CIF 系列
附 GPIB、RS232 三相控制器
操 作 手 冊

REVISION:1.1

FILE:CPC3-CIF

PN:800-01024

2005/03/31

目录：

附录：

壹.电气规格

本三相控制器是由数字方式产生之具频率稳定；振幅稳定，失真率低，温度漂移少，....等优良性能之正弦波三相信号产生器，是目前三相交流电源供应器信号来源之最佳选择。

规格：

1. 输入电源　：AC 110V/220V ±10％　50/60Hz。

2. 输出振幅　：(1)SIN 1500mV(Rms)　RL=10KΩ。
(2)稳定度：±0.02％。

3. 输出频率　：＿＿＿＿～＿＿＿＿ ± ＿0.01＿Hz。

4. 波形失真　：≦0.2％。

5. 温度系数　：100PPM。

6. 相位差　　：120 度 ± 1 度。

7. 具缓激活装置，电源投入时振幅由小而大，无突波现象。

8. 具 RS-232，GPIB 接口，可由计算机联机控制。

9. 联机指令请参考界面指令说明书。

注意：使用本机前，请先详阅操作程序说明。

贰.面板说明：(请参照 FIG 2-1)

(1)本机电源开关 ：按 I 为开，按 O 为关。

(2)电源指示灯 ：当电源开关开启时，指示灯则亮起。

(3)输出控制按钮 ：按一次为开,再按一次为关;开机时为关状态。

(4)主机输出电磁开关指示灯 ：亮为开，不亮为关。

(5)本机输出信号振幅粗调 ：左转到底可调至 OV，右转到底可调至 1500mV(Rms)。

(6)本机输出信号振幅细调 ：细部微调。

(7)“C” 相前板输出 BNC 插座，最大输出：SIN 1500mV(Rms) RL=10KΩ。

(8)“B” 相前板输出 BNC 插座，最大输出：SIN 1500mV(Rms) RL=10KΩ。

(9)“A” 相前板输出 BNC 插座，最大输出：SIN 1500mV(Rms) RL=10KΩ。

(10)复归按钮开关 ：当机器在异常情况排除后复归按钮(机器异常时亮红灯,且有警示声)

(11)RANGE ：输出电压文件位选择(HI or LOW)

(12)输出信号之频率设定指拨开关

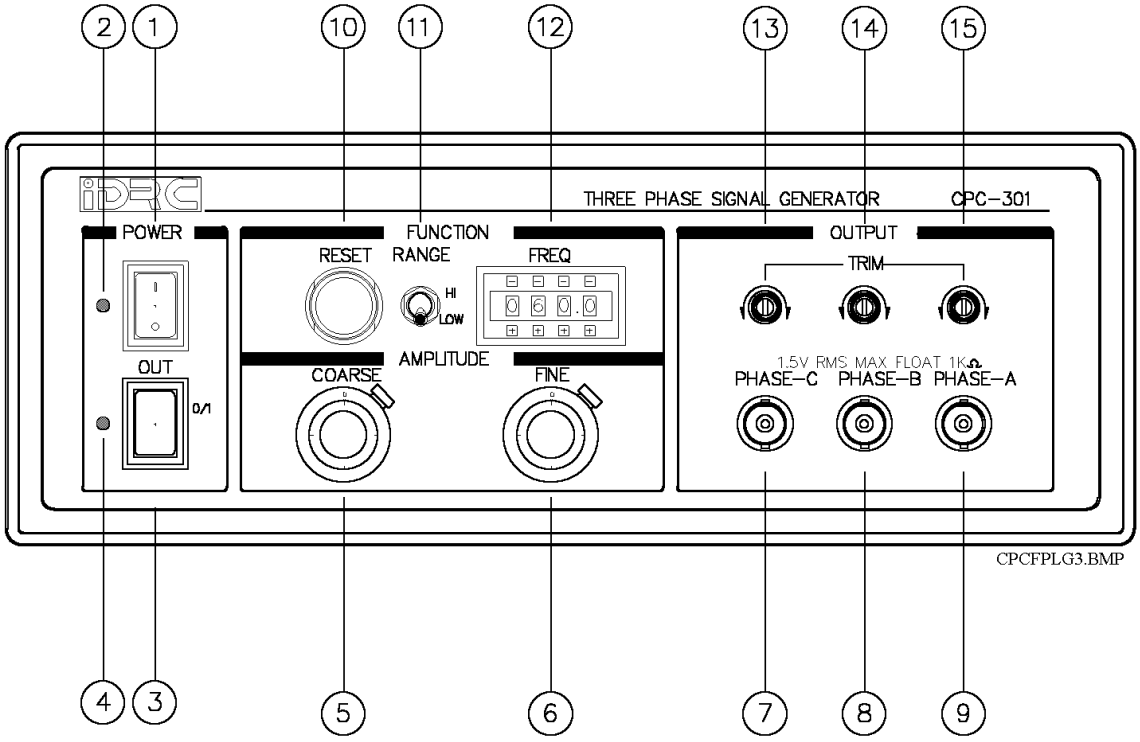
(13) “C” 相输出振幅微调 VR，可调±20％。

(14) “B” 相输出振幅微调 VR，可调±20％。

(15) “A” 相输出振幅微调 VR，可调±20％。

注：(1)各相间信号相差 120 度。
(2)前板及背板 BNC 座是并连连接。

(FIG 2-1)


2
1
10
11
12
13
14
15
THREE PHASE SIGNAL GENERATOR
CPC-301
POWER
OUT
0/1
RESET
FUNCTION
RANGE
HI
LOW
FREQ
0600
AMPLITUDE
COARSE
FINE
OUTPUT
TRIM
1.5V RMS MAX FLOAT 1K
PHASE-C
PHASE-B
PHASE-A
CPCFPLG3.BMP
4
3
5
6
7
8
9

参.背板说明：(请参照 FIG 3-1)

(1)“A” 相背板输出 BNC 插座，最大输出：SIN 1500mV(Rms) RL=10KΩ。

(2)“B” 相背板输出 BNC 插座，最大输出：SIN 1500mV(Rms) RL=10KΩ。

(3)“C” 相背板输出 BNC 插座，最大输出：SIN 1500mV(Rms) RL=10KΩ。

(4)接地端子 ：本机外壳接地端子。

(5)本机电源保险丝座 ：保险丝容量为 110V 1A、220V 0.5A 速断型。

(6)电源电压选择开关 ：电源电压选择开关。

(7)本机电源插座 ：请接至适合本机电压电源插座。

(8)A 相控制线插座 ：9PIN D 型接头(请勿与 RS232 混用)

(9)B 相控制线插座 ：9PIN D 型接头(请勿与 RS232 混用)

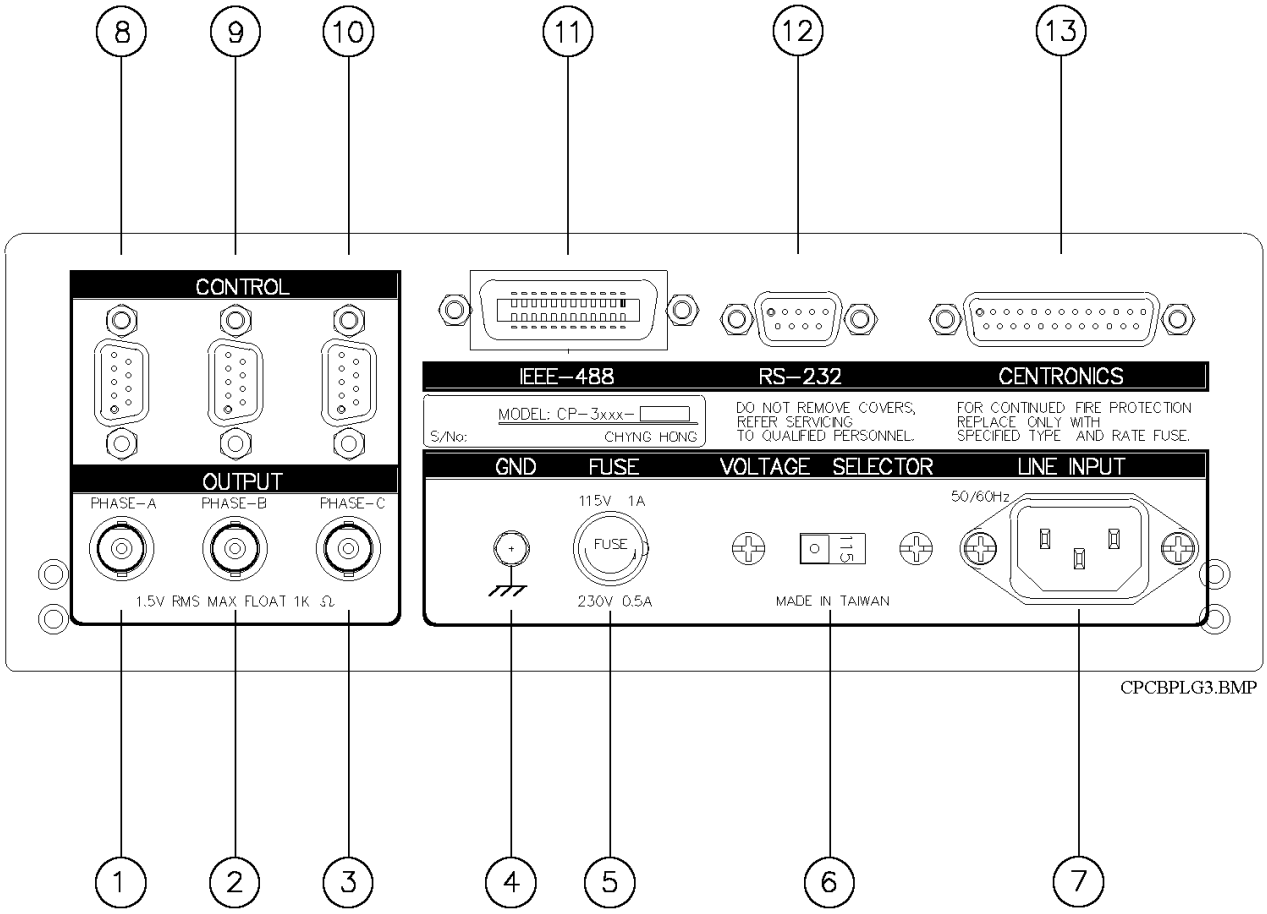
(10)C 相控制线插座 ：9PIN D 型接头(请勿与 RS232 混用)

(11) GPIB 插座 ：IEEE-488 GPIB 插座

(12) RS232 插座 ：RS232 插座(请勿与各相控制线混用)

(13) 预留插座

(FIG 3-1)



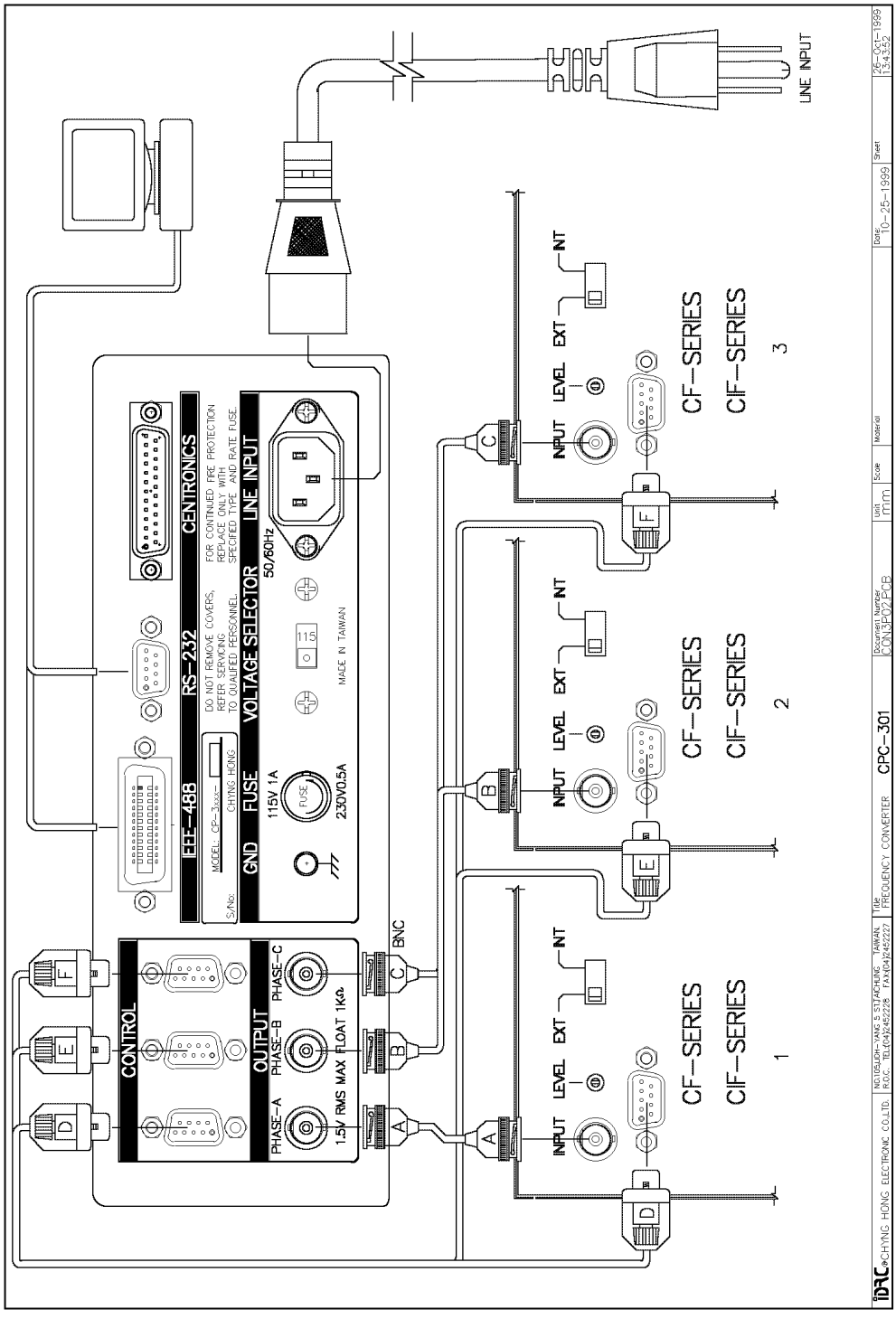
CPCBPLG3.BMP

肆.信号接线图说明：(请参照 FIG 4-1)

(1)三相控制器下方为 AC POWER SOURCE，即交流电源供应器之后板信号输入端部份。

(2)连接时，请将 AC POWER SOURCE 背板开关切至 EXT 处，连结之信号线请用 BNC 插头之单隔离线。

(FIG 4-1)



伍.操作程序

1. 将所有电源及输出开关置于 OFF 状态。

2. 检查 AC SOURCE 后板外部输入信号选择开关是否切至 EXT 上。

3. 将输出端子台 3Ø 短路线需短路。(将 N 极全部接在一起)

4. 确认本机输入 3Ø220V 电源输入开关后，接上电源。

5. 开启本机总电源输入开关。

6. 先将 CPC-301(三相控制器)电源开启。

7. 再开启每一台电源无溶丝开关；而后打开电源开关，之后再开启 OUTPUT 开关。

8. 确认每一台电压文件位 RANGE 无误，若大于 150V 请选择 300V 檔。

9. 调整 CPC-301(三相控制器)V-ADJ，并检视各 AC SOURCE 表头显示电压，约过 15V 后 AC SOURCE 各表头频率开始显示，此三台 AC SOURCE 显示之频率应一致。

10. 调至所需电压时，若三台电压不一致，请用小螺丝起子调整 CPC-301(三相控制器)面板 LEVEL 微调 VR，使三相输出电压一致。

11. 接上负载后，将每一台输出无熔丝关打开；每单一台开启后再开启本机总输出电源开关

12. 关闭时先将受测物及输出端子座间之无熔丝开关关闭，再关掉所有机器之电源。

注意：为避免在开关各 AC SOURCE 之电源，而发生欠相问题；所以请注意主输出无熔丝开关之开关时机。

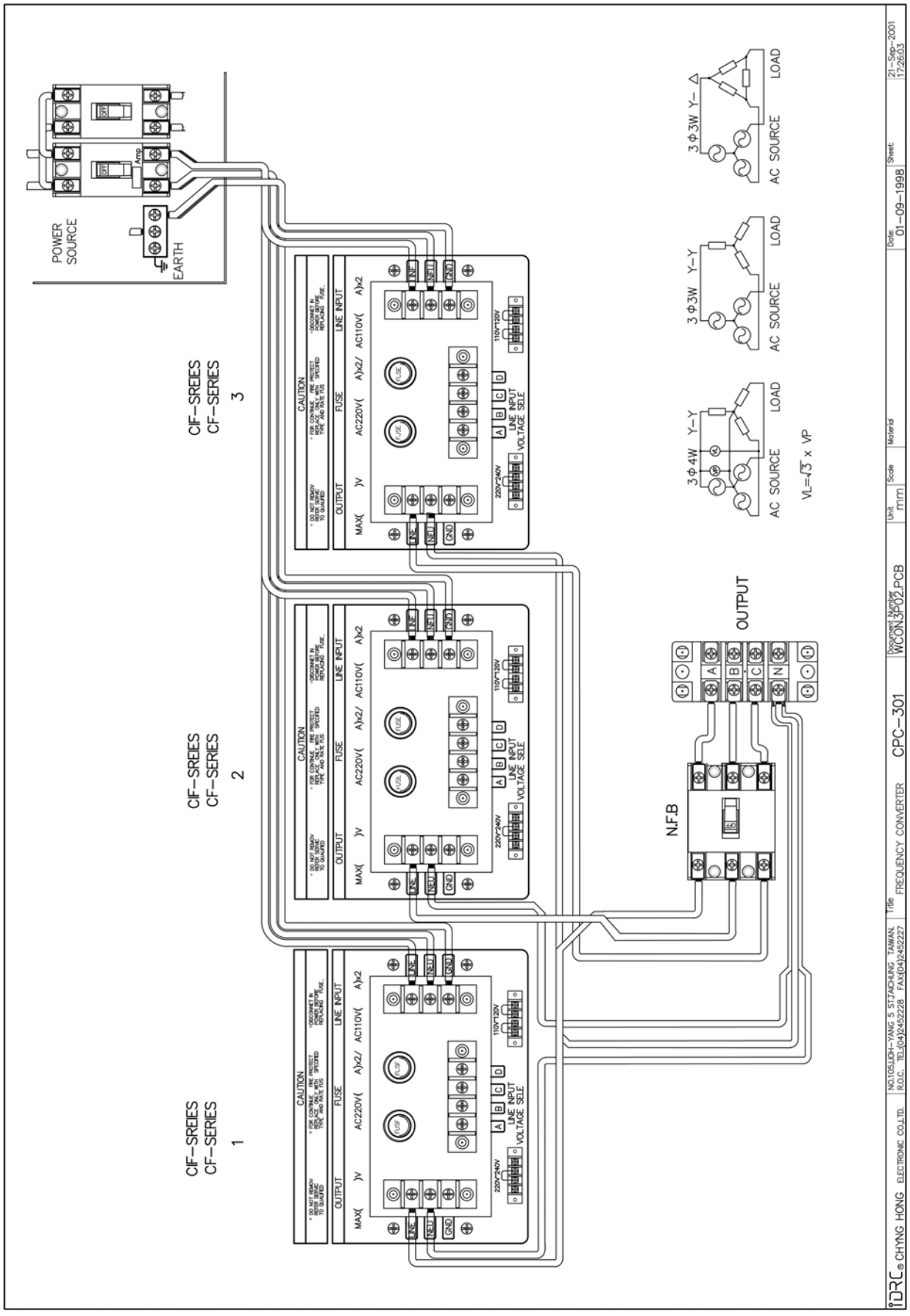
陆.三相控制器和交流电源供应器应用之电源及输出接线图：(请参照 FIG 6-1~FIG 6-6)

1.电源线及输出线线径请参考附录壹、附录贰配置。

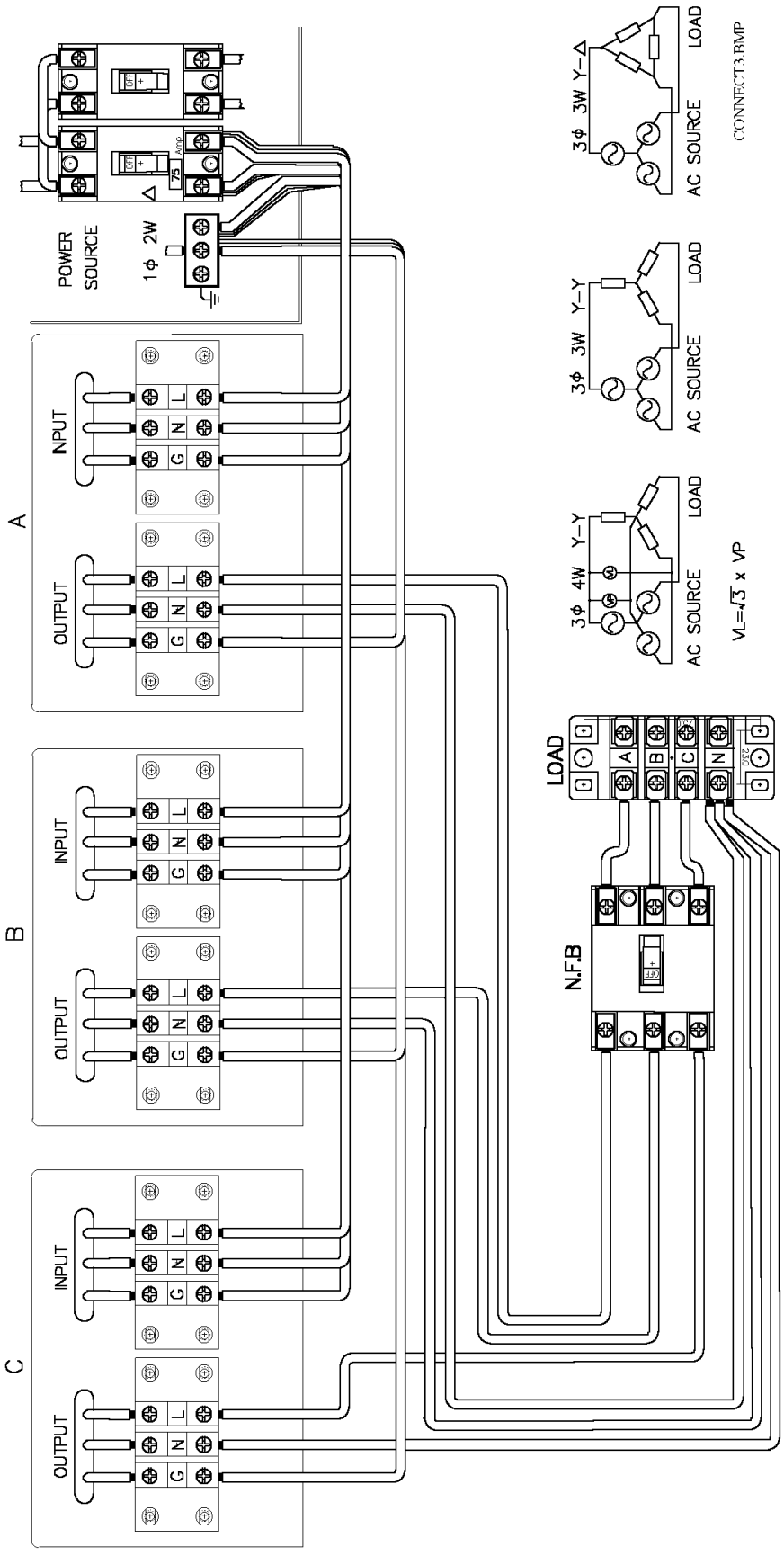
2.输出及受测产品间应加三相无熔丝开关做测试电源之关闭或启用（参考 ☐ FIG 6-1 ☐ FIG 6-2 ☐ FIG 6-3 ☐ FIG 6-4 ☐ FIG 6-5 ☐ FIG 6-6）。

3.确认配线无误及电源电压无误后再送电。

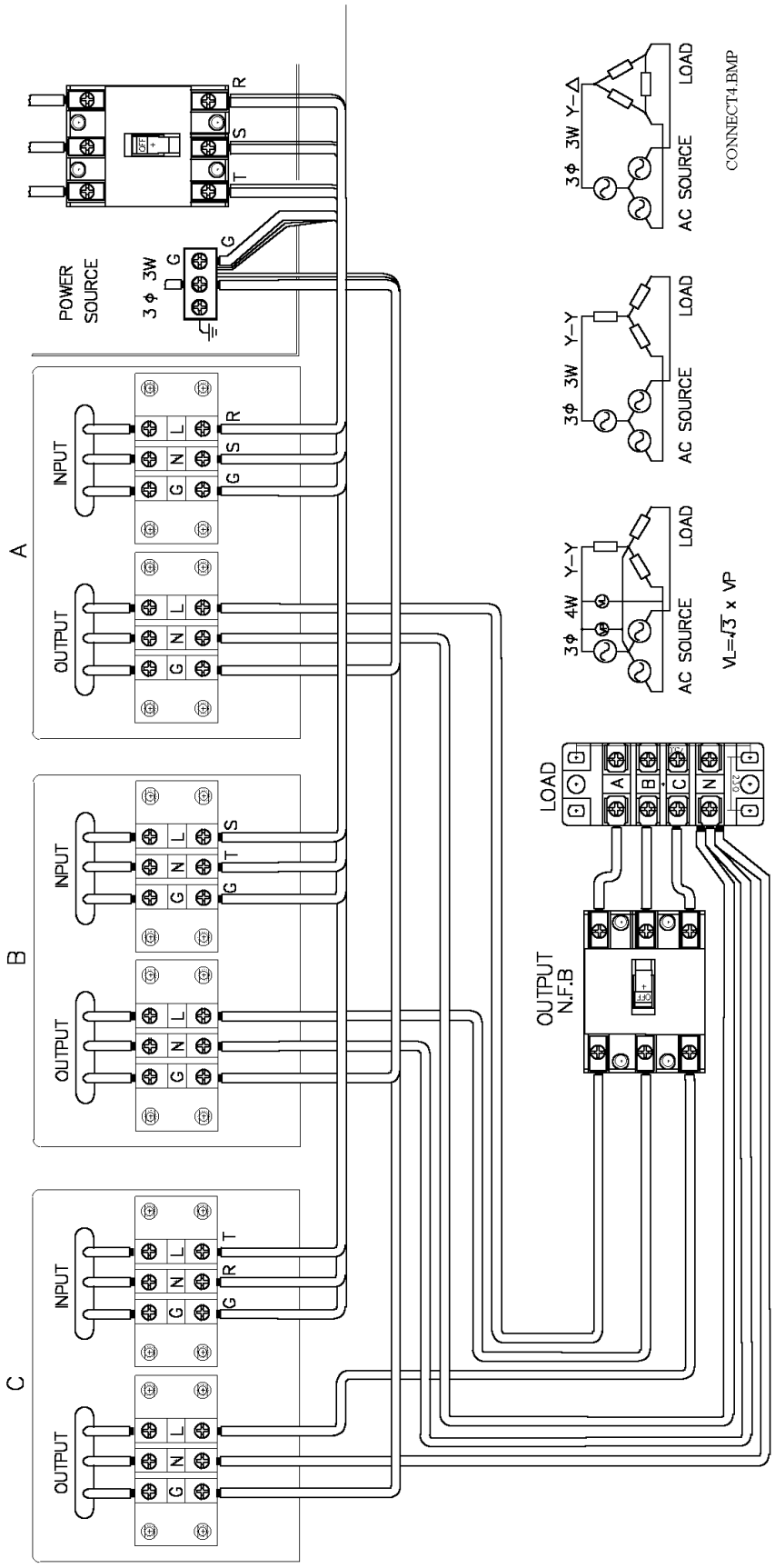
(FIG 6-1)


POWER SOURCE
EARTH
CIF-SREIES
CF-SERIES
1
CIF-SREIES
CF-SERIES
2
CIF-SREIES
CF-SERIES
3
N.F.B
OUTPUT
3 Φ 4W Y-Y
AC SOURCE
LOAD
3 Φ 3W Y-Y
AC SOURCE
LOAD
3 Φ 3W Y- △
AC SOURCE
LOAD
VL=√3 x VP
CHYNG HONG ELECTRONIC CO.,LTD.
NO.105,JUDH-YANG 5 ST.,TACHUNG TAIWAN, R.O.C. TEL:(04)2452228 FAX(04)2452227
Title FREQUENCY CONVERTER CPC-301
Document Number WCON3P02.PCB
Unit mm
Scale
Material
Date: 01-09-1998
Sheet:
21-Sep-2001 17:26:03

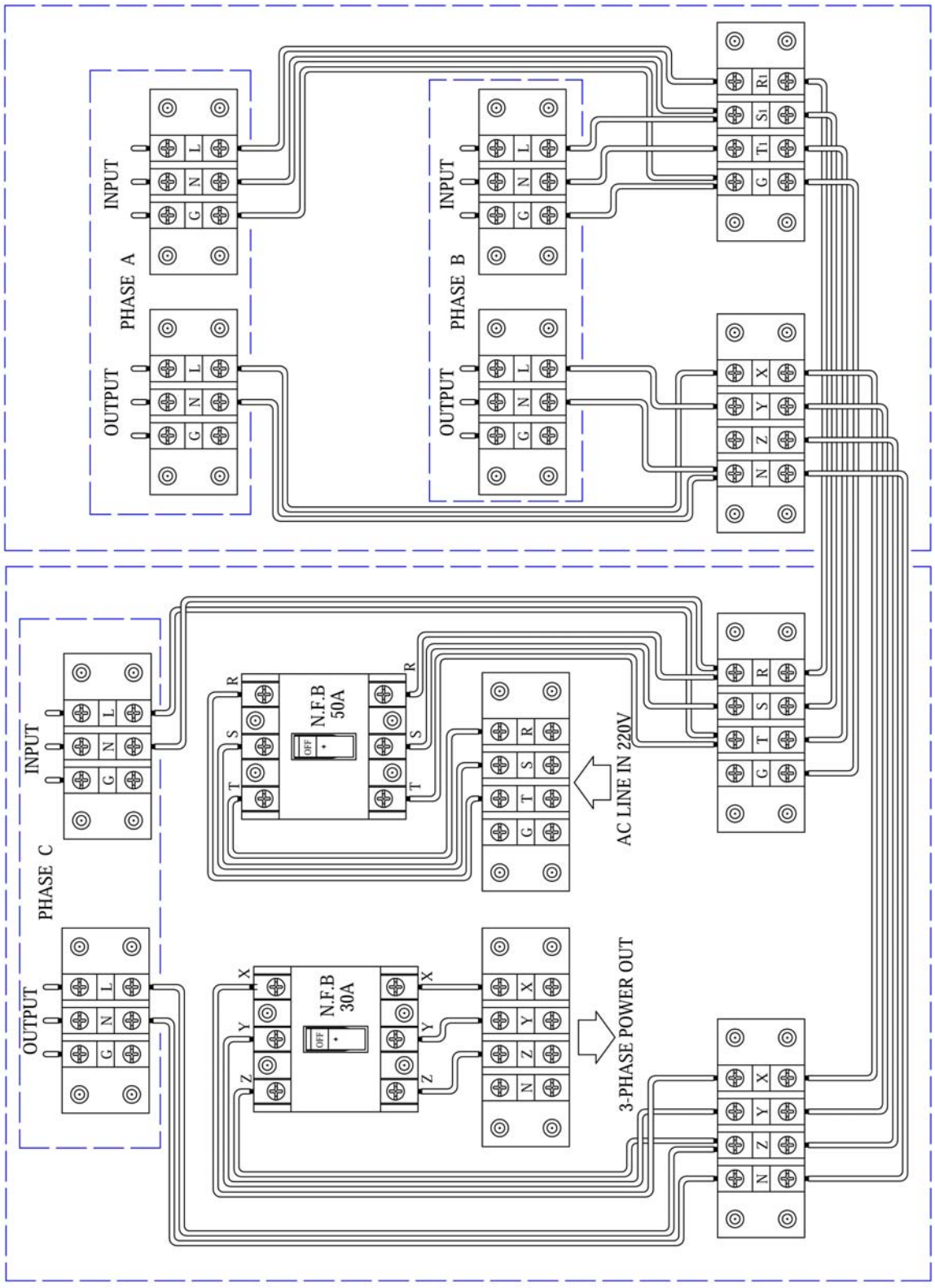
(FIG 6-2)


POWER SOURCE
1φ 2W
A
B
C
INPUT
OUTPUT
L
N
G
LOAD
A
B
C
N
N.F.B
3φ 3W Y-△
3φ 3W Y-Y
3φ 4W Y-Y
AC SOURCE
LOAD
VL=√3 x VP
CONNECT3.BMP

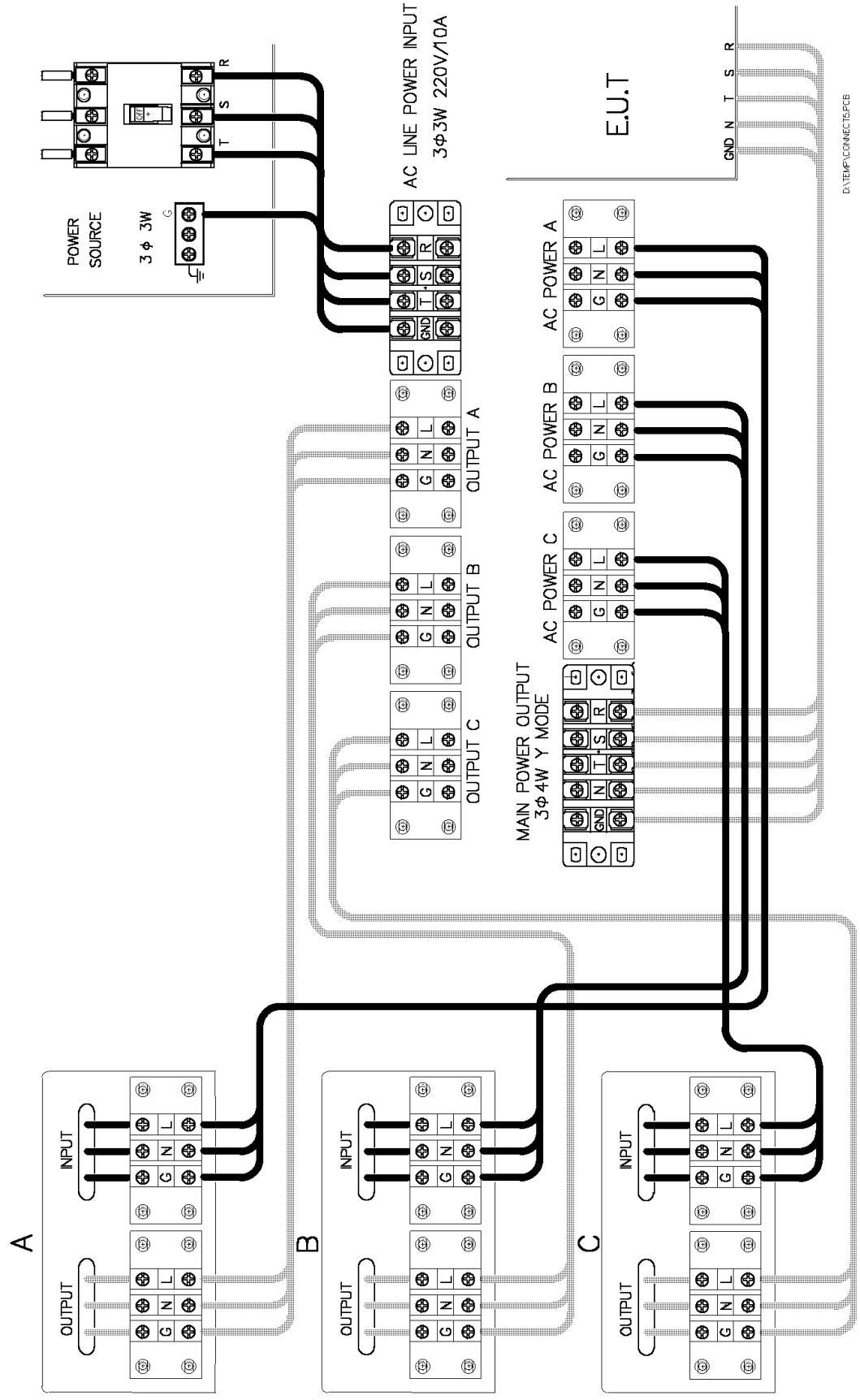
(FIG 6-3)


POWER SOURCE
3φ 3W
R
S
T
G
A
B
C
INPUT
OUTPUT
L
N
G
OUTPUT N.F.B
LOAD
A
B
C
N
3φ 3W Y-Δ
3φ 3W Y-Y
3φ 4W Y-Y
AC SOURCE
LOAD
VL=√3 x VP
CONNECT4.BMP

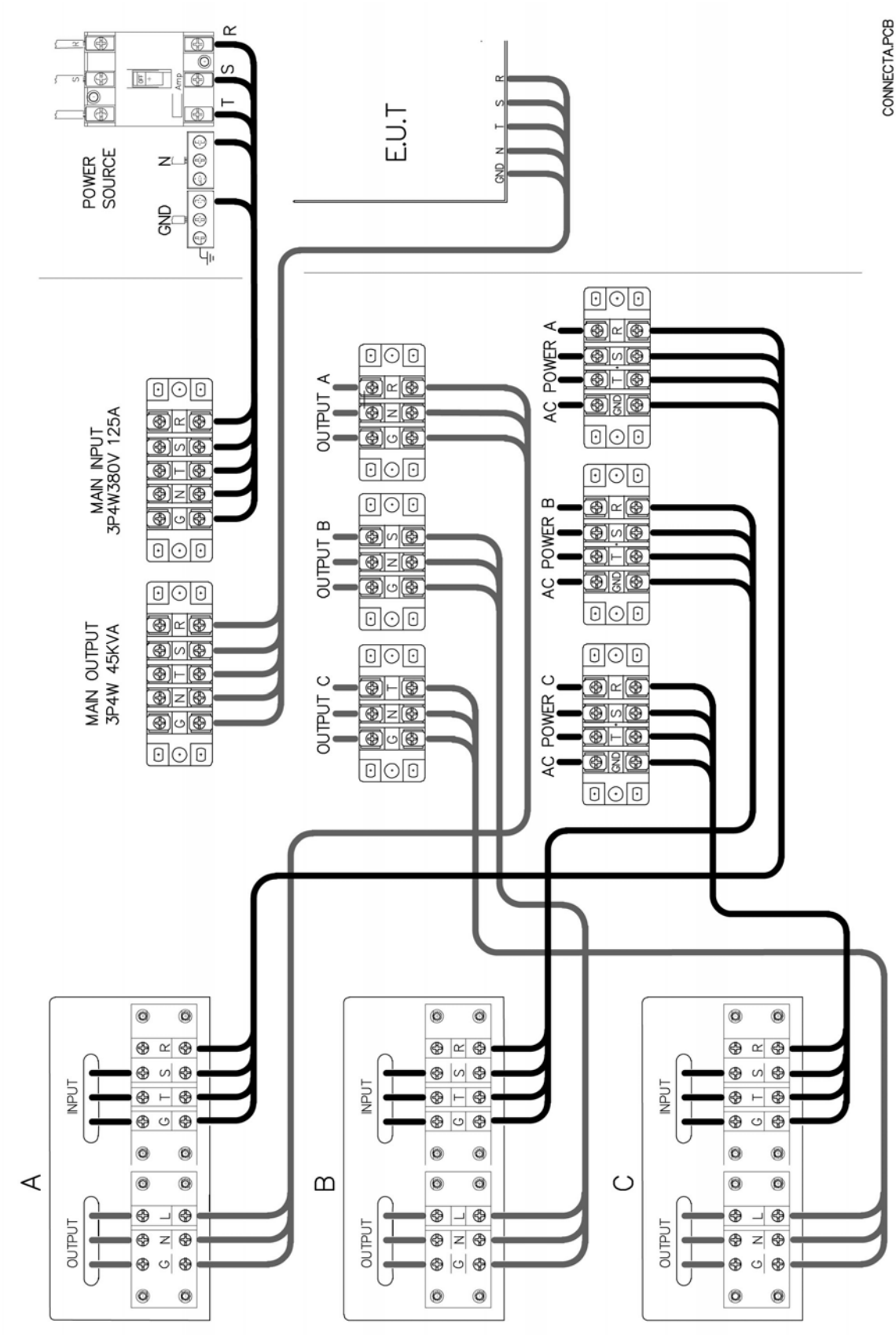
(FIG 6-4)


INPUT
PHASE A
OUTPUT
INPUT
PHASE B
OUTPUT
INPUT
PHASE C
OUTPUT
N.F.B 50A
N.F.B 30A
OFF
AC LINE IN 220V
3-PHASE POWER OUT
AUTOCAD R14\PWSWU\CPC301\CON3P21 10-29-1999

(FIG 6-5)


POWER SOURCE
3 φ 3W
R
S
T
AC LINE POWER INPUT
3φ3W 220V/10A
GND
OUTPUT A
OUTPUT B
OUTPUT C
MAIN POWER OUTPUT
3φ4W Y MODE
AC POWER C
AC POWER B
AC POWER A
E.U.T
GND N T S R
D:\TEMP\CONNECT5.PCB
A
B
C
INPUT
OUTPUT
L
N
G

(FIG 6-6)


POWER SOURCE
N
GND
R
S
T
E.U.T
GND N T S R
CONNECTA.PCB
MAIN INPUT 3P4W380V 125A
MAIN OUTPUT 3P4W 45KVA
OUTPUT A
OUTPUT B
OUTPUT C
AC POWER A
AC POWER B
AC POWER C
A
B
C
INPUT
OUTPUT
G N L
G N T S R

附录壹：配线说明

一、CIF 系列交流电源供应器配线说明

机 型	输入 电压	输入 最大电流	输入 N.F.B	输入 使用线径	输出 最大电流（ Rms ）	输出 使用线径
CIF-2000					0-140V 16.6A	
2KVA	1Ø2W 220V	18 A	20A	3.5mm²	0-280V 8.3A	3.5mm²
CIF-3000					0-140V 25A	
3KVA	1Ø2W 220V	27.2A	30A	5.5mm²	0-280V 12.5A	5.5mm²
CIF-50000	1Ø2W 220V	45.5 A	50A	8 mm²	0-140V 41.6A	
5KVA	3Ø3W 220V	30.3A	40A	5.5mm²	0-280V 20.8A	8.0mm²
CIF-7500	1Ø2W 220V	68A	75A	14mm²	0-140V 62.5A	
7.5KVA	3Ø3W 220V	45.3 A	50A	8 mm²	0-280V 31.2A	14mm²
CIF-1030	1Ø2W 220V	90 A	100 A	22mm²	0-140V 83.3A	
10KVA	3Ø3W 220V	60 A	75A	14mm²	0-280V 41.6A	22mm²
CIF-1530					0-140V 125A	
15KVA	3Ø3W 220V	90A	100 A	22mm²	0-280V 67.5A	38mm²
CIF-2030					0-140V 167A	
20KVA	3Ø3W 220V	121.2 A	125 A	38mm²	0-280V 83.5A	60mm²
CIF-3030					0-140V 250A	
30KVA	3Ø3W 220V	181 A	200 A	38mm² X2	0-280V 125A	60mm²

注：(1)配线建议采用多心绞线。

(2)配线时，导线应对绞。若导线超过 3 公尺时应再加粗一级，
例：原 3.5mm² 改为 5.5mm² 。

(3)N.F.B.：无熔丝断路器。

(4)线材及无熔丝断路器，请使用优良品牌。

(5)若输出常需 0-150V 或 0-300V 切换使用时，导线应使用 0-150V 设定之导线线径。

(6)2000VA 以上输入电源固定为 AC 220V，如需改为 AC 110V 时，请另行订制。

二、CF 系列交流电源供应器配线说明

机　　型	输　　入				输　　出	
	电源	最大电流	建议使用无熔丝开关	建议使用线径	最大电流 (Rms)	建议使用线径
CF-500A 500VA	1Ø2W 110V	15A	15A	2.0mm²	0-150V 4.2A 0-300V 2.1A	2.0mm²
	1Ø2W 220V	7.5A	7.5A	2.0mm²		
CF-1000A 1KVA	1Ø2W 110V	30A	30A	5.5mm²	0-150V 8.4A 0-300V 4.2A	2.0mm²
	1Ø2W 220V	15A	15A	2.0mm²		
CF-2000A 2KVA	1Ø2W 220V	30A	30A	5.5mm²	0-150V 16.8A 0-300V 8.4A	2.0mm²
CF-3000A 3KVA	1Ø2W 220V	45A	50A	8mm²	0-150V 25.2A	3.5mm²
					0-300V 12.6A	2.0mm²
CF-5000A 5KVA	1Ø2W 220V	75A	75A	14mm²	0-150V 43A	8mm²
					0-300V 21.5A	3.5mm²

注：(1)配线建议采用多心绞线。

(2)配线时，导线应对绞。若导线超过 3 公尺时应再加粗一级，
例：原 3.5mm² 改为 5.5mm² 。

(3)N.F.B.：无熔丝断路器。

(4)线材及无熔丝断路器，请使用优良品牌。

(5)若输出常需 0-150V 或 0-300V 切换使用时，导线应使用 0-150V 设定之导线线径。

(6)2000VA 以上输入电源固定为 AC 220V，如需改为 AC 110V 时，请另行订制。

附录贰：导线线径与电流规格表

※请注意！线材规格请依下列表格，方能正常使用。

使用 CD-SERIES 直流电源供应器或 CF、CIF 交流电源供应器时，需特别注意输入与输出导线之线径问题，以防止因电流太大引起过热，而造成意外，下列表格为导线在不同温度下之线径与电流规格表。

AWG NO	线径 (约略值)	铜线温度			
		60℃	75℃	85℃	90℃
		电流(A)			
14	2mm²	20	20	25	25
12	3.5mm²	25	25	30	30
10	5.5mm²	30	35	40	40
8	8mm²	40	50	55	55
6	14mm²	55	65	70	75
4	22mm²	70	85	95	95
3	30mm²	85	100	110	110
2	38mm²	95	115	125	130
1	50mm²	110	130	145	150
0	60mm²	125	150	165	170
00	70mm²	145	175	190	195
000	80mm²	165	200	215	225
0000	100mm²	195	230	250	260

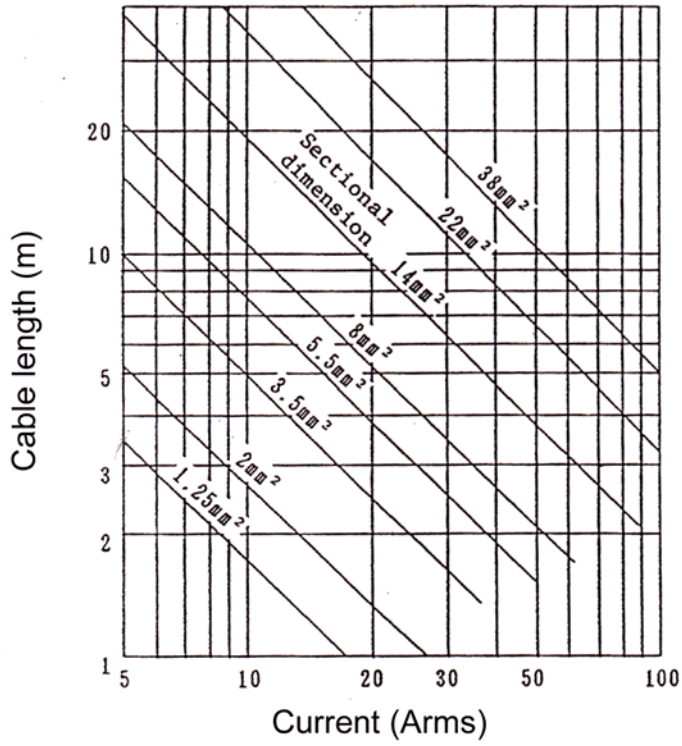
导线的阻抗与其长度成正比与线径成反比，并且直接影响 CD-SERIES 直流电源供应器或 CF、CIF 交流电源供应器的输出特性；所以，往往在输出端子上所量测出来的电压不同于负载上的电压，一般而言，这个电位差不得大于 0.5V(参考图 A3-1)。**备注：当电位差大于 0.5V 时，可将线径加粗 1 倍或 2 倍甚至 3 倍。**

下列表格为导线线径每公尺所产生之阻抗规格表。

AWG NO	Resistance in mΩ/M (at 20℃)
22	52.8
20	33.5
18	20.96
16	13.19
14	8.30
12	5.22
10	3.277

FIG A3-1 (Cable length and voltage drop)

Condition：Voltage drop 0.5V　　JIS C3307 IV Cable



附录参：GPIB、RS-232 界面指令说明

1. 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.

2.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: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

3.RS-232 Interface

• Mode : Start stop synchronization

• Baud rate : 1200.2400.4800.9600bps

• Command :

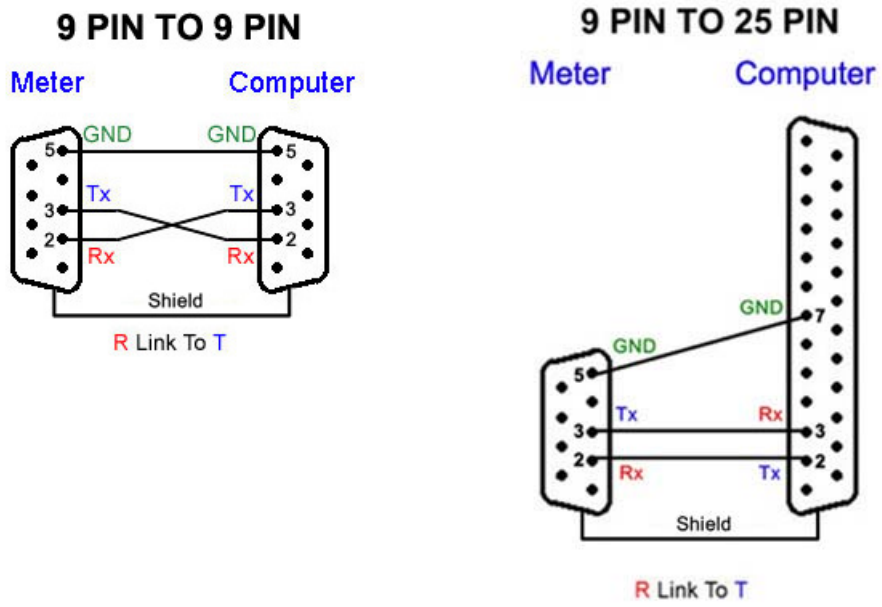
:SYSTem:LOCal(No Query)

Sets the system to local control.

:SYSTem:REMote(No Query)

Sets the system to remote control.

4.RS-232 PC To CF/CIF AC SOURCE Connection



5.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	

附录肆：错误码对照表

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	60	ROM FAILED"
-112	Program mnemonic too long	61	RAM FAILED"
-113	Undefined header	62	EEPROM R/W FAILED"
-121	Invalid character in number	63	USER SETTING LOST"
-123	Exponent too large	64	Command allowed only with RS-232
-124	too many digits	65	Command not allowed in local
-128	Numeric data not allowed	100	FAULT
-131	Invaild suffix	101	OLP
-138	Suffix not allowed	102	OTP
-140	Character data error	103	ODP
-141	Invalid character data	104	FUSE
-144	Character data too long		
-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		

附录伍：RS232 鲍率与 GPIB 地址设定法：

1.关闭电源开关(POWER)

2.频率指拨开关拨于 0069

3.按着 RESET 开关同时开启电源开关(POWER)

4.以上步骤若操作正确则 REMOT 灯亮，表示已进入设定模式

5.RS232 鲍率设定

a.指拨开关与鲍率关系如下：

0100：1200

0101：2400

0102：4800

0103：9600

b.依上列所示选定所需鲍率来设定指拨开关，如：9600 为 0103

c.按 RESET 键完成输入动作

d.拨指拨开关于 0300 位置

e.按 RESET 键完成储存动作，并跳出设定模式进入正常操作模式

f.将指拨开关拨于所需输出电源频率，如：50.00

6.GPIB 地址设定：

a.指拨开关与地址关系如下：

0200：00

0201：01

0231：31

b.依上列所示选定所需地址，来设定指拨开关，如 0206 表示设定地址为 06

c.按 RESET 键完成输入动作

d.调指拨开关于 0300 位置

e.按 RESET 键完成储存动作，并跳出设定模式进入正常操作模式

f.将指拨开关设于所需输出电源频率，如：50.00

注：GPIB 内定 Address 为 06

RS232 内定鲍率为 9600

附录陆：程序范例

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;
                                 }
                        }
        }
```