)break;
      if(buf[i]!='\r'&&buf[i]!='\n')
      {
       wsprintf
       (RdDataStr+strlen(RdDataStr),
        "%c"
        ,buf[i]
       );nReceive++;
      }
      if
      (
       i&&
       (
        buf[i-1]=='\r'&&buf[i]=='\n'
       )
      ){goto COMMANDOK;}
      else if(buf[i]=='\n')
      {
       goto COMMANDOK;
      }
     }
     dwTime=GetTickCount()+Wait_TicketTime;
    }
   }continue;
   COMMANDOK:
   if(StopTest)
   {
    StopTest=0;
    PostMessage(hWnd,WM_SYSCOMMAND,SC_CLOSE,0);
    return 0;
   }
   return nReceive;
  }
  if(StopTest)
  {
   StopTest=0;
   PostMessage(hWnd,WM_SYSCOMMAND,SC_CLOSE,0);
   return 0;
  }
  return 0;
 }
```

The application example for above sub-program:

Exam:SEND? 19 '*IDN?'
Rp: ADVANTEST,R3131,22286039,B02

```
 WriteAndReadBus
 (
  1000,hComm,
  "SEND? 19 \'*IDN?\'\r\n",
  RdDataStr
 )

  "ADVANTEST,R3131,22286039,B02"
```

The instruction of "SEND? 19 '*IDN?'" will be transmitted to instrument A19 through ki-usb/gpib-controller after sub-program execute completely, and then the response from instrument A19

```
  "ADVANTEST,R3131,22286039,B02"
```

will be read and saved in RdDataStr

Exam:SEND* 17 ':FORM:DATA ASC'':CALC1:DATA:SDAT?'
Rp: +1.00489401832E+000,+1.14720556199E-002,+1.00375867860E+00 …

```
 WriteAndReadBus
 (
  1000,hComm,
  "SEND* 17 \':FORM:DATA ASC\'\':CALC1:DATA:SDAT?\'\r\n",
  RdDataStr
 )
```

The instruction of ':FORM:DATA ASC'':CALC1:DATA:SDAT?' will be transmitted to instrument A17 through ki-usb/gpib-controller after sub-program execute completely, and then the response will be read from instrument A17

```
 "+1.00489401832E+000,+1.14720556199E-002,+1.00375867860E+00 …"
```

will be read and saved in RdDataStr

```
/*************************************************************
 The sub-program is used for transmiting BLOCK-DATA to
 instrument from communication port through
 GPIB-CONTROLLER
 DataSrc:          Address of buffer is used to save the
                   BLOCK-DATA which will be outppput from the
                   communication port.
 nBytesTransfer:  Number bytes of BLOCK-DATA for transmition
                  in DataSrc,example:
                   1000, it means that there are 1000 bytes
                   in DataSrc to transmite to instrument
                   through GPIB-CONTROLLER
 Wait_TicketTime: To setup the additional delay-time after the
                  data are output from communication port
                  each time.time unit is 1/1000 seconds, it is
                  necessary for time to match the requirement
                  for an interval of time between periods of
                  instrument receiving data from
                  GPIB-CONTROLLER each time,
                  for example: 50, it means 50 mili-seconds.
 DevAddress:      The value of instrument GPIB-ADDRESS.
 hComm:           the handle of communication port
                   Open method：
                     HANDLE hComm=::CreateFile
                     (
                      "COM3:128000,N,8,1",GENERIC_READ|GENERIC_WRITE,
                      0,NULL,
                      OPEN_EXISTING,
                      0,NULL
                     );
                   Close method:
                     CloseHandle(hfile);
*************************************************************/
void BlockDataTransfer
(
 HANDLE hComm,
 char *DataSrc, int nBytesTransfer,
 int Wait_TicketTime,
 char DevAddress
)
{
 int i;DWORD nBytesWrite,dwTime;char codebuf[256];
 HGLOBAL hglobal;LPSTR gbuf;gbuf=(LPSTR)GlobalLock(hglobal);
 memcpy(gbuf,DataSrc,nBytesTransfer);
 for(i=0;;i+=200)
 {
  if((nBytesTransfer-i)<=200)
  {
   if(!i){nBytesWrite=wsprintf(codebuf,"SEND %d",DevAddress);}
   memcpy(codebuf+nBytesWrite,"\'",1);nBytesWrite+=1;
   memcpy(codebuf+nBytesWrite,gbuf,nBytesTransfer-i);nBytesWrite+=(nBytesTransfer-i);
   memcpy(codebuf+nBytesWrite,"\'\r\n",3);nBytesWrite+=3;
  }
  else
  {
   nBytesWrite=wsprintf(codebuf,"SEND| %d",DevAddress);
   memcpy(codebuf+nBytesWrite,"\'",1);nBytesWrite+=1;
```

```
    memcpy(codebuf+nBytesWrite,gbuf,200);nBytesWrite+=200;gbuf+=200;
    memcpy(codebuf+nBytesWrite,"\'\r\n",3);nBytesWrite+=3;
   }
   WriteFile
   (hComm,
    codebuf,nBytesWrite,&nBytesWrite,NULL
   );
   if((nBytesTransfer-i)<=200){break;}
   dwTime=::GetTickCount()+Wait_TicketTime;
   while(1){if(::GetTickCount()>=dwTime)break;}
  }
  ::GlobalUnlock(hglobal);::GlobalFree(hglobal);
 }

/****************************************************************
  Sub-program is used for transforming the data in DataSrc
  into the format of IEEE488-DEFINITE-ARBITRARY-BLOCK-LENGTH
  which will be transmitted to instrument through
  GPIB-CONTROLLER
  DataSrc:         Address of buffer to save the BLOCK-DATA
                   which will be output from communication
                   port.
  nBytesTransfer:  Number bytes of the BLOCK-DATA for
                   transmition in DataSrc,
                   for example: 1000,
                    it means that there are 1000 bytes
                    in DataSrc to transmite to instrument
                    through GPIB-CONTROLLER
  Wait_TicketTime: To setup the additional delay-time after the
                   data are output from communication port
                   each time.time unit is 1/1000 seconds, it is
                   necessary for time to match the requirement
                   for an interval of time between periods of
                   instrument receiving data from
                   GPIB-CONTROLLER each time,
                   for example: 50, it means 50 mili-seconds.
  DevAddress:      The value of instrument GPIB-ADDRESS.
  hComm:           the handle of communication port
                    Open method：
                      HANDLE hComm=::CreateFile
                      (
                       "COM3:128000,N,8,1",
                       GENERIC_READ|GENERIC_WRITE,
                       0,NULL,
                       OPEN_EXISTING,
                       0,NULL
                      );
                    Close method:
                      CloseHandle(hfile);
 ****************************************************************/
```

```
void IEEEBDataTransfer
(
 HANDLE hComm,
 char *DataSrc, int nBytesTransfer,
 int Wait_TicketTime,
 char DevAddress
)
{
 HGLOBAL hglobal;LPSTR gbuf;char codebuf[256];
 int i,j,glen;DWORD nBytesWrite,dwTime;
 hglobal=GlobalAlloc(GHND,nBytesTransfer+240);
 gbuf=(LPSTR)GlobalLock(hglobal);
 j=1;while(1)
 {
  if(j>=nBytesTransfer){break;}
  j*=10;
 }
 glen=0;
 gbuf[glen++]='#';
 gbuf[glen++]=0x30+i;i=nBytesTransfer;
 while(1)
 {
  gbuf[glen++]=(i/j)+0x30;i=i%j;
  if(j==1){break;}
  else{j/=10;}
 }
 memcpy(gbuf,DataSrc,nBytesTransfer);gbuf-=glen;
 glen = nBytesTransfer + glen;
 for(i=0;;i+=200)
 {
  if((glen-i)<=200)
  {
   if(!i){nBytesWrite=wsprintf(codebuf,"SEND %d",DevAddress);}
   memcpy(codebuf+nBytesWrite,"\'",1);nBytesWrite+=1;
   memcpy(codebuf+nBytesWrite,gbuf,glen-i);nBytesWrite+=(glen-i);
   memcpy(codebuf+nBytesWrite,"\'\r\n",3);nBytesWrite+=3;
  }
  else
  {
   nBytesWrite=wsprintf(codebuf,"SENDl %d",DevAddress);
   memcpy(codebuf+nBytesWrite,"\'",1);nBytesWrite+=1;
   memcpy(codebuf+nBytesWrite,gbuf,200);nBytesWrite+=200;gbuf+=200;
   memcpy(codebuf+nBytesWrite,"\'\r\n",3);nBytesWrite+=3;
  }
  WriteFile
  (hComm,
   codebuf,nBytesWrite,&nBytesWrite,NULL
  );
  if((glen-i)<=200){break;}
  dwTime=::GetTickCount()+Wait_TicketTime;
```

```
   while(1){if(::GetTickCount()>=dwTime)break;}
  }
  ::GlobalUnlock(hglobal);::GlobalFree(hglobal);
 }

/*************************************************************
  Subprogram is used for transforming the data in DataSrc
  into the format of IEEE488-DEFINITE-ARBITRARY-BLOCK-LENGTH
  which will be encoded with Hex and then transmitted to
  instrument through GPIB-CONTROLLER
  DataSrc:         Address of buffer to save the BLOCK-DATA
                   which will be output from communication
                   port.
  nBytesTransfer:  Number bytes of the BLOCK-DATA for
                   transmition in DataSrc,
                   for example: 1000,
                    it means that there are 1000 bytes
                    in DataSrc to transmite to instrument
                    through GPIB-CONTROLLER
  Wait_TicketTime: To setup the additional delay-time after the
                   data are output from communication port
                   each time.time unit is 1/1000 seconds, it is
                   necessary for time to match the requirement
                   for an interval of time between periods of
                   instrument receiving data from
                   GPIB-CONTROLLER each time,
                   for example: 50, it means 50 mili-seconds.
  DevAddress:      The value of instrument GPIB-ADDRESS.
  hComm:           the handle of communication port
                    Open method：
                      HANDLE hComm=::CreateFile
                      (
                       "COM3:128000,N,8,1",GENERIC_READ|GENERIC_WRITE,
                       0,NULL,
                       OPEN_EXISTING,
                       0,NULL
                      );
                    Close method:
                      CloseHandle(hfile);
*************************************************************/
 void IEEEBHexDataTransfer
 (
  HANDLE hComm,
  char *DataSrc, int nBytesTransfer,
  int Wait_TicketTime,
  char DevAddress
 )
 {
  HGLOBAL hglobal;LPSTR gbuf;char codebuf[256];
  int i,j,glen;DWORD nBytesWrite,dwTime;
 char Hex[16]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
 hglobal=GlobalAlloc(GHND,nBytesTransfer*2+240);
 gbuf=(LPSTR)GlobalLock(hglobal);
 j=1;while(1)
 {
  if(j>=nBytesTransfer){break;}
  j*=10;
 }
 glen=0;
 gbuf[glen++]='#';
 gbuf[glen++]=0x30+i;i=nBytesTransfer;
 while(1)
 {
  gbuf[glen++]=(i/j)+0x30;i=i%j;
  if(j==1){break;}
  else{j/=10;}
 }
 for(i=0;nBytesTransfer>i;i+=2)
 {
  gbuf[i]  = Hex[DataSrc[i/2]/16];
  gbuf[i+1]= Hex[DataSrc[i/2]%16];
 }
 gbuf-=glen;
 glen = nBytesTransfer*2 + glen;
 for(i=0;;i+=200)
 {
  if((glen-i)<=200)
  {
   if(!i){nBytesWrite=wsprintf(codebuf,"SEND^ %d",DevAddress);}
   memcpy(codebuf+nBytesWrite,"\'",1);nBytesWrite+=1;
   memcpy(codebuf+nBytesWrite,gbuf,glen-i);nBytesWrite+=(glen-i);
   memcpy(codebuf+nBytesWrite,"\'\r\n",3);nBytesWrite+=3;
  }
  else
  {
   nBytesWrite=wsprintf(codebuf,"SEND|^ %d",DevAddress);
   memcpy(codebuf+nBytesWrite,"\'",1);nBytesWrite+=1;
   memcpy(codebuf+nBytesWrite,gbuf,200);nBytesWrite+=200;gbuf+=200;
   memcpy(codebuf+nBytesWrite,"\'\r\n",3);nBytesWrite+=3;
  }
  WriteFile
  (hComm,
   codebuf,nBytesWrite,&nBytesWrite,NULL
  );
  if((glen-i)<=200){break;}
  dwTime=::GetTickCount()+Wait_TicketTime;
  while(1){if(::GetTickCount()>=dwTime)break;}
 }
 ::GlobalUnlock(hglobal);::GlobalFree(hglobal);
}
```

```
/****************************************************************
 description  :
  it utilize meter 34410a of Agilent-Technologies
  to calibrate and test power-supplier Hp662xa
  (Hp6621a,Hp6622a,Hp6623a,Hp6624a,Hp6627a) of Agilent Technologies)
 Application program designed for <b>GPIB-GONTROLLER</b>-
 windows is same as application program designed for
 rs232-windows, some of necessary sub-programs are almost
 same stated as below, instructions given are also in the
 same ways, but these programs can control multi-instruments
 in the same time(synchronously) and make the data send
 back from instrument through <b>GPIB-GONTROLLER</b> to pc
 quickly and stably, which are better than rs232.
 interface to control only. the program run a testing
 course (30 testing points), it take about 16 seconds,
 a testing point take less than a second, test report
 can be completed automatically once test finished, it
 will save a lot of testing cost.
****************************************************************/
#define ADDRESS34410A 7
#define ADDRESS662XA  5
HACCEL haccel;
char File_Place_Csn[256],irtool[256],modelno[256],channelno[256];
int model,channel;
HBRUSH hbr,hbrush,hbrushbtn,hbrushedit,hbrushstatic;
DWORD COMMAND_EDIT;
#define CmdBufSize                    10240
int StopTest=0,StopCheck=0;
static COMMTIMEOUTS commtimeouts;
COMMCONFIG cc;COMSTAT comstat;
HANDLE hComm=INVALID_HANDLE_VALUE;
const char Hex[16]=
{
 '0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'
};
HMENU menuportbaud,menuport;
char param[256];
#define BUFSIZE 10240
HINSTANCE hInst;
DWORD Wait_TicketTime=2000;
const struct BTAB
{
 char*  str;
}baud[56]=
{
  "2400"              ,"4800"              ,"6000"
 ,"6912"              ,"7500"              ,"7680"
 ,"8640"              ,"9000"              ,"9375"
 ,"9600"              ,"10000"             ,"10800"
 ,"11520"              ,"12000"              ,"12500"
 ,"12800"              ,"13824"              ,"14400"
 ,"15000"              ,"15625"              ,"16000"
 ,"17820"              ,"18000"              ,"18750"
 ,"19200"              ,"21600"              ,"23040"
 ,"24000"              ,"25000"              ,"28800"
 ,"30000"              ,"31250"              ,"32000"
 ,"34100"              ,"34560"              ,"36000"
 ,"36864"              ,"37500"              ,"38400"
 ,"43200"              ,"44300"              ,"46785"
 ,"46900"              ,"48000"              ,"49400"
 ,"50000"              ,"53600"              ,"57600"
 ,"57700"              ,"60000"              ,"62500"
 ,"64000"              ,"115200"             ,"128000"
 ,"256000"             ,""
};
LRESULT CALLBACK  GetProReg(LPSTR,LPSTR,LPSTR,LPSTR);
LRESULT CALLBACK  SetProReg(LPSTR,LPSTR,LPSTR);
LRESULT CALLBACK  WindowsProc(HWND,UINT,WPARAM,LPARAM);
void CALLBACK     InitCommPort(HWND,char *);
LRESULT CALLBACK  SelOutFile(HWND);
LPSTR CALLBACK WrToBusAndRdFrBus
(
 HWND,DWORD,LONG,HANDLE,char *,char *
);
LRESULT CALLBACK SetProReg
(
 LPSTR secstr,LPSTR keystr,LPSTR txtstr
)
{
 long lResult=0;HKEY hkGlobal= 0;
 DWORD dwDisposition=REG_CREATED_NEW_KEY;
 char keybuf[256],section[256];
 wsprintf(section,"Irtool\\%s",secstr);
 lResult=
 ::RegCreateKeyEx
 (
  HKEY_CURRENT_USER,
  section,
  0,
  "",
  0,
  KEY_ALL_ACCESS,
  NULL,
  &hkGlobal,
  &dwDisposition
 );
 if(lResult==ERROR_SUCCESS)
 {
  lResult=::RegSetValueEx
```

```
   (
    hkGlobal,
    keystr,
    0,
    REG_SZ,
    (CONST BYTE *)txtstr,
    strlen(txtstr)
   );
   ::RegCloseKey(hkGlobal);
  }
  return lResult;
}
LRESULT CALLBACK GetProReg
(
 LPSTR secstr,LPSTR keystr,LPSTR txtstr,LPSTR defstr
)
{
 long lResult=0;HKEY hkGlobal= 0;
 DWORD dwDisposition=REG_CREATED_NEW_KEY,dwType=REG_SZ,dwData;
 char section[256],keybuf[256],txtbuf[256];
 wsprintf(section,"Irtool\\%s",secstr);
 lResult=
 ::RegOpenKeyEx
 (
  HKEY_CURRENT_USER,
  section,
  0,
  KEY_ALL_ACCESS,
  &hkGlobal
 );
 dwData=sizeof(txtbuf);
 if(lResult==ERROR_SUCCESS)
 {
  lResult=::RegQueryValueEx
  (
   hkGlobal,
   keystr,
   0,
   &dwType,
   (BYTE*)txtbuf,
   &dwData
  );
  if(lResult==ERROR_SUCCESS && dwType==REG_SZ)
  {
   ::RegCloseKey(hkGlobal);
   memcpy(txtstr,txtbuf,dwData);
  }
  else
  {
   memcpy(txtstr,defstr,strlen(defstr));
```

```
    goto REGDEFAULT;
   }
  }
  else
  {
   memcpy(txtstr,defstr,strlen(defstr));
   REGDEFAULT:
   lResult=::RegCreateKeyEx
   (
    HKEY_CURRENT_USER,
    section,
    0,
    "",
    0,
    KEY_ALL_ACCESS,
    NULL,
    &hkGlobal,
    &dwDisposition
   );
   if(lResult==ERROR_SUCCESS)
   {
    lResult=::RegSetValueEx
    (
     hkGlobal,
     keystr,
     0,
     REG_SZ,
     (CONST BYTE *)defstr,
     strlen(defstr)
    );
    ::RegCloseKey(hkGlobal);
   }
  }
  return lResult;
}
/************************************************************
 sub-program is called by windows-proc will be utilized
 to calibrate Hp662xa voltage-accuracy and also take
 responsible for pc communication device which make
 communication and cooperate actions between 34410a and
 Hp662xa through <b>ki-usb/gpib-controller</b>.
 hWnd:       handle of main-window
 hComm:      handle to communications device
 ID_EDITBOX: identifier of editbox control
************************************************************/
LRESULT CALLBACK VCal
(
 HWND hWnd,HANDLE hComm,LONG ID_EDITBOX
)
{
```

```
int i,nlen;
char buf[256],inbuf[256],outbuf[256],textbuf[1024];
DWORD dwTime,nBytesWrite;float VoltLow,VoltHigh;
InitCommPort(hWnd,param);
::SetWindowText(::GetDlgItem(hWnd,IDC_EDIT1),"");
UpdateWindow(::GetDlgItem(hWnd,IDC_EDIT1));
/*********************************************************
 configuration made as below:
 HP662xa configured to MOD1
 34410a  configured to 'volt/dc/auto range'
*********************************************************/
sprintf
(inbuf,
 "SEND %d \'CMOD 1\';SEND %d \'conf:volt:dc auto\'\r\n"
 ,ADDRESS662XA,ADDRESS34410A
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,0,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,
 0,(LPARAM)inbuf
);
::WriteFile(hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL);</font>
/*********************************************************
 the contents of data displayed in title-bar of
 main-window are the data sent from pc to Hp662xa
 and 34410a through <b>ki-usb/gpib-controller</b>.
*********************************************************/
sprintf
(inbuf,
 "SEND %d \'VLO %d\';SEND? %d \'MEAS?\'\r\n"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
/*********************************************************
 according to the regulation of HP662xa-voltage-calibration
 in HP662xa user manual, pc communicate with Hp662xa and
 34410a through <b>ki-usb/gpib-controller</b> which make Hp662xa
 and 34410a do actions for requirement.
*********************************************************/
sprintf
(inbuf,
 "SEND %d \'VLO %d\';SEND? %d \'MEAS?\'"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
WrToBusAndRdFrBus
(
 hWnd,Wait_TicketTime,ID_EDITBOX,hComm,inbuf,outbuf
);
VoltLow=atof(outbuf);
sprintf
(inbuf,
 "%s\r\n"
 ,outbuf
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
sprintf
(inbuf,
 "SEND %d \'VHI %d\';SEND? %d \'MEAS?\'\r\n"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
sprintf
(inbuf,
 "SEND %d \'VHI %d\';SEND? %d \'MEAS?\'"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
WrToBusAndRdFrBus
(
 hWnd,Wait_TicketTime,ID_EDITBOX,hComm,inbuf,outbuf
);
VoltHigh=atof(outbuf);
sprintf
(inbuf,
 "%s\r\n"
 ,outbuf
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
sprintf
(inbuf,
 "SEND %d \'VDATA %d,%f,%f\'\r\n"
 ,ADDRESS662XA,channel,VoltLow,VoltHigh
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
::WriteFile
(
 hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL
);
```

```
sprintf(inbuf,"SEND %d \'OVCAL\'\r\n",ADDRESS662XA);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
::WriteFile
(
 hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL
);
dwTime=::GetTickCount()+11000;
while(1){if(::GetTickCount()>=dwTime)break;}
sprintf
(inbuf,
 "SEND %d \'CMOD 0\'\r\n"
 ,ADDRESS662XA
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,
 0,(LPARAM)inbuf
);
::WriteFile
(
 hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
```

```
 (
  ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)"VCal ok!"
 );
 return 0L;
}
/**********************************************************
 sub-program is called by windows-proc will be utilized
 to calibrate Hp662xa current-accuracy and also take
 responsible for pc communication device which make
 communication and cooperate actions between 34410a
 and Hp662xa through <b>ki-usb/gpib-controller</b>.
 hWnd:       handle of main-window
 hComm:      handle to communications device
 ID_EDITBOX: identifier of editbox control
**********************************************************/
LRESULT CALLBACK ICal
(
 HWND hWnd,HANDLE hComm,LONG ID_EDITBOX
)
{
 int i,nlen;char buf[256],inbuf[256],outbuf[256];
 DWORD dwTime,nBytesWrite;char cpybuf[1024];
 float VoltLow,VoltHigh;
 InitCommPort(hWnd,param);
 ::SetWindowText(::GetDlgItem(hWnd,IDC_EDIT1),"");
 UpdateWindow(::GetDlgItem(hWnd,IDC_EDIT1));
 /**********************************************************
  configuration made as below:
  HP662xa configured to MOD1
  34410a  configured to 'volt/dc/auto range'
 **********************************************************/
 sprintf
 (inbuf,
  "SEND %d \'CMOD 1\';SEND %d \'conf:volt:dc auto\'\r\n"
  ,ADDRESS662XA,ADDRESS34410A
 );
 nlen=::SendMessage
 (
  ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
 );
 ::SendMessage
 (
  ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,0,nlen
 );
 ::SendMessage
 (
  ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
 );
 ::WriteFile
 (
 hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL
);
/*********************************************************
 the contents of data displayed in title-bar of
 main-window are the data sent from pc to Hp662xa
 and 34410a through <b>ki-usb/gpib-controller</b>.
*********************************************************/
sprintf
(inbuf,
 "SEND %d \'ILO %d\';SEND? %d \'MEAS?\'\r\n"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
/*********************************************************
 according to the regulation of Hp662xa-current-calibration
 in Hp662xa user manual, pc communicate with Hp662xa and
 34410a through <b>ki-usb/gpib-controller</b> which make Hp662xa
 and 34410a do actions for requirement.
*********************************************************/
sprintf
(inbuf,
 "SEND %d \'ILO %d\';SEND? %d \'MEAS?\'"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
WrToBusAndRdFrBus
(
 hWnd,Wait_TicketTime,ID_EDITBOX,hComm,inbuf,outbuf
);
VoltLow=atof(outbuf);
sprintf
(inbuf,
 "%s\r\n"
 ,outbuf
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
```

```
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
sprintf
(inbuf,
 "SEND %d \'IHI %d\';SEND? %d \'MEAS?\'\r\n"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
sprintf
(inbuf,
 "SEND %d \'IHI %d\';SEND? %d \'MEAS?\'"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
WrToBusAndRdFrBus
(
 hWnd,Wait_TicketTime,ID_EDITBOX,hComm,inbuf,outbuf
);
VoltHigh=atof(outbuf);
sprintf
(inbuf,
 "%s\r\n"
 ,outbuf
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
sprintf
(inbuf,
 "SEND %d \'IDATA %d,%f,%f\'\r\n"
  ,ADDRESS662XA,channel,VoltLow,VoltHigh
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
::WriteFile
(
 hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL
);
sprintf
(inbuf,
 "SEND %d \'CMOD 0\'\r\n",ADDRESS662XA
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)inbuf
);
::WriteFile
(
 hComm,inbuf,strlen(inbuf),&nBytesWrite,NULL
);
nlen=::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
);
::SendMessage
(
 ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
);
```

```
 ::SendMessage
 (
  ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)"ICal ok!"
 );
 return 0L;
}
/*******************************************************
 sub-program is called by windows-proc and executed to make
 communication from pc to instruments through GPIB-GONTROLLER
 hWnd:       handle of main-window
 hComm:      handle to communications device
 ID_EDITBOX: identifier of editbox control
 WrDataStr:  address of buffer for text-data which is
             prepared to send from communication port to
             instrument through <b>ki-usb/gpib-controller</b>.
*******************************************************/
LRESULT CALLBACK CheckPower
(
 HWND hWnd,HANDLE hComm,LONG ID_EDITBOX
)
{
 int i,nlen,typeoftest;
 char inbuf[256],outbuf[256],textbuf[1024];
 DWORD dwTime;HANDLE hfile;DWORD nByteWrite;
 /*****************************************************
  The list of testing point voltage-value for Hp662xa.
 *****************************************************/
 const struct
 {
  float point[30];
 }test=
 {
  1,1.5,2,2.5,3,3.5,4,4.5,5,5.5,6,
  6.5,7,7.5,8,8.5,9,9.5,10,10.5,
  11,12,13,14,15,16,17,18,19,20
 };
 const struct
 {
  float point[30];
 }test3=
 {
  1.0, 2.6, 4.2, 5.8, 7.4, 9.0,10.6,12.2,13.8,15.4,
  17, 18.7,20.4,22.1,23.8,25.5,27.1,28.8,30.5,32.2,
  34, 35.9,37.8,39.7,41.6,43.5,45.4,47.3,49.2,50.0
 };
 InitCommPort(hWnd,param);
 ::SetWindowText(::GetDlgItem(hWnd,IDC_EDIT1),"");
 UpdateWindow(::GetDlgItem(hWnd,IDC_EDIT1));
 /*******************************************************
  The channel of power-supplier(HP662xa) which we are testing
 will be configured to 'output-on',and 34410a meter
 configured to 'volt/dc/auto range'
*******************************************************/
sprintf
(inbuf,
 "SEND %d \'OUT %d,1\';SEND %d \'conf:volt:dc auto\'"
 ,ADDRESS662XA,channel,ADDRESS34410A
);
::WriteFile
(
 hComm,inbuf,strlen(inbuf),&nByteWrite,NULL
);
if
(
 model == 6622                  ||
 model == 6623 && channel ==3  ||
 model == 6624 && channel ==3  ||
 model == 6624 && channel ==4  ||
 model == 6627
){typeoftest=3;}
else{typeoftest=0;}
/*****************************************************
 to establish new and empty file of testing report
*****************************************************/
hfile=
::CreateFile
(
 File_Place_Csn,GENERIC_READ|GENERIC_WRITE,0,NULL,
 CREATE_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL
);
SetFilePointer(hfile,0,NULL,FILE_BEGIN);
CloseHandle(hfile);
for
(
 i=0;
 i<(sizeof(test)/sizeof(float))  && typeoftest==0 ||
 i<(sizeof(test3)/sizeof(float)) && typeoftest==3    ;
 i++
)
{
 if(StopCheck){StopCheck=0;break;}
 /*****************************************************
  pc give instruction "send" through GPIB-GONTROLLER
  to send "volt" (command of 6611c power supply) to Hp662xa
  and also the value of measuring will be read from 34410a
  through <b>GPIB-GONTROLLER</b>.
 *****************************************************/
 if(typeoftest==0)
 {
  sprintf
  (inbuf,
   "SEND %d \'VSET %d,%f\';SEND? %d \'MEAS?\'"
   ,ADDRESS662XA,channel,test.point[i],ADDRESS34410A
  );
  WrToBusAndRdFrBus
  (
   hWnd,Wait_TicketTime,ID_EDITBOX,hComm,inbuf,outbuf
  );
  *textbuf=0;
  while(strlen(textbuf)<9){strcat(textbuf," ");}
  sprintf
  (textbuf+strlen(textbuf),
   "INPUT = %+3.3f",test.point[i]
  );
 }
 else
 {
  sprintf
  (inbuf,
   "SEND %d \'VSET %d,%f\';SEND? %d \'MEAS?\'"
   ,ADDRESS662XA,channel,test3.point[i],ADDRESS34410A
  );
  WrToBusAndRdFrBus
  (
   hWnd,Wait_TicketTime,ID_EDITBOX,hComm,inbuf,outbuf
  );
  *textbuf=0;
  while(strlen(textbuf)<9){strcat(textbuf," ");}
  sprintf
  (textbuf+strlen(textbuf),
   "INPUT = %+3.3f",test3.point[i]
  );
 }
 while(strlen(textbuf)<31){strcat(textbuf," ");}
 sprintf(textbuf+strlen(textbuf),"OUTPUT = %s",outbuf);
 while(strlen(textbuf)<63){strcat(textbuf," ");}
 /*****************************************************
  for combination with format of testing report, pc read
  the value of measurement (A) from 34410a meter and VOLT
  value (B) given to Hp662xa from pc and the accurate
  value (C)=(B-A)/B calculating from (A) and (B), and then
  the value A, B, C will be made into arrangement as well
  as formulation to display in the main-window of edit-box
  and sent to the file of testing report at the same time.
 *****************************************************/
 if(typeoftest==3)
 {
  sprintf
  (textbuf+strlen(textbuf),
   "ACCURACY = %+3.3f %%\r\n"
```

```
    ,((atof(outbuf)-test3.point[i])/fabs(test3.point[i]))*100
   );
  }
  else
  {
   sprintf
   (textbuf+strlen(textbuf),
    "ACCURACY = %+3.3f %%\r\n"
    ,((atof(outbuf)-test.point[i])/fabs(test.point[i]))*100
   );
  }
  nlen=::SendMessage
  (
   ::GetDlgItem(hWnd,ID_EDITBOX),WM_GETTEXTLENGTH,0,0
  );
  ::SendMessage
  (
   ::GetDlgItem(hWnd,ID_EDITBOX),EM_SETSEL,nlen,nlen
  );
  ::SendMessage
  (
   ::GetDlgItem(hWnd,ID_EDITBOX),EM_REPLACESEL,0,(LPARAM)textbuf
  );
  hfile=
  ::CreateFile
  (
   File_Place_Csn,GENERIC_READ|GENERIC_WRITE,0,NULL,
   OPEN_ALWAYS,FILE_ATTRIBUTE_NORMAL,NULL
  );
  SetFilePointer(hfile,0,NULL,FILE_END);
  WriteFile
  (
   hfile,textbuf,strlen(textbuf),&nByteWrite,NULL
  );
  CloseHandle(hfile);
 }
 return 0L;
}
/*******************************************************
 subprogram is called by CheckPower to send the data to
 instrument from communication device through GPIB-CONTROLLER.
 In the same way, the data responded to Communicate device from
 instrument will be read through GPIB-CONTROLLER.
 hWnd:           handle of main-window
 WrDataStr:      address of buffer for text-data which is
                 prepared to send to communications device
 ID_EDITBOX:     identifier of editbox control
 hComm:          handle to communications device
 RdDataStr:      address of buffer for saving data received
                 from communications device
```

```
**********************************************************/
LPSTR CALLBACK WrToBusAndRdFrBus
(
 HWND hWnd,DWORD Wait_TicketTime,LONG ID_EDITBOX,
 HANDLE hComm,char *WrDataStr,char *RdDataStr
)
{
 MSG Message;RECT btnRect;int i,n;char buf[256],rbuf[1024];
 DWORD nBytesWrite,nBytesRead,dwEvent,dwError,dwTime;
 /**********************************************************
  pc send the contents of WrDataStr to 34410a(multi-meter)
  through <b>GPIB-GONTROLLER</b>
 **********************************************************/
 ::SetWindowText(hWnd,WrDataStr);
 wsprintf(rbuf,"%s\r\n",WrDataStr);
 ::WriteFile(hComm,rbuf,strlen(rbuf),&nBytesWrite,NULL);
 dwTime=GetTickCount()+Wait_TicketTime;*RdDataStr=0;
 while(1)
 {
  if(GetTickCount()>=dwTime)
  {
   if(StopTest)
   {
    StopCheck=1;StopTest=0;
    PostMessage(hWnd,WM_SYSCOMMAND,SC_CLOSE,0);
   }
   return NULL;
  }
  /**********************************************************
   pc used the ways of background-processing to do works
   that maintain the window's operation for requirement.
  **********************************************************/
  if(::PeekMessage(&Message,NULL,0,0,PM_REMOVE))
  {
   if
   (
    (Message.message==WM_NCLBUTTONDOWN&&Message.wParam==0x14)  ||
    (Message.message==WM_SYSCOMMAND&&Message.wParam==SC_CLOSE) ||
    (Message.message==WM_KEYDOWN&&LOWORD(Message.wParam)==0x1b)
   )
   {
    StopTest=1;
   }
   else if
   (
    Message.message==WM_RBUTTONDOWN                &&
    Message.hwnd==::GetDlgItem(hWnd,IDC_BUTTON3)
   )
   {
    /**********************************************************
    to set up the baudrate of communication port for rs232.
   *******************************************************/
   ::GetWindowRect(::GetDlgItem(hWnd,IDC_BUTTON3),&btnRect);
   ::TrackPopupMenu
   (
    menuportbaud,
    TPM_LEFTALIGN|TPM_RIGHTBUTTON,
    btnRect.left,btnRect.bottom-1,0,
    hWnd,NULL
   );
  }
  else if
  (
   Message.message==WM_LBUTTONDOWN                &&
   Message.hwnd==::GetDlgItem(hWnd,IDC_BUTTON3)
  )
  {
   /*******************************************************
    to set up the port-number of communication port for rs232.
   *******************************************************/
   ::GetWindowRect(::GetDlgItem(hWnd,IDC_BUTTON3),&btnRect);
   menuport=GetSubMenu
   (
    LoadMenu
    (
     (HINSTANCE)GetWindowLong(hWnd,GWL_HINSTANCE),
     MAKEINTRESOURCE(IDC_HP662XACAL)
    ),
    2
   );
   ::TrackPopupMenu
   (
    menuport,
    TPM_LEFTALIGN|TPM_RIGHTBUTTON,
    btnRect.left,btnRect.bottom-1,0,
    hWnd,NULL
   );
  }
  else
  {
   ::TranslateMessage(&Message);::DispatchMessage(&Message);
   if(haccel!=NULL)
   {
    (::TranslateAccelerator(hWnd,haccel,&Message));
   }
  }
 }
/*******************************************************
  same situation as above, pc used the ways of
  background-processing to poll the communication
  port, once there are the data responded from
  instrument to pc through <b>GPIB-GONTROLLER</b>,
  pc will read those data and save in buffer of
  RdDataStr;
 *************************************************************/
 if(::GetCommMask(hComm,&dwEvent))
 {
  ::ClearCommError(hComm,&dwError,&comstat);
  if
  (
   ::ReadFile(hComm,rbuf,comstat.cbInQue,&nBytesRead,NULL) &&
   nBytesRead
  )
  {
   rbuf[nBytesRead]=0;n=nBytesRead;
   wsprintf(buf,"%s",rbuf);
   for(i=0;i<nBytesRead;i++)
   {
    if(buf[i]!='\r'&&buf[i]!='\n')
    {
     wsprintf
     (RdDataStr+strlen(RdDataStr),
      "%c"
      ,buf[i]
     );
    }
    if
    (
     i&&
     (
      buf[i-1]=='\r'&&buf[i]=='\n'
     )
    ){goto COMMANDOK;}
    else if(buf[i]=='\n')
    {
     goto COMMANDOK;
    }
   }
  }
 }
 continue;
 COMMANDOK:
 wsprintf(buf,"%s",RdDataStr);
 if(StopTest)
 {
  StopCheck=1;StopTest=0;
  PostMessage(hWnd,WM_SYSCOMMAND,SC_CLOSE,0);
  return NULL;
 }
 return (LPSTR)buf;
```

```
 }
 if(StopTest)
 {
  StopCheck=1;StopTest=0;
  PostMessage(hWnd,WM_SYSCOMMAND,SC_CLOSE,0);
  return NULL;
 }
 return NULL;
}
/*******************************************************
 In accordance with the contents of requirement from
 parameter. the communication port pointed out by content
 of para will be configured to the format requested by para
 contents. for instance : when the content of para is
 "com3:128000,n,8,1" , it represents that communication-port
 `3` will be configured to the format of :
     baudrate-128000,no-parity,data-8bit,stop-1bit

 hWnd:       handle of main-window
*******************************************************/
void CALLBACK  InitCommPort(HWND hWnd,char *para)
{
 #define PORTCOUNT 256
 char paraZip[32],*ptr;int i,j;DWORD dwError;
 char ComText[256],PortText[256],BaudText[256];
 ::ClearCommError(hComm,&dwError,&comstat);
 ::CloseHandle(hComm);
 wsprintf(PortText,"%s",para);
 ptr=strstr(PortText,":");if(ptr)*ptr=0;
 for(i=0;i<strlen(PortText);i++)
 {
  PortText[i]=toupper(PortText[i]);
 }
 ptr=strstr(PortText,"COM");
 if(ptr){i=atoi(ptr+3);}else{return;}
 wsprintf(ComText,"\\\\.\\%s",para);
 ptr=strstr(ComText,":");
 if(ptr){*ptr=0;}else{goto quit;}
 wsprintf(paraZip,"%s",para);
 hComm=::CreateFile
 (
  ComText,GENERIC_READ|GENERIC_WRITE,0,NULL,
  OPEN_EXISTING,0,NULL
 );
 if(hComm!=INVALID_HANDLE_VALUE)
 {
  goto FINDPORT;
 }
 else
 {
  for(j=1;j<=PORTCOUNT;j++)
  {
   wsprintf(PortText,"%s",paraZip+3);
   ptr=strstr(PortText,":");if(ptr){*ptr=0;}
   wsprintf(BaudText,"%s",ptr+1);
   i=atoi(PortText)+1;if(i>PORTCOUNT){i=1;}
   wsprintf(ComText,"\\\\.\\COM%d",i);
   wsprintf(paraZip,"COM%d:%s",i,BaudText);
   hComm=::CreateFile
   (
    ComText,GENERIC_READ|GENERIC_WRITE,0,NULL,
    OPEN_EXISTING,0,NULL
   );
   if(hComm!=INVALID_HANDLE_VALUE)goto FINDPORT;
  }
  if(j==PORTCOUNT){goto quit;}
 }
 FINDPORT:
 strcpy(param,paraZip);
 ::SetWindowText(::GetDlgItem(hWnd,IDC_BUTTON3),param);
 if(::BuildCommDCB(param,&cc.dcb)==FALSE)
 {
  ERRRTN:
  ::ClearCommError(hComm,&dwError,&comstat);
  goto quit;
 }
 if(::SetCommState(hComm,&cc.dcb)==FALSE)
 {
  goto ERRRTN;
 }
 quit:
 ::SetupComm(hComm,BUFSIZE,BUFSIZE);
 ::SetCommMask(hComm,EV_RXCHAR);
 ::ClearCommError(hComm,&dwError,&comstat);
 return;
}
/*******************************************************
 subprogram utilized to save the file name and path of
 test report for the data of measuring results.
 for instance : c:\temp\listfile\34410aCheck6611c.rpt
*******************************************************/
LRESULT CALLBACK SelOutFile(HWND hWnd)
{
 HANDLE hFile;
 char inipath[256],ext[256],SendBuf[1024],*ptr,*multptr;
 int i,j,n;DWORD nByteWrite;HANDLE hfile;char buf[256];
 OPENFILENAME FAR ofn;memset(&ofn,0,sizeof(OPENFILENAME));
 char pathbuf[256],mkdirbuf[256],dirbuf[256],zfname[256];
 ::GetProReg(irtool,"SelectFileShowIpath",inipath,"c:\\temp");
 ::SetCurrentDirectory(inipath);
```

```
memset(zfname,0,sizeof(zfname));
::GetProReg(irtool,"OutFile",zfname,"*.rpt");
wsprintf(buf,"%s",zfname);
GetFullPathName(buf,sizeof(zfname),zfname,NULL);
ptr=strstr(zfname,".");if(ptr)sprintf(ext,"*.%s",ptr);
if(ptr)
{
 sprintf
 (
  SendBuf,
  "%s files (*.%s)|*.%s|rpt files (*.rpt)|*.rpt|"
  "All files (*.*)|*.*||"
  ,ptr+1,ptr+1,ptr+1
 );
}
else
{
 strcpy
 (
  SendBuf,
  "rpt files (*.rpt)|*.rpt|"
  "All files (*.*)|*.*||"
 );
}
for(i=strlen(SendBuf);i>=0;i--)
{
 if(SendBuf[i]=='|')SendBuf[i]=0;
}
ofn.lStructSize=sizeof(OPENFILENAME);
ofn.hwndOwner = NULL;
ofn.lpstrFilter = (LPSTR)SendBuf;
ofn.nFilterIndex = 1;
ofn.lpstrFile= (LPSTR)zfname;
ofn.nMaxFile=sizeof(zfname);
ofn.nFileOffset=0;
ofn.lpstrFileTitle =NULL;
ofn.nMaxFileTitle = 0;
ofn.lpstrInitialDir = NULL;
ofn.Flags=OFN_SHOWHELP|OFN_ENABLESIZING|OFN_EXPLORER|OFN_LONGNAMES;
ofn.lpstrTitle="select filename for list box items";
if(::GetOpenFileName(&ofn)){multptr=zfname;}else{return 0;}
wsprintf(File_Place_Csn,"%s",multptr);
if(_access(File_Place_Csn,0))
{//
 wsprintf(pathbuf,"%s",File_Place_Csn);
 ptr=strrchr(pathbuf,'\\');if(ptr)*ptr=0;
 if
 (
  !SetCurrentDirectory(pathbuf)
 )
```

```
  {
   wsprintf(mkdirbuf,"%s",pathbuf);
   ptr=strstr(mkdirbuf,":");
   if(ptr==NULL)
   {
    GetCurrentDirectory(sizeof(dirbuf),dirbuf);       dirbuf[2]=0;
    wsprintf(mkdirbuf,"%s\\",dirbuf);SetCurrentDirectory(mkdirbuf);
    wsprintf(mkdirbuf,"%s",pathbuf);
   }
   else
   {
    wsprintf(dirbuf,"%s",mkdirbuf);                     dirbuf[2]=0;
    wsprintf(mkdirbuf,"%s\\",dirbuf);SetCurrentDirectory(mkdirbuf);
    ptr=strstr(pathbuf,"\\");if(ptr){wsprintf(mkdirbuf,"%s",ptr+1);}
   }
   if(*mkdirbuf)
   {
    wsprintf(mkdirbuf+strlen(mkdirbuf),"\\");
    ptr=strtok(mkdirbuf,"\\");
    if(ptr)
    {
     _mkdir(ptr);
     wsprintf(dirbuf+strlen(dirbuf),"\\%s",ptr);
     while(1)
     {
      ptr=strtok(NULL,"\\");
      if(ptr)
      {
       SetCurrentDirectory(dirbuf);
       _mkdir(ptr);
       wsprintf(dirbuf+strlen(dirbuf),"\\%s",ptr);
       continue;
      }
      else
      {
       break;
      }
     }
    }
   }
  }
  SetProReg(irtool,"OutFile",File_Place_Csn);
  ::SendDlgItemMessage
  (
   hWnd,IDC_FILENAME,WM_SETTEXT,0,
   (LPARAM)((LPSTR)File_Place_Csn)
  );
 }
}
/*******************************************************
```

```
 main window procedure called by subprogram-DialogBox
 hWnd    : handle of main-window
 message : value identifying the window-message
 wParam  : first message parameter
 lParam  : second message parameter
************************************************************/
LRESULT CALLBACK WindowsProc
(
 HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam
)
{
 int i,nCX,nCY;RECT btnRect,newrect,rect;MSG Message;
 char *ptr,*sptr,*sptr2,path[512],CurDir1[512],inipath[512];
 char irpath[512],fname[512],exename[512],title[512],irfname[512];
 char text[512],textbuf[512],ifname[512],buf[1024];
 switch (message)
 {
  case WM_INITDIALOG:
  {
   HICON hicon=::LoadIcon
   (
    hInst,MAKEINTRESOURCE(IDI_HP662XACAL)
   );
   haccel=::LoadAccelerators
   (
    (HINSTANCE)GetWindowLong(hWnd,GWL_HINSTANCE),
    MAKEINTRESOURCE(IDC_HP662XACAL)
   );
   ::SetClassLong(hWnd,GCL_HICON,(LONG)hicon);
   hbr=::CreateSolidBrush((COLORREF)0x00ffff80);
   hbrushbtn=::CreateSolidBrush((COLORREF)13673215);
   hbrush=(HBRUSH)::GetStockObject(HOLLOW_BRUSH);
   hbrushstatic=::CreateSolidBrush((COLORREF)(65280));
   ::GetModuleFileName(hInst,exename,sizeof(exename));
   ptr=&(exename[strlen(exename)]);
   while(1)
   {
    if((*ptr)=='\\'||(*ptr)==':'){ptr++;break;}
    else ptr--;
   }
   strcpy(fname,ptr);ptr=strstr(fname,".");if(ptr){*ptr=0;}
   strcpy(irpath,exename);sptr2=strstr(irpath,ptr);
   if(sptr2!=NULL){if((*(sptr2-1))=='\\')(*(sptr2-1))=0;
   else (*(sptr2 ))=0;}
   strcpy(path,irpath);strcat(path,"\\");
   strcpy(irtool,fname);sptr2=strstr(irtool,".");
   if(sptr2)*sptr2=0;
   GetProReg(irtool,"COMM",ifname,"Hp662xaCal");
   strcpy(title,irfname);
   strcpy(path,inipath);
```

```
ptr=&(path[strlen(path)]);
while(1){if((*ptr)=='\\'||(*ptr)==':'){break;}else ptr--;}
if(*ptr=='\\'){*(ptr)=0;}else{*(ptr+1)=0;}
::SendMessage
(
 hWnd,WM_SETTEXT,0,
 (LPARAM)"Hp662xaCal"
);
/**********************************************************
 to move the main-window to center of desktop
**********************************************************/
nCX=GetSystemMetrics(SM_CXSCREEN)/2;
nCY=GetSystemMetrics(SM_CYSCREEN)/2;
::GetWindowRect(hWnd,&rect);
newrect.top=nCY-(rect.bottom-rect.top)/2-20;
newrect.left=nCX-(rect.right-rect.left)/2;
newrect.bottom=(rect.bottom-rect.top);
newrect.right=(rect.right-rect.left);
::MoveWindow
(
 hWnd,newrect.left,newrect.top,newrect.right,
 newrect.bottom,TRUE
);
/**********************************************************
 centering-main-window to complete
**********************************************************/
::ShowWindow(hWnd,SW_SHOW);
menuport=::CreatePopupMenu();
for(i=0;i<256;i++)
{
 wsprintf(buf,"COM%d",i+1);
 if(i&&((i)%29)==0)
 {
  AppendMenu
  (
   menuport,MF_STRING|MF_MENUBREAK,IDM_COMPORT1+i,buf
  );
 }
 else
 {
  AppendMenu
  (
   menuport,MF_STRING,IDM_COMPORT1+i,buf
  );
 }
}
menuportbaud=::CreatePopupMenu();
for(i=0;;i++)
{
 if(*(baud[i].str)==0){break;}
 wsprintf(buf,"%s",baud[i].str);
 if(i && ((i)%14)==0)
 {
  AppendMenu
  (
   menuportbaud,MF_STRING|MF_MENUBREAK ,IDM_BAUD2400+i,buf
  );
 }
 else
 {
  AppendMenu
  (
   menuportbaud,MF_STRING,IDM_BAUD2400+i,buf
  );
 }
}
GetProReg(irtool,"COMM",buf,"COM1:128000,N,8,1");
InitCommPort(hWnd,buf);
commtimeouts.ReadIntervalTimeout =  MAXDWORD;
commtimeouts.ReadTotalTimeoutMultiplier =  1;
commtimeouts.WriteTotalTimeoutMultiplier=  1;
commtimeouts.WriteTotalTimeoutConstant =   0;
commtimeouts.ReadTotalTimeoutConstant = 1000;
SetCommTimeouts(hComm,&commtimeouts);
GetProReg(irtool,"ModelChannel",buf,"6623Ch1");
wsprintf(channelno,"%s",buf+strlen("662x"));
wsprintf(modelno,"%s",buf);modelno[strlen("662x")]=0;
if(!strcmp(modelno,"6621"))
{
 model=6621;
}
else if(!strcmp(modelno,"6622"))
{
 model=6622;
}
else if(!strcmp(modelno,"6623"))
{
 model=6623;
}
else if(!strcmp(modelno,"6624"))
{
 model=6624;
}
else if(!strcmp(modelno,"6627"))
{
 model=6627;
}
else
{
 wsprintf(modelno,"6623");
 model=6623;
}
if
(
 !strcmp(modelno,"6621")||
 !strcmp(modelno,"6622")
)
{
 if(!strcmp(channelno,"Ch1"))
 {
  channel=1;
 }
 else if(!strcmp(channelno,"Ch2"))
 {
  channel=2;
 }
 else
 {
  wsprintf(channelno,"Ch1");
  channel=1;
 }
}
else if
(
 !strcmp(modelno,"6624")||
 !strcmp(modelno,"6627")
)
{
 if(!strcmp(channelno,"Ch1"))
 {
  channel=1;
 }
 else if(!strcmp(channelno,"Ch2"))
 {
  channel=2;
 }
 else if(!strcmp(channelno,"Ch3"))
 {
  channel=3;
 }
 else if(!strcmp(channelno,"Ch4"))
 {
  channel=4;
 }
 else
 {
  wsprintf(channelno,"Ch1");
  channel=1;
 }
}
```

```
else if
(
 !strcmp(modelno,"6623")
)
{
 if(!strcmp(channelno,"Ch1"))
 {
  channel=1;
 }
 else if(!strcmp(channelno,"Ch2"))
 {
  channel=2;
 }
 else if(!strcmp(channelno,"Ch3"))
 {
  channel=3;
 }
 else
 {
  wsprintf(channelno,"Ch1");
  channel=1;
 }
}
wsprintf(buf,"%s%s",modelno,channelno);
::SetWindowText(::GetDlgItem(hWnd,IDC_SELOUTP),buf);
SetProReg(irtool,"modelchannel",buf);
wsprintf(buf,"%s%s",modelno,channelno);
SetProReg(irtool,"ModelChannel",buf);
::SetWindowText(::GetDlgItem(hWnd,IDC_SELOUTP),buf);
wsprintf(buf,"c:\\temp\\listfile\\%s.rpt",irtool);
GetProReg(irtool,"OutFile",File_Place_Csn,buf);
SetProReg(irtool,"OutFile",File_Place_Csn);
::SetWindowText
(
 ::GetDlgItem(hWnd,IDC_FILENAME),File_Place_Csn
);
while(1)
{
 /**********************************************************
  For the purpose of executing the action taken for our
  requirement and maintaining the necessary work of window
  operrtion during the special event or message happens,
  the reentrance feature of application-program and the
  background-processing method of windows are used to
  achieve the purpose.
 **********************************************************/
 if(::PeekMessage(&Message,NULL,0,0,PM_REMOVE))
 {
  if
  (
 (Message.message==WM_NCLBUTTONDOWN&&Message.wParam==0x14)   ||
 (Message.message==WM_SYSCOMMAND&&Message.wParam==SC_CLOSE)  ||
 (Message.message==WM_KEYDOWN&&LOWORD(Message.wParam)==0x1b)
)
{
 /*********************************************************
  work done and close main-window
 *********************************************************/
 DeleteObject((HBRUSH)hbr);
 DeleteObject((HBRUSH)hbrushbtn);
 DeleteObject((HBRUSH)hbrush);
 EndDialog(hWnd, LOWORD(Message.wParam));
 PostQuitMessage(0);
 return TRUE;
}
else if
(
 Message.message==WM_LBUTTONDOWN                 &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_BUTTON3)
)
{
 /*********************************************************
  to set up communication port number.
 *********************************************************/
 ::GetWindowRect(::GetDlgItem(hWnd,IDC_BUTTON3),&btnRect);
 ::TrackPopupMenu
 (
  menuport,
  TPM_LEFTALIGN|TPM_RIGHTBUTTON,
  btnRect.left,btnRect.bottom-1,0,
  hWnd,NULL
 );
}
else if
(
 Message.message==WM_RBUTTONDOWN &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_BUTTON3)
)
{
 /*********************************************************
  Baudrate that is used for rs232 communication port must
  configuried as 128000 when to use <b>ki-usb/gpib-controller</b>
 *********************************************************/
 ::GetWindowRect(::GetDlgItem(hWnd,IDC_BUTTON3),&btnRect);
 ::TrackPopupMenu
 (
  menuportbaud,
  TPM_LEFTALIGN|TPM_RIGHTBUTTON,
  btnRect.left,btnRect.bottom-1,0,
  hWnd,NULL
 );
}
else if
(
 Message.message==WM_LBUTTONDOWN &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_BUTTON8)
)
{
 ::GetWindowText
 (
  ::GetDlgItem(hWnd,IDC_FILENAME),File_Place_Csn,
  sizeof(File_Place_Csn)
 );
 CheckPower(hWnd,hComm,IDC_EDIT1);
}
else if
(
 Message.message==WM_LBUTTONDOWN          &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_VCAL)
)
{
 VCal(hWnd,hComm,IDC_EDIT1);
}
else if
(
 Message.message==WM_LBUTTONDOWN          &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_ICAL)
)
{
 ICal(hWnd,hComm,IDC_EDIT1);
}
else if
(
 Message.message==WM_RBUTTONDOWN            &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_SELOUTP)
)
{
 /*******************************************************
  programs will designate channel no of Hp662xa which is
  adjusted by pc through <b>ki-usb/gpib-controller</b>
 *******************************************************/
 ::GetWindowText
 (
  ::GetDlgItem(hWnd,IDC_SELOUTP),buf,sizeof(buf)
 );
 wsprintf(channelno,"%s",buf+strlen("662x"));
 wsprintf(modelno,"%s",buf);modelno[strlen("662x")]=0;
 if
 (
  strcmp(modelno,"6621")&&
  strcmp(modelno,"6622")&&
  strcmp(modelno,"6623")&&
  strcmp(modelno,"6624")&&
  strcmp(modelno,"6627")
 )
 {
  wsprintf(modelno,"6623");
  model=6623;
 }
 if
 (
  !strcmp(modelno,"6621")||
  !strcmp(modelno,"6622")
 )
 {
  if(!strcmp(channelno,"Ch1"))
  {
   wsprintf(channelno,"Ch2");
   channel=2;
  }
  else
  {
   wsprintf(channelno,"Ch1");
   channel=1;
  }
 }
 else if
 (
  !strcmp(modelno,"6624")||
  !strcmp(modelno,"6627")
 )
 {
  if(!strcmp(channelno,"Ch1"))
  {
   wsprintf(channelno,"Ch2");
   channel=2;
  }
  else if(!strcmp(channelno,"Ch2"))
  {
   wsprintf(channelno,"Ch3");
   channel=3;
  }
  else if(!strcmp(channelno,"Ch3"))
  {
   wsprintf(channelno,"Ch4");
   channel=4;
  }
  else
  {
   wsprintf(channelno,"Ch1");
```

```
   channel=1;
  }
 }
 else if
 (
  !strcmp(modelno,"6623")
 )
 {
  if(!strcmp(channelno,"Ch1"))
  {
   wsprintf(channelno,"Ch2");
   channel=2;
  }
  else if(!strcmp(channelno,"Ch2"))
  {
   wsprintf(channelno,"Ch3");
   channel=3;
  }
  else
  {
   wsprintf(channelno,"Ch1");
   channel=1;
  }
 }
 wsprintf(buf,"%s%s",modelno,channelno);
 ::SetWindowText(::GetDlgItem(hWnd,IDC_SELOUTP),buf);
 SetProReg(irtool,"modelchannel",buf);
}
else if
(
 Message.message==WM_LBUTTONDOWN            &&
 Message.hwnd==::GetDlgItem(hWnd,IDC_SELOUTP)
)
{
 /*******************************************************
  The programs make designation of model-no that is power
  -supplier(Hp662xa) and also calibrated by pc through
  GPIB-CONTROLLER.
 *******************************************************/
 ::GetWindowText
 (
  ::GetDlgItem(hWnd,IDC_SELOUTP),buf,sizeof(buf)
 );
 wsprintf(channelno,"%s",buf+strlen("662x"));
 wsprintf(modelno,"%s",buf);modelno[strlen("662x")]=0;
 if(!strcmp(modelno,"6621"))
 {
  wsprintf(modelno,"6622");
  model=6622;
 }
```

```
else if(!strcmp(modelno,"6622"))
{
 wsprintf(modelno,"6623");
 model=6623;
}
else if(!strcmp(modelno,"6623"))
{
 wsprintf(modelno,"6624");
 model=6624;
}
else if(!strcmp(modelno,"6624"))
{
 wsprintf(modelno,"6627");
 model=6627;
}
else if(!strcmp(modelno,"6627"))
{
 wsprintf(modelno,"6621");
 model=6621;
}
else
{
 wsprintf(modelno,"6623");
 model=6623;
}
if
(
 !strcmp(modelno,"6621")||
 !strcmp(modelno,"6622")
)
{
 if
 (
  strcmp(channelno,"Ch1")&&
  strcmp(channelno,"Ch2")
 )
 {
  wsprintf(channelno,"Ch1");
  channel=1;
 }
}
else if
(
 !strcmp(modelno,"6624")||
 !strcmp(modelno,"6627")
)
{
 if
 (
  strcmp(channelno,"Ch1")&&
    strcmp(channelno,"Ch2")&&
    strcmp(channelno,"Ch3")&&
    strcmp(channelno,"Ch4")
   )
   {
    wsprintf(channelno,"Ch1");
    channel=1;
   }
  }
  else if
  (
   !strcmp(modelno,"6623")
  )
  {
   if
   (
    strcmp(channelno,"Ch1")&&
    strcmp(channelno,"Ch2")&&
    strcmp(channelno,"Ch3")
   )
   {
    wsprintf(channelno,"Ch1");
    channel=1;
   }
  }
  wsprintf(buf,"%s%s",modelno,channelno);
  ::SetWindowText(::GetDlgItem(hWnd,IDC_SELOUTP),buf);
  SetProReg(irtool,"modelchannel",buf);
 }
 else if
 (
  Message.message==WM_LBUTTONDOWN              &&
  Message.hwnd==::GetDlgItem(hWnd,IDC_OUTFILE)
 )
 {
  /*******************************************************
   to set up file name and path of the testing report
   by calling subprogram "SelOutFile"
  *******************************************************/
  SelOutFile(hWnd);
 }
 else
 {
  ::TranslateMessage(&Message);::DispatchMessage(&Message);
  if(haccel!=NULL)
  {
   (::TranslateAccelerator(hWnd,haccel,&Message));
  }
 }
}
```

```
  }
 PurgeComm(hComm, PURGE_RXCLEAR);
 ::SetCommMask(hComm, 0L);
 if(hComm!=INVALID_HANDLE_VALUE)::CloseHandle(hComm);
 return FALSE;
}
case WM_ERASEBKGND :
{
 ::GetClientRect(hWnd,&btnRect);
 FillRect((HDC)wParam,&btnRect,hbr);
 return TRUE;
}
case WM_CTLCOLORDLG:
{
 return (LRESULT)hbr;
}
case WM_CTLCOLOREDIT:
{
 if((HWND)lParam==::GetDlgItem(hWnd,IDC_EDIT1))
 {
  ::SetTextColor((HDC)wParam,(COLORREF)RGB(0,0,0));
  ::SetBkColor((HDC)wParam,(COLORREF)0x00ffff80);
  return (LRESULT)hbrushbtn;
 }
 else
 {
  return NULL;
 }
}
case WM_CTLCOLORBTN:
{
 if((HWND)lParam==::GetDlgItem(hWnd,IDC_BUTTON3))
 {
  return (LRESULT)hbrushbtn;
 }
 else
 {
  return NULL;
 }
}
case WM_CTLCOLORLISTBOX:
{
 return (LRESULT)hbr;
}
case WM_CTLCOLORSTATIC:
{
 if((HWND)lParam==::GetDlgItem(hWnd,IDC_CMDFILESEL))
 {
  ::SetTextColor((HDC)wParam,(COLORREF)RGB(0,0,0));
  ::SetBkColor((HDC)wParam,(COLORREF)(63355));
```

```
    return (LRESULT)hbrushstatic;
   }
   else
   {
    return NULL;
   }
  }
  case WM_COMMAND:
  {
   if
   (
    wParam >= IDM_COMPORT1 && wParam <= IDM_COMPORT256
   )
   {
    GetMenuString
    (
     menuport,wParam,buf,60,MF_BYCOMMAND
    );
    wsprintf(buf+strlen(buf),":128000,n,8,1");
    SetProReg(irtool,"COMM",buf);
    InitCommPort(hWnd,buf);
    return TRUE;
   }
   else if
   (
    wParam >= IDM_BAUD2400 && wParam <= IDM_BAUD256000
   )
   {
    GetMenuString(menuportbaud,wParam,buf,60,MF_BYCOMMAND);
    GetProReg(irtool,"COMM",text,"COM3:128000,N,8,1");
    sptr=strstr(text,":");ptr=strstr(text,",");
    wsprintf(CurDir1,"%s",ptr);
    *sptr=0;wsprintf(textbuf,"%s:%s%s",text,buf,CurDir1);
    SetProReg(irtool,"COMM",textbuf);
    InitCommPort(hWnd,textbuf);
   }
  }
 }
 return FALSE;
}
/************************************************************
 subprogram for entry point of main-window
************************************************************/
int APIENTRY WinMain
(
 HINSTANCE hInstance,
 HINSTANCE hPrevInstance,
 LPSTR     lpCmdLine,
 int       nCmdShow
)
```

```
{
 hInst = hInstance;
 /*********************************************************
  to create main-window and initialie the main-windows
  procedure by calling subprogram DialogBox
 *********************************************************/
 return
 (
  DialogBox(hInst, (LPCTSTR)IDD_HP662XACAL,
  NULL, (DLGPROC)WindowsProc)
 );
}
```