CF・CIF-AP 系列可程式交流電源供應器 GPIB , RS-232 介面指令說明

REVISION 1.0

FILE:PSCPI02.

PN:800-01007

2003/5/3

目錄

頁次

Chapter I. Programable AC Source SCPI: & IEEE488.2 Command

IEEE488.2 Interface

Specification : Standard IEE488.2

Function :

1.SH1: Full Source Handshake
2.Ah1: Full Acceptor Handshake
3.T6: Basic Talker
4.L4: Basic Listener
5.SR0: Without Service Reguest
6.RL1: Remote/Local Change
7.PR0: Without Parallel Polling
8.DC1: Device Clear
9.DT0: Without Device Trigger
10.C0: Without Controllen function

Command :

*CLS Clear the status byte summary register and all event registers .
*ESE <Enable Value>
Enable bits in the standard Event status enable register.
*ESE? Query standard Event ebable register.
*ESR? Query standard Event register.
*IDN Identification query.
*OPC Sets the “operation complete” bit(bit 0) in the Standard Event.
*OPC Return “1” to the output buffer after the command is executed.
*RST Reset the instrucment to its power-on configuration.
*SRE <Enable Value>
Enable bits in the Status Byte enable register.
*SRE? Query the Status Byte enable register.
*STB? Read status byte query.
*TST? Perform a complete self-test of instrument .
Return “0” if the self-test is successful , or “1” if it test fails.

SCPI Command

OUTPut

:OUTPut {ON/OFF/1/0}

:OUTPut:STATe {ON/OFF/1/0}

Sets the output of the AC source.

SOURce

:SOURce:FREQuency <NRf>

Sets or query the output signal frequency.

:SOURce:RANGe {HIGH|LOW|AUTO}

Sets or query the output range.

:SOURce:SPPHase <NRf>

Sets or query the stop phase.(Only CF-SERIES)

:SOURce:STPHase <NRf>

Sets or query Start phase.(Only CF-SERIES)

:SOURce:TURN {ON/OFF/1/0}

Sets or Query the output signal status.

:SOURce:VOLTage <NRf>

Sets or query the output voltage.

STATus

:PRESent

Clears the Questionable status register.both the event and enable registers are cleared.

:QUEStionable:CONDition?

Query the contents of condition register of Questionable status register group.

:QUEStionable:ENABle

Sets or queries the enable register of Questionable status register group.

:QUEStionable[:EVENt]?

Query the contents of event register of Questionable status register group.

SYSTem

:SYSTem:BEEPer

:SYSTem:BEEPer:IMMediate

:SYSTem:BEEPer:STATe {ON/OFF/1/0}

Sets the beeper to ON/OFF,queries the current setting.

:SYSTem:ERRor? (Query Only)

Queries the occurred error code and message.

:SYSTem:KLOCk

Sets or queries whether the front-panel keys are locked.

:SYSTem:PRESet(No Query)

Resets to the default state.

:SYSTem:VERSion

Queries the value corresponding to the SCPI version to which the instrucment complies

RS-232 Interface

Mode : Start stop synchronization

Baud rate : 1200.2400.4800.9600bps

Command :

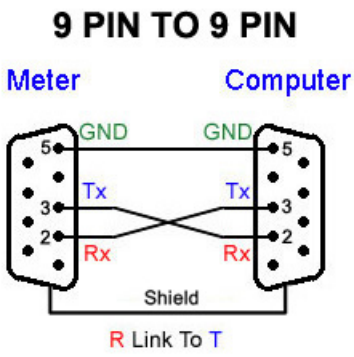
:SYSTem:LOCal(No Query)

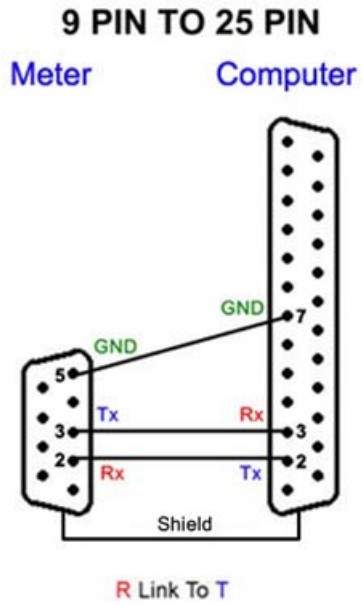
Sets the system to local control.

:SYSTem:REMote(No Query)

Sets the system to remote control.

RS-232 PC To CF/CIF AC SOURCE Connection





SCPI Status System

Questionable Data

Event Register .or. Enable Register

Bit	Condition	
10	FAULT	:System Fault.
9	FUSE	:Fuse Break.
8	ODP	: Over Drive Protection.
3	OTP	:Over Temperature Protection.
1	OLP	:Over Load Protection.

Standard Event

Event Register .or. Enable Register

Bit	Conditon
7	Not used
6	Not used
5	Command Error
4	Not used
3	Not used
2	Query Error
1	Not used
0	Operation Complete

Status Byte

Summary Register .or. Enable Register

Bit	Conditon
7	Operation Data
6	Request Service
5	Standard Event
4	Message Avilable
3	Questionable Data
2	
1	
0	

Chapter II. Communication-related Error Messages:

Code	Message	Code	Message
0	No Error		
-100	Command error	-330	Self-test failed
-101	Invalid character	-350	Queue overflow
-102	Syntax error	-400	Query errors
-103	Invalid separator	-410	Query INTERRUPTED
-104	Data type error	-420	Query UNTERMINATED
-105	GET not allowed	-430	Query DEADLOCKED
-108	Parameter not allowed	-440	Query UNTERMINATED after indefinite response
-109	Missing parameter		
-112	Program mnemonic too long	60	ROM FAILED"
-113	Undefined header	61	RAM FAILED"
-121	Invalid character in number	62	EEPROM R/W FAILED"
-123	Exponent too large	63	USER SETTING LOST"
-124	too many digits	64	Command allowed only with RS-232
-128	Numeric data not allowed	65	Command not allowed in local
-131	Invaild suffix	100	FAULT
-138	Suffix not allowed	101	OLP
-140	Character data error	102	OTP
-141	Invalid character data	103	ODP
-144	Character data too long	104	FUSE
-148	Character data not allowed		
-150	String data error		
-151	Invalid string data		
-158	String data not allowed		
-160	Block data error		
-161	Invalid block data		
-168	Block data not allowed		
-170	Expression error		
-171	Invalid expression		
-178	Expression data not allowed		
-200	Execution errors		
-211	Trigger ignored		
-213	Init ignored		
-221	Setting conflict		
-222	Data out of rang		
-223	Too much data		
-224	Illegal parameter valve		
-230	Data corrupt or stale		
-241	Hardware missing		
-310	System error		
-311	Memory error		
-313	Calibration memory lost		

ChapterⅢ. RS232 鮑率與 GPIB 位址設定法：

1. 關閉電源開關(POWER)
2. 頻率指撥開關撥於 0069
3. 按著 RESET 開關同時開啓電源開關(POWER)
4. 以上步驟若操作正確則 REMOT 燈亮，表示已進入設定模式
5. RS232 鮑率設定

a.指撥開關與鮑率關係如下：

0100：1200
0101：2400
0102：4800
0103：9600

b.依上列所示選定所需鮑率來設定指撥開關，如：9600 爲 0103
c.按 RESET 鍵完成**輸入**動作
d.撥指撥開關於 0300 位置
e.按 RESET 鍵完成**儲存**動作，並跳出設定模式進入正常操作模式
f.將指撥開關撥於所需輸出電源頻率，如：50.00

6. GPIB 位址設定：

a. **指撥開關與位址關係如下：**

0200：00
0201：01
0231：31

b. 依上列所示選定所需位址，來設定指撥開關，如 0206 表示設定位址爲 06
c.按 RESET 鍵完成**輸入**動作
d.調指撥開關於 0300 位置
e.按 RESET 鍵完成**儲存**動作，並跳出設定模式進入正常操作模式
f.將指撥開關設於所需輸出電源頻率，如：50.00

註：GPIB 內定 Address 爲 06
RS232 內定鮑率爲 9600

ChapterⅣ. Example Program :

EX1: CFxx Series GPIB demo Program

```
REM Example Program using NI-488 board level functions
REM QBDECL.BAS contains constants, declarations, and subroutine prototypes.
REM $INCLUDE: 'qbdecl.bas'
REM GPIBERR is an error subroutine that is called when a NI-488 function fails.
REM DVMERR is an error subroutine that is called when the IDRC CP-600 POWER
REM ANALYZER does not have valid data to send.
    DECLARE SUB gpiberr (msg$)
    DECLARE SUB dvmerr (msg$, rd$)
    CLS
    LOCATE 2, 3
    PRINT "CFxx Series demo Program ....."
    PRINT
    bdname$ = "GPIB0"
    CALL ibfind(bdname$, brd0%)
    IF brd0% < 0 THEN CALL gpiberr("Ibfind Error")
REM Send the Interface Clear (IFC) message.
    CALL ibsic(brd0%)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibsic Error")
REM Turn on the Remote Enable (REN) signal.
    CALL ibsre(brd0%, 1)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibsre Error")
REM Inhibit front panel control with the Local Lockout (LLO) command (hex 11).
REM Place the IDRC CP-600 in remote mode by addressing it to listen (ASCII "&").
REM Send the Device Clear (DCL) message to clear device (hex 14).
REM Address the GPIB interface board to talk (hex 40 or ASCII "@").
    cmd$ = CHR$(&H11) + "&" + CHR$(&H14) + "@"
    CALL ibcmd(brd0%, cmd$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibcmd Error")
REM set range to 150V
    wrt$ = ":SOUR:RANG LOW;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
REM set voltage to 100V
    wrt$ = ":SOUR:VOLT 100;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
REM set frequency to 60Hz
    wrt$ = ":SOUR:FREQ 60;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
REM set start phase
    wrt$ = ":SOUR:STPH   0;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")

REM set stop phase
    wrt$ = ":SOUR:SPPH   0;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")

REM output relay on
    wrt$ = ":OUTP ON;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")

REM turn on signal
    wrt$ = ":SOUR:TURN ON;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
    SLEEP (3)

REM set voltage to 120V
    wrt$ = ":SOUR:VOLT 120;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
   SLEEP (3)

REM set frequency to 50Hz
    wrt$ = ":SOUR:FREQ 50;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
    SLEEP (3)
```

```
REM set voltage to 90V
    wrt$ = ":SOUR:VOLT 90;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
    SLEEP (3)

REM turn on signal
    wrt$ = ":SOUR:TURN OFF;"
    CALL ibwrt(brd0%, wrt$)
    IF (ibsta% AND EERR) THEN CALL gpiberr("Ibwrt Error")
    SLEEP (3)

REM Call the IBONL function to disable the hardware and software.
    CALL ibonl(brd0%, 0)
END

SUB dvmerr (msg$, rd$) STATIC

    PRINT msg$

    PRINT "Status Byte = "; rd$
REM Call the IBONL function to disable the hardware and software.
    CALL ibonl(brd0%, 0)
    STOP
END SUB

SUB gpiberr (msg$) STATIC
    PRINT msg$
    PRINT "ibsta = &H"; HEX$(ibsta%); " <";
    IF ibsta% AND EERR THEN PRINT " ERR";
    IF ibsta% AND TIMO THEN PRINT " TIMO";
    IF ibsta% AND EEND THEN PRINT " END";
    IF ibsta% AND SRQI THEN PRINT " SRQI";
    IF ibsta% AND RQS THEN PRINT " RQS";
    IF ibsta% AND SPOLL THEN PRINT " SPOLL";
    IF ibsta% AND EEVENT THEN PRINT " EVENT";
    IF ibsta% AND CMPL THEN PRINT " CMPL";
    IF ibsta% AND LOK THEN PRINT " LOK";
    IF ibsta% AND RREM THEN PRINT " REM";
    IF ibsta% AND CIC THEN PRINT " CIC";
    IF ibsta% AND AATN THEN PRINT " ATN";
    IF ibsta% AND TACS THEN PRINT " TACS";
    IF ibsta% AND LACS THEN PRINT " LACS";
    IF ibsta% AND DTAS THEN PRINT " DTAS";
    IF ibsta% AND DCAS THEN PRINT " DCAS";
    PRINT " >"
    PRINT "iberr = "; iberr%;
    IF iberr% = EDVR THEN PRINT " EDVR <DOS Error>"
    IF iberr% = ECIC THEN PRINT " ECIC <Not CIC>"
    IF iberr% = ENOL THEN PRINT " ENOL <No Listener>"
    IF iberr% = EADR THEN PRINT " EADR <Address error>"
    IF iberr% = EARG THEN PRINT " EARG <Invalid argument>"
    IF iberr% = ESAC THEN PRINT " ESAC <Not Sys Ctrlr>"
    IF iberr% = EABO THEN PRINT " EABO <Op. aborted>"
    IF iberr% = ENEB THEN PRINT " ENEB <No GPIB board>"
    IF iberr% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
    IF iberr% = ECAP THEN PRINT " ECAP <No capability>"
    IF iberr% = EFSO THEN PRINT " EFSO <File sys. error>"
    IF iberr% = EBUS THEN PRINT " EBUS <Command error>"
    IF iberr% = ESTB THEN PRINT " ESTB <Status byte lost>"
    IF iberr% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>"
    IF iberr% = ETAB THEN PRINT " ETAB <Table Overflow>"
    PRINT "ibcnt = "; ibcnt%

REM Call the IBONL function to disable the hardware and software.
    CALL ibonl(brd0%, 0)
    STOP
END SUB
```

EX2: CFxx Series GPIB demo Program

```
#include <string.h>
#include "decl.h"

void gpiberr(char *msg);
void dvmerr(char *msg, char *rd);

   char       read[512];                    /* read data buffer                  */
   int        bd,                           /* board or device number             */
              i;                            /* FOR loop counter                   */

void main() {

      clrscr();

      gotoxy(3,2);
      printf("CFxx Series demo Program .....");

      bd = ibfind ("GPIB0");
      if (bd < 0) gpiberr("ibfind Error");

/*   Send the Interface Clear (IFC) message.       */

      ibsic (bd);
      if (ibsta & ERR) gpiberr("ibsic Error");

/*   Turn on the Remote Enable (REN) signal.       */

      ibsre (bd,1);
      if (ibsta & ERR) gpiberr("ibsre Error");

/*
 *   Inhibit front panel control with the Local Lockout (LLO) command
 *   (hex 11).  Place the IDRC CP-600 in remote mode by addressing it to listen
 *   (hex 26 or ASCII "&").  Send the Device Clear (DCL) message to clear
 *   internal device functions (hex 14).  Address the GPIB interface board to
 *   talk (hex 40 or ASCII "@").
 */

      ibcmd (bd,"\021&\024@",4L);
      if (ibsta & ERR) gpiberr("ibcmd Error");

      /* set range to 150V              */
      ibwrt (bd,":SOUR:RANG LOW;", 15L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      /* set voltage to 100V            */
      ibwrt (bd,":SOUR:VOLT 100;", 15L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      /* set frequency to 60Hz          */
      ibwrt (bd,":SOUR:FREQ 60;", 14L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      /* set start phase                */
      ibwrt (bd,":SOUR:STPH  0;", 14L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      /* set stop phase                 */
      ibwrt (bd,":SOUR:SPPH  0;", 14L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      /* output relay on                */
      ibwrt (bd,":OUTP ON;", 9L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      /* turn on signal                 */
      ibwrt (bd,":SOUR:TURN ON;", 14L);
      if (ibsta & ERR) gpiberr("ibwrt Error");

      sleep(3);
      /* set voltage to 120V            */
      ibwrt (bd,":SOUR:VOLT 120;", 15L);
      if (ibsta & ERR) gpiberr("ibwrt Error");
```

```
    sleep(3);
    /* set frequency to 50Hz        */
    ibwrt (bd,":SOUR:FREQ 50;", 14L);
    if (ibsta & ERR) gpiberr("ibwrt Error");

    sleep(3);
    /* set voltage to 90V           */
    ibwrt (bd,":SOUR:VOLT 90;", 14L);
    if (ibsta & ERR) gpiberr("ibwrt Error");

    sleep(3);
    /* turn on signal               */
    ibwrt (bd,":SOUR:TURN OFF;", 15L);
    if (ibsta & ERR) gpiberr("ibwrt Error");

    sleep(3);

/*  Call the ibonl function to disable the hardware and software.       */

    ibonl (bd,0);

}

void gpiberr(char *msg) {

    printf ("%s\n", msg);

    printf ("ibsta = &H%x  <", ibsta);
    if (ibsta & ERR )  printf (" ERR");
    if (ibsta & TIMO)  printf (" TIMO");
    if (ibsta & END )  printf (" END");
    if (ibsta & SRQI)  printf (" SRQI");
    if (ibsta & RQS )  printf (" RQS");
    if (ibsta & SPOLL) printf (" SPOLL");
    if (ibsta & EVENT) printf (" EVENT");
    if (ibsta & CMPL)  printf (" CMPL");
    if (ibsta & LOK )  printf (" LOK");
    if (ibsta & REM )  printf (" REM");
    if (ibsta & CIC )  printf (" CIC");
    if (ibsta & ATN )  printf (" ATN");
    if (ibsta & TACS)  printf (" TACS");
    if (ibsta & LACS)  printf (" LACS");
    if (ibsta & DTAS)  printf (" DTAS");
    if (ibsta & DCAS)  printf (" DCAS");
    printf (" >\n");

    printf ("iberr = %d", iberr);
    if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
    if (iberr == ECIC) printf (" ECIC <Not CIC>\n");
    if (iberr == ENOL) printf (" ENOL <No Listener>\n");
    if (iberr == EADR) printf (" EADR <Address error>\n");
    if (iberr == EARG) printf (" EARG <Invalid argument>\n");
    if (iberr == ESAC) printf (" ESAC <Not Sys Ctrlr>\n");
    if (iberr == EABO) printf (" EABO <Op. aborted>\n");
    if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
    if (iberr == EOIP) printf (" EOIP <Async I/O in prg>\n");
    if (iberr == ECAP) printf (" ECAP <No capability>\n");
    if (iberr == EFSO) printf (" EFSO <File sys. error>\n");
    if (iberr == EBUS) printf (" EBUS <Command error>\n");
    if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
    if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
    if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");

    printf ("ibcnt = %d\n", ibcnt);
    printf ("\n");

/*  Call the ibonl function to disable the hardware and software.       */

    ibonl (bd,0);

    exit(1);

}

void dvmerr(char *msg,char *rd) {

    printf ("%s\n", msg);
```

```
    printf("Status byte = %x\n", rd[0]);

/*  Call the ibonl function to disable the hardware and software.       */

    ibonl (bd,0);

    exit(1);
}
```

EX3: CFxx Series RS232 demo Program

```
CLS
LOCATE 2, 3
PRINT "CFxx Series demo Program ....."
OPEN "COM1:9600,N,8,1,RS,CS0,DS0,CD0" FOR RANDOM AS #1
COM(1) ON

PRINT #1, ":SYST:REM;"          ' set device to remote mode
PRINT #1, ":SOUR:RANG LOW;"     ' set range to 150V
PRINT #1, ":SOUR:VOLT 100;"     ' set voltage to 100V
PRINT #1, ":SOUR:FREQ 60;"      ' set frequency to 60Hz
PRINT #1, ":SOUR:STPH  0;"      ' set start phase
PRINT #1, ":SOUR:SPPH  0;"      ' set stop phase
PRINT #1, ":OUTP ON;"           ' output relay on
PRINT #1, ":SOUR:TURN ON;"      ' turn on signal

SLEEP (3)
PRINT #1, ":SOUR:VOLT 120;"     ' set voltage to 120V

SLEEP (3)
PRINT #1, ":SOUR:FREQ 50;"      ' set frequency to 50Hz

SLEEP (3)
PRINT #1, ":SOUR:VOLT 90;"      ' set voltage to 90V

SLEEP (3)
PRINT #1, ":SOUR:TURN OFF;"     ' turn off signal

SLEEP (3)
PRINT #1, ":SYSTEM:LOC;"        ' set device to local mode
END
```

EX4: CFxx Series RS232 demo Program

```
#include "bios.h"
#include "conio.h"
#include "stdio.h"
#include "dos.h"

char err_no=0;

main()
{
        unsigned char readbuf[100],*p;
        int i;

        clrscr();
        gotoxy(3,2);
        printf("CFxx Series demo Program .....");

        initial_sys();                  /* initial RS-232 */

        send(":SYSTEM:REM;");           /* set device to remote mode  */


        send(":SOUR:RANG LOW;");        /* set range to 150V            */
        send(":SOUR:VOLT 100;");        /* set voltage to 100V          */
        send(":SOUR:FREQ 60;");         /* set frequency to 60Hz        */
        send(":SOUR:STPH  0;");         /* set start phase              */
        send(":SOUR:SPPH  0;");         /* set stop phase               */
        send(":OUTP ON;");              /* output relay on              */
        send(":SOUR:TURN ON;");         /* turn on signal               */

        sleep(3);
        send(":SOUR:VOLT 120;");        /* set voltage to 120V          */

        sleep(3);
        send(":SOUR:FREQ 50;");         /* set frequency to 50Hz        */

        sleep(3);
        send(":SOUR:VOLT 90;");         /* set voltage to 90V           */


        sleep(3);
        send(":SOUR:TURN OFF;");        /* turn off signal              */

        sleep(3);
        send(":SYSTEM:LOC;");           /* set device to local mode     */

 }

initial_sys()
{
        outportb(0x3fb,0x80);
        outportb(0x3f8,0x0c);
        outportb(0x3f9,0x00);
        outportb(0x3fb,0x07);
        outportb(0x3fb,0x03);
};

send(unsigned char *p)
{
        unsigned status;
        int         i,j,k;

        j=strlen(p);
        for(i=0;i<j;i++)
                {
                for(k=0;k<10000;k++)
                        {
                         status=inportb(0x3fd);
                         if(status & 0x20)
                                {
                                 outportb(0x3f8,*p);
                                 p++;
                                 break;
                                 }
```

```
                        }

                if(k==10000)
                        {
                         err_no=1;
                         gotoxy(3,22);
                         printf("RS232 sending timeout.....");
                         break;
                         }
                }
}

read(unsigned char *p)
{
        unsigned char status,over;
        int         i,j;
        long int    k;

        over=0;
        while(1)
                {
                for(k=0;k<2000000;k++)
                        {
                         status=inportb(0x3fd);
                         if(status & 0x01)
                                {
                                 *p=inportb(0x3f8);
                                 if(*p==0x0a)
                                     {
                                      *++p=0;
                                      over=1;
                                      }

                                 p++;
                                 break;
                                 }
                        }
                if(over==1)
                        break;
                if(k==2000000)
                        {
                         err_no=1;
                         gotoxy(3,22);
                         printf("RS232 reading timeout.....\n");
                         break;
                         }
                }
}
```