Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA EN
TECNOLOGÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Plataforma software de automatización de medidas de laboratorio.

Apéndice y anexos.

Alumno: Agirre Ezama, Harkaitz
Director: de Diego Rodrigo, José Miguel

Curso: 2018-2019

Fecha: *12 de Noviembre de 2018*

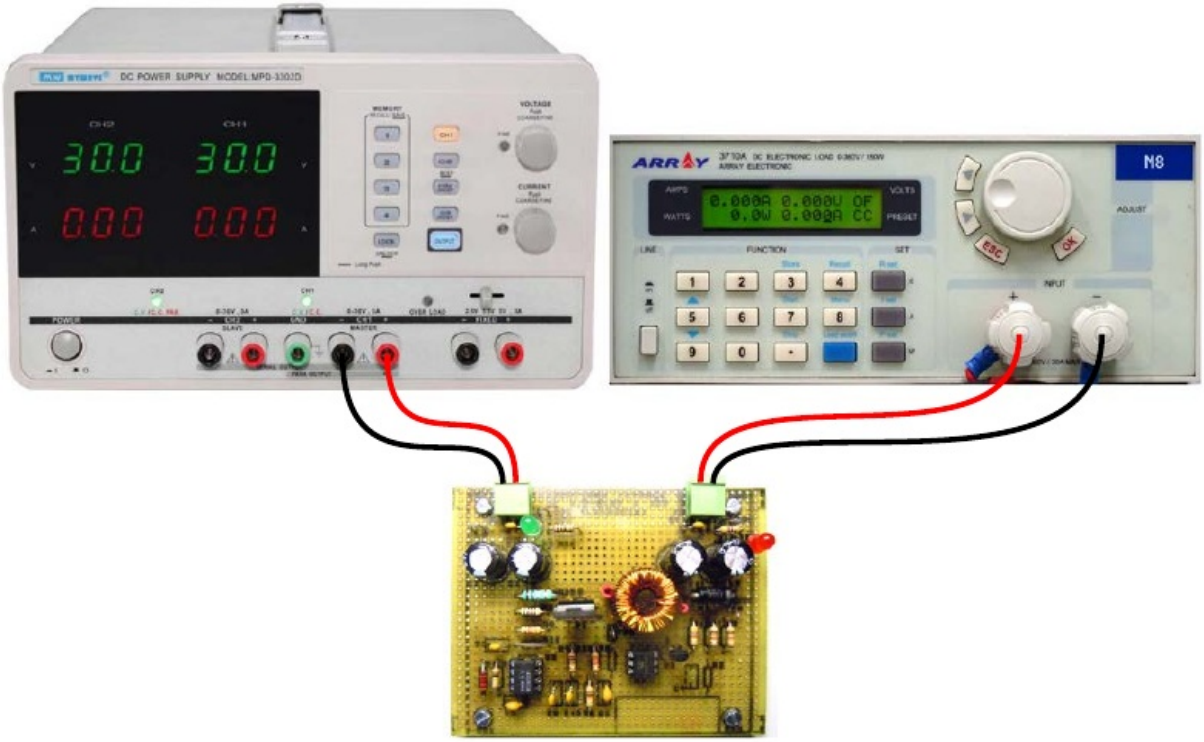
Índice

Apéndice A - Manual 5
Manual de usuario de *HARKD* 8
1 Programa gráfico HARKDW 8
1.1 Descarga de la aplicación. 8
1.2 Descomprimir la aplicación. 8
1.3 Medida de la curva característica de un un conversor DC/DC. 9
2 Programa de terminal HARKDC. 13
De forma interactiva 13
Ejecutar una prueba integrada 13
Ejecutar una prueba personalizada 13
2.1 Descripción de los comandos. 14
2.2 Relación de comandos disponibles con la aplicación *harkdc.exe*: 15
3 Ejemplos de uso. 16
3.1 Establecer distintas cargas. 16
3.2 Establecer distintas tensiones. 16
Apéndice B - Codigo fuente. 18
1 Codigo fuente del programa HARKDC 18
harkdc/src/tests/tests.h 18
harkdc/src/harkd.h 19
harkdc/src/devices/checksum_flymake.h 23
harkdc/src/devices/protocols.h 24
harkdc/src/devices/checksum.h 25
harkdc/src/devices/devices.h 26
harkdc/src/harkd-hitz.c 27
harkdc/src/tests/example.c 30
harkdc/src/tests/dcdc.c 31
harkdc/src/harkd-table.c 33
harkdc/src/devices/checksum.c 35
harkdc/src/devices/mpd.c 36
harkdc/src/devices/example.c 38
harkdc/src/devices/protocols.c 40
harkdc/src/devices/array.c 41
harkdc/src/harkd-main.c 45
harkdc/src/harkd.c 47
harkdc/src/harkd-core.c 51
harkdc/src/harkd-serial.c 53
2 Codigo fuente del programa HARKDW. 55
harkdw/bin/harkdw-dcdc.tcl 55
harkdw/harkdw/pkgIndex.tcl 56
harkdw/harkdw/harkd.tcl 57
harkdw/harkdw/harkdw.tcl 58
Apendice C - Manuales de instrumentos. 62

Apéndice A - Manual

Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO
Teknologia Elektronikoa Saila
Departamento de Tecnología Electrónica

Manual de usuario de la plataforma software de automatización de medidas de laboratorio HARKD.



v2.0 H. Agirre, 12 de Noviembre de 2018

Índice

Manual de usuario de *HARKD*
. 5
1 Programa gráfico HARKDW 8
1.1 Descarga de la aplicación. 8
1.2 Descomprimir la aplicación. 8
1.3 Medida de la curva característica de un un conversor DC/DC. 9
2 Programa de terminal HARKDC. 12
Ejecutar de forma interactiva 13
Ejecutar una prueba integrada 13
Ejecutar una prueba personalizada 13
2.1 Descripción de los comandos. 13
2.2 Relación de comandos disponibles con la aplicación *harkdc.exe*: 14
2.3 Ejemplos de uso. 15
Establecer distintas cargas. 16
Establecer distintas tensiones. 16

Manual de usuario de *HARKD*

La plataforma software de automatización *HARKD* se compone de dos programas:

1. El interfaz gráfico *HARKDW*.
2. El programa de consola *HARKDC*.

Ambos programas son portables, de modo que no necesaria su instalación desde windows. Simplemente se copian a un directorio y se ejecutan.

1 Programa gráfico HARKDW

Este programa permite trazar la curva característica de un convertidor DC-DC de forma automática, utilizando los siguientes instrumentos conectados por USB a un ordenador con Windows 7 o superior:

1. Fuente de alimentación programable MYWAVE, modelo MPD-3305D.
2. Carga electrónica programable ARRAY, modelo 3710A.

El resultado se almacena en un fichero Excel de Microsoft Office v2007 o superior (.xlsx).

1.1 Descarga de la aplicación.

Ambas aplicaciones, junto con su manual de usuario se pueden descargar desde eGela del curso actual:

1. https://egela1819.ehu.eus/

Figura 1. Página web de descarga del software.

También esta disponible el código fuente del software en los siguientes links;

1. https://github.com/harkaitz/harkdc
2. https://github.com/harkaitz/harkdw

1.2 Descomprimir la aplicación.

El software *HARKD* se halla contenido en un fichero comprimido *harkd.zip*:

1. El intérprete de comandos *harkdc.exe*
2. La interfaz gráfica *harkdw.exe*
3. El manual de usuario *harkd-manual.pdf*

Para ejecutar la aplicación **se debe descomprimir el fichero** *harkd.zip* en un directorio del sistema. Es importante que todos los programas se encuentren en la misma carpeta.

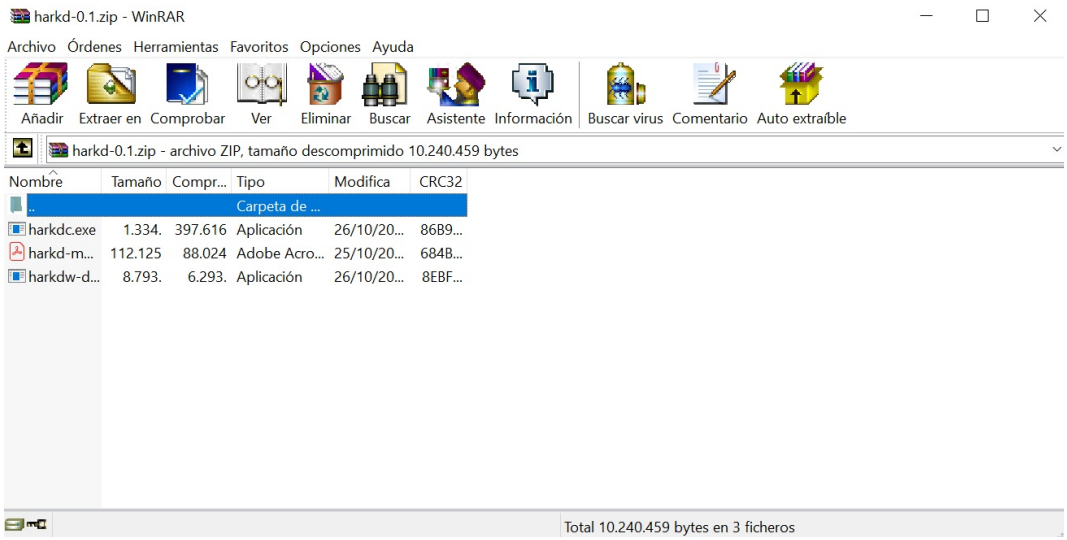


Figura 2. Fichero zip de distribución del software.

1.3 Medida de la curva característica de un un conversor DC/DC.

Asegurarse que los dos instrumentos se encuentran conectados por USB al ordenador desde el

cual se está ejecutando el programa. Al hacer dos «clicks» en el programa *harkdw.exe*, se abre la siguiente ventana;

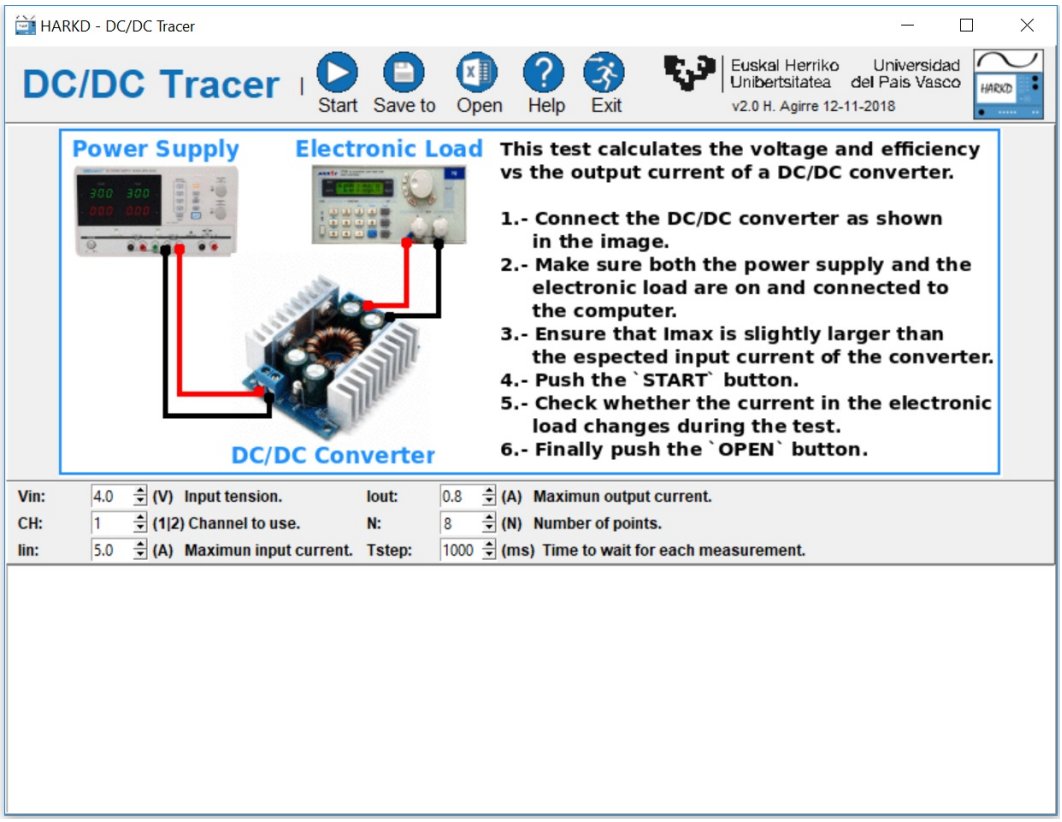


Figura 3. Ventana inicial de la interfaz gráfica.

El siguiente paso es establecer los valores en los instrumentos para la prueba.

De la fuente de alimentación. «V_{in}» , «I_{in}» y el canal a utilizar «CH».

De la carga electrónica. «I_{out}» , número de puntos «N» y el intervalo entre medidas «T_{step}»

Por último, pulsar «**Start**».

El botón «**Save to**» permite guardar el archivo Excel en otra carpeta y con otro nombre. (por defecto deja un fichero «*dcdc.xlsx*» en el mismo directorio desde donde se ejecuta el programa).

Durante la prueba se observa cómo se va llenando la tabla de abajo.

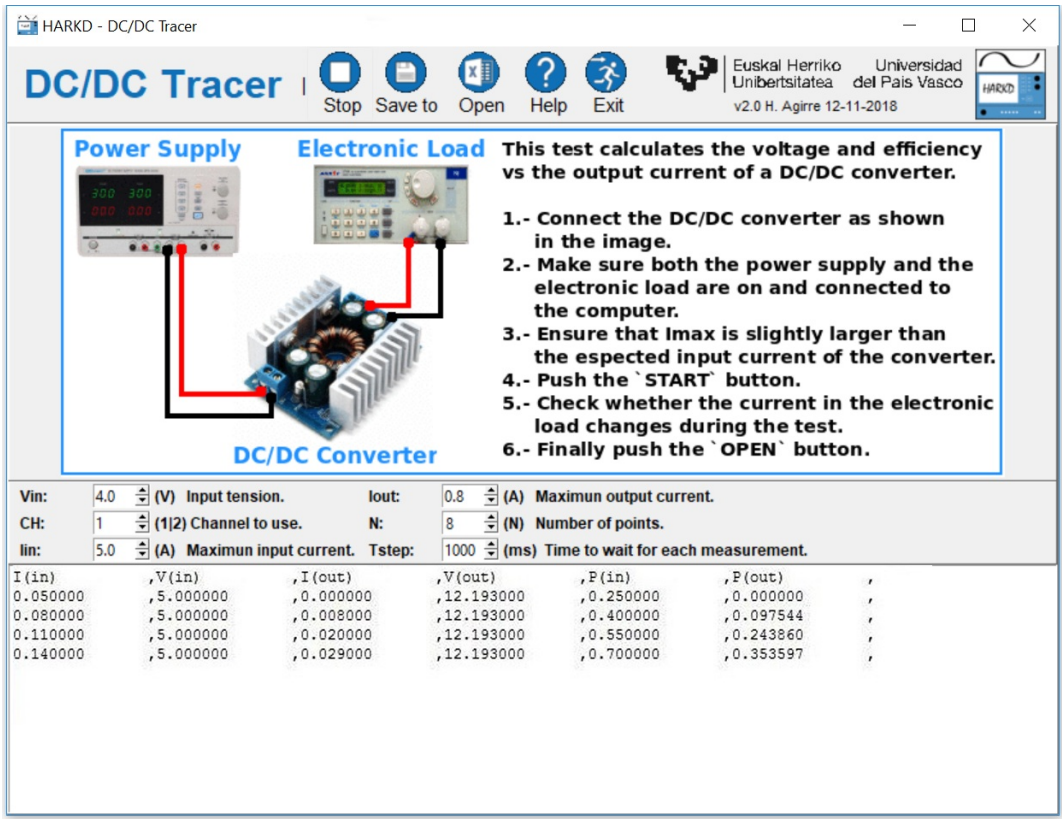


Figura 4. Interfaz gráfica durante una medida.

En cualquier momento se pulsar el botón «STOP» para detener las medidas.
Una vez finalizada la prueba, el botón cambia a «Restart».

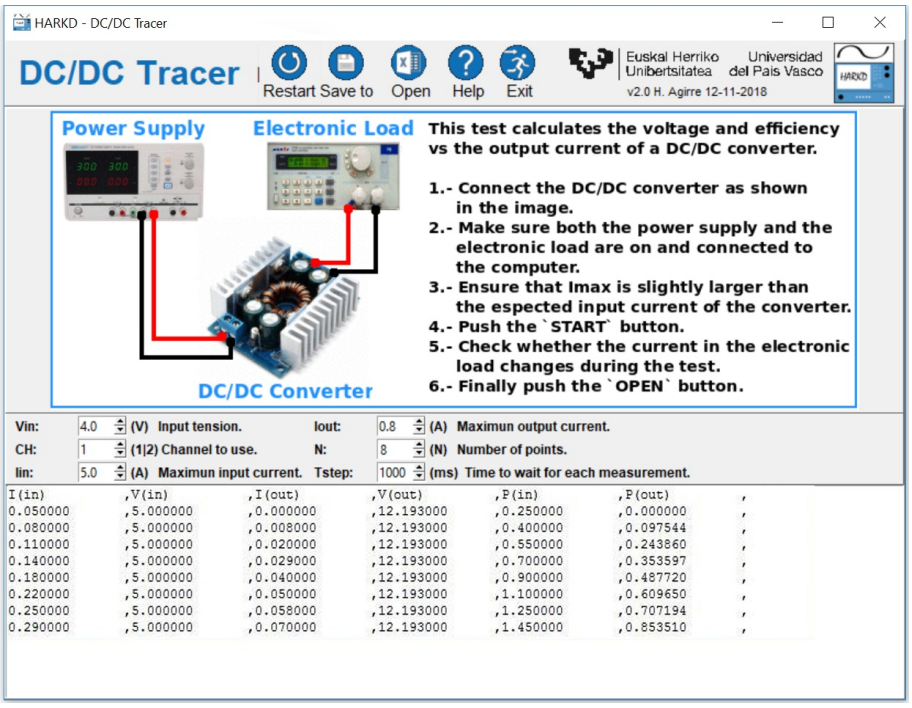


Figura 5. Interfaz gráfica una vez finalizada la medida.

Para ver los resultados en una hoja Excel pulsa el botón «OPEN».

	A	B	C	D	E	F	G
1	I(in)	V(in)	I(out)	V(out)	P(in)	P(out)	η(%)
2	0,05	5,0	0,000	12,193	0,25	0,000	0,00
3	0,08	5,0	0,010	12,193	0,40	0,122	30,48
4	0,11	5,0	0,020	12,193	0,55	0,244	44,34
5	0,14	5,0	0,029	12,193	0,70	0,354	50,51
6	0,18	5,0	0,040	12,193	0,90	0,488	54,19
7	0,22	5,0	0,050	12,193	1,10	0,610	55,42
8	0,25	5,0	0,058	12,193	1,25	0,707	56,58
9	0,29	5,0	0,070	12,193	1,45	0,854	58,86
10	0,33	5,0	0,080	12,193	1,65	0,975	59,12
11	0,36	5,0	0,089	12,193	1,80	1,085	60,29
12	0,38	5,0	0,099	12,193	1,90	1,207	63,53
13	0,41	5,0	0,109	12,166	2,05	1,326	64,69
14	0,49	5,0	0,119	12,166	2,45	1,448	59,09
15	0,53	5,0	0,129	12,166	2,65	1,569	59,22
16	0,57	5,0	0,139	12,166	2,85	1,691	59,34
17	0,60	5,0	0,149	12,166	3,00	1,813	60,42
18	0,65	5,0	0,160	12,166	3,25	1,947	59,89
19	0,68	5,0	0,170	12,166	3,40	2,068	60,83
20	0,72	5,0	0,180	12,166	3,60	2,190	60,83
21	0,74	5,0	0,190	12,166	3,70	2,312	62,47
22	0,77	5,0	0,200	12,166	3,85	2,433	63,20
23	0,81	5,0	0,210	12,166	4,05	2,555	63,08
24	0,88	5,0	0,220	12,166	4,40	2,677	60,83
25	0,94	5,0	0,230	12,166	4,70	2,798	59,54
26	1,00	5,0	0,240	12,166	5,00	2,920	58,40
27	1,02	5,0	0,250	12,166	5,10	3,042	59,64
28	1,05	5,0	0,260	12,166	5,25	3,163	60,25
29	1,09	5,0	0,270	12,166	5,45	3,285	60,27
30	1,11	5,0	0,280	12,166	5,55	3,406	61,38
31	1,16	5,0	0,290	12,166	5,80	3,528	60,83
32	1,22	5,0	0,300	12,166	6,10	3,650	59,83
33	1,25	5,0	0,310	12,139	6,25	3,763	60,21
34	1,31	5,0	0,320	12,139	6,55	3,884	59,31
35	1,35	5,0	0,330	12,139	6,75	4,006	59,35
36	1,39	5,0	0,340	12,139	6,95	4,127	59,39
37	1,43	5,0	0,350	12,139	7,15	4,249	59,42

Figura 6. Archivo Excel generado.

2 Programa de terminal HARKDC.

Este es un programa de ordenador para linea de comandos.

Hay tres formas de iniciar el programa;

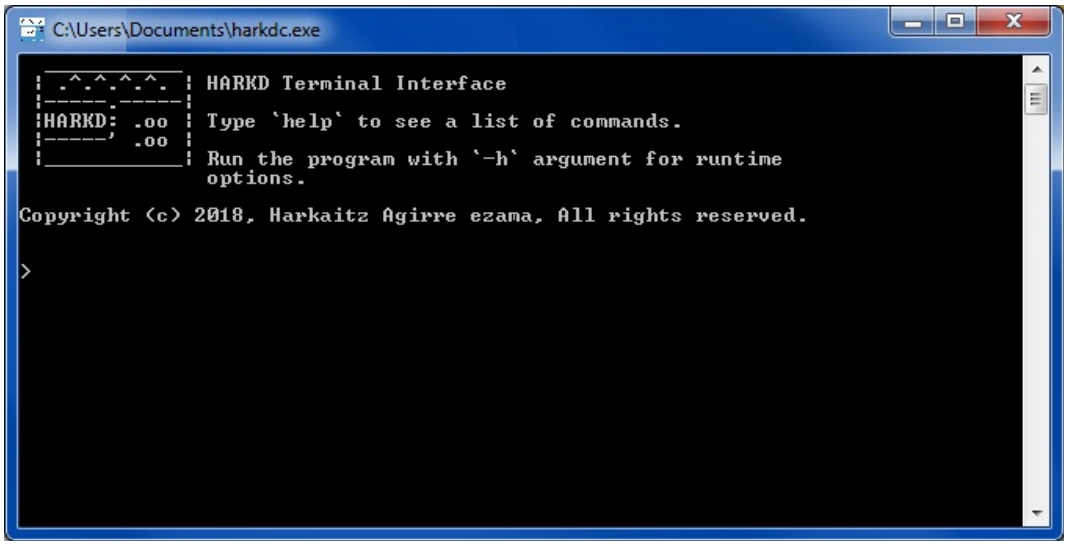


Figura 7. Interfaz de comandos iniciada de forma interactiva.

Ejecutar de forma interactiva

Es la opción por defecto. En el sistema operativo *Microsoft Windows* solo se necesita dar dos «clicks» al programa para que se abra la linea de comandos.

```
> harkdc
```

Ejecutar una prueba integrada mediante argumentos del programa

El programa HARKD cuando recibe argumentos de la linea de comandos en vez de inicial el interprete interactivo ejecuta el modulo de pruebas.

Si el primer argumento es «list» se imprime una lista de pruebas, en cambio, si como primer argumento se especifica el nombre de la prueba es ejecuta la prueba.

Si se especifica la palabra «help» como argumento a la prueba, se imprime la ayuda concerniente a la prueba.

LINEA DE COMANDOS DEL S.O

```
> harkdc list

DC/DC      Measure the efficiency of a DC/DC converter.
example    Simply prints the variables specified.

> harkdc dc/dc help

This test measures the efficiency of a DC/DC converter
using an ‘electronic load‘ and a ‘DC power supply‘.

+ Variables;
 + Vin         = 12
 + Imax        = 2
 + N           = 10
 + Tstep       = 2000
 + CH          = 1
 + Output      = dcdc.xlsx

> harkdc dc/dc Vin=13 Imax=2 N=5

I(in)      ,V(in)      ,I(out)     ,V(out)     ,P(in)      ,P(out)     ,
0.130000   ,13.000000  ,0.000000   ,19.697000  ,1.690000   ,0.000000   ,
2.110000   ,13.000000  ,0.400000   ,19.563000  ,27.430000  ,7.825200   ,
2.850000   ,13.000000  ,0.800000   ,11.586000  ,37.050000  ,9.268800   ,
2.460000   ,13.000000  ,1.200000   ,6.708000   ,31.980000  ,8.049600   ,
2.160000   ,13.000000  ,1.600000   ,3.819000   ,28.080000  ,6.110400   ,
```

Ejecutar una prueba personalizada

Se pueden especificar los comandos a realizar desde un fichero mediante una redirección.

```
> harkdc < fichero.txt
```

2.1 Descripción de los comandos.

La mayoría de comandos soportan argumentos opcionales, por ejemplo el comando «interfaces».

Si se ejecuta este comando sin argumentos, se imprime una lista de dispositivos soportados, en cambio, si como se muestra en la imagen se especifica el modelo de dispositivo como argumento se imprime la lista de entradas y salidas soportadas.

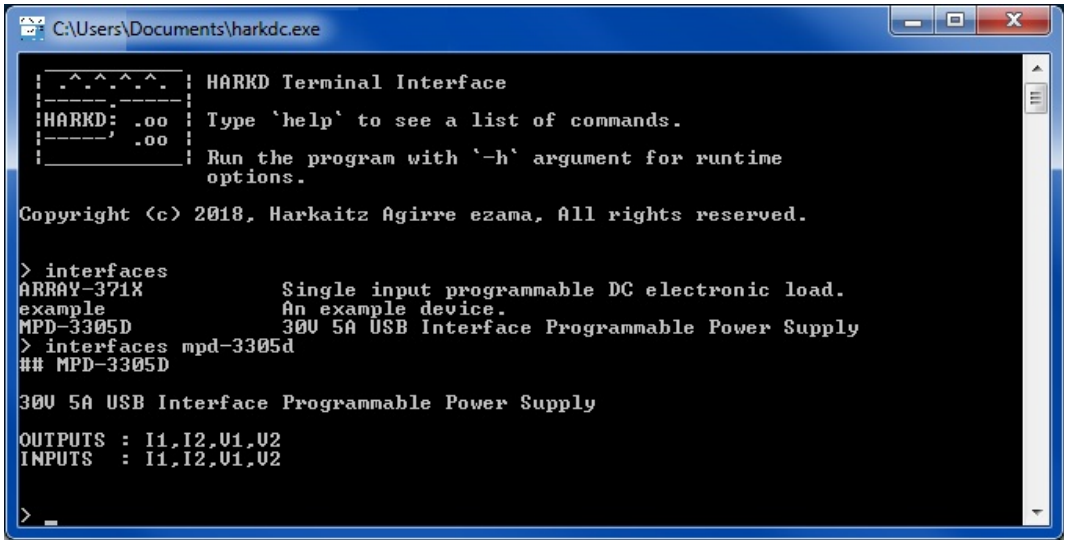


Figura 8. Ejemplo de comando en HARKD.

Comando: **help**	Imprime la lista de comandos disponibles.

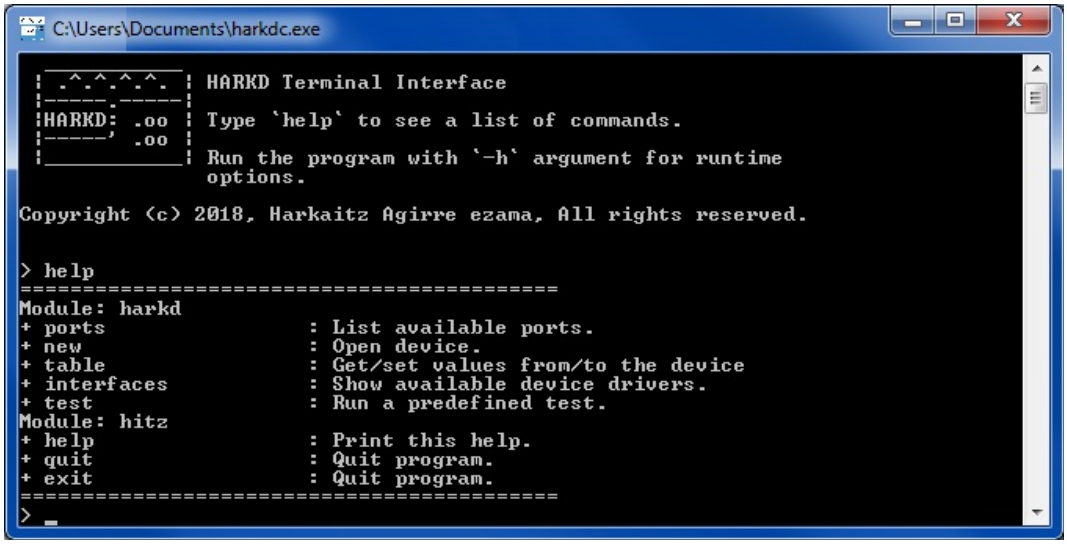


Figura 9. Salida del comando «help».

2.2 Relación de comandos disponibles con la aplicación *harkdc.exe*:

Comando: **help**	Imprime la lista de comandos disponibles.

Comando: **ports**	Imprime la lista de puertos serie disponibles.

Comandos: **quit**,**exit**	Se sale del programa.

Comando: **interfaces**	Visualiza una lista de dispositivos soportados.
Argumentos:	
«modelo» (opcional)	Si se especifica, imprime las entradas y salidas del dispositivo.

Comando: **test**	Visualiza la lista de pruebas integradas soportadas
Argumentos:	
«prueba» (opcional)	Si se especifica, se ejecuta la prueba.
help	Si el segundo argumento es help, se imprime una lista de variables que soporta la prueba.
«opción»=«valor»	Las opciones de la prueba.

Comando: **new**	Se conecta con un dispositivo.
Sin argumentos:	Imprime una breve descripción.
Argumentos:	
«nombre»	Nombre con la que se referirá al dispositivo en el comando table.
«modelo»	Modelo del dispositivo.
search (opcional)	Si se especifica busca automáticamente el puerto correspondiente.

Comando: **table**	Establece un valor en una maquina, o obtiene un valor.
Argumentos:	
«nombre»/«salida»=«valor»	Se fija una salida.
«nombre»/«entrada»	Se obtiene un valor.

2.3 Ejemplos de uso.

Establecer distintas cargas.

Además de la utilización de la carga mediante la propia botonera del instrumento, con la interfaz de comandos también se puede realizar de forma sencilla.

Ejemplo de establecimiento de la carga con corriente constante de 3 y 2 amperios.

```
harkdc> new carga ARRAY-371X search
harkdc> table carga/I=3
harkdc> table carga/I=2
```

Establecer distintas tensiones.

Ejemplo de establecimiento de la fuente con tensión de salida 3 y 4 voltios.

```
harkdc> new fuente MPD-3305D search
harkdc> table fuente/V1=3
harkdc> table fuente/V2=4
```

Apéndice B - Codigo fuente.

1 Codigo fuente del programa HARKDC

Source file: **harkdc/src/tests/tests.h**

```
/* List of ‘const harkd_test_t‘ typed variables.*/
extern const harkd_test_t HARKD_DCDC_TEST;
extern const harkd_test_t HARKD_EXAMPLE_TEST;
const harkd_test_t *harkd_tests[] = {
    &HARKD_DCDC_TEST,
    &HARKD_EXAMPLE_TEST,
    NULL
};
```

Source file: **harkdc/src/harkd.h**

```
#ifndef _HARKD_H_
#define _HARKD_H_
#include <stddef.h>
#define HARKD_MAX_NAME 20

typedef struct hitz_s        hitz_t;
typedef struct sp_port       sp_port_t;
typedef struct harkd_test_s  harkd_test_t;
typedef struct harkd_table_s harkd_table_t;
typedef struct hitz_module_s hitz_module_t;
typedef int harkd_r;
#define HARKD_OK   1
#define HARKD_ERR -1


typedef struct harkd_dev_obj_s       harkd_dev_obj_t;



void harkd_library_init  (hitz_t *opt_hitz);
void harkd_library_clean (void);


/**
 * Some usefull macros.
 */
#define HARKD_LIST_FOREACH(TYPE,CURR,LIST)        \
     for (TYPE **_nxt = &(LIST),*CURR = (LIST);   \
          CURR;                                   \
          _nxt=&((*_nxt)->next),CURR=*(_nxt))
#define HARKD_ARRAY_FOREACH(TYPE,CURR,LIST) \
     for(TYPE **_nxt=(LIST),*CURR=*_nxt;CURR;_nxt++,CURR=*_nxt)
```

```
#ifndef NL
#  define NL "\n"
#endif
```

```
/**
 * A structure to represent a device's interface.
 */
typedef struct harkd_dev_itf_s   harkd_dev_itf_t;
typedef harkd_r (*harkd_init_f) (harkd_dev_obj_t *harkd,const char *port,const char
*args[]);
typedef harkd_r (*harkd_cmd_f)  (harkd_dev_obj_t *harkd,const char *args[]);
typedef harkd_r (*harkd_opr_f)  (harkd_dev_obj_t *harkd);
typedef harkd_r (*harkd_set_f)  (harkd_dev_obj_t *harkd,const char *var,double *val);
struct harkd_dev_itf_s {
     const char *name;  /**< The name for `harkd_get_def` */
     const char *help;  /**< A descriptive help. */
#    define harkd_help_paragraph(TXT)          TXT "\n\n"
#    define harkd_help_inputs(TXT)             "INPUTS  : " TXT "\n"
#    define harkd_help_outputs(TXT)            "OUTPUTS : " TXT "\n"
#    define harkd_help_commands()              "\n### Commands\n\n"
#    define harkd_help_command(NAME,ARGS,HELP) "```\n" NAME " " ARGS ": " HELP "\n```\n"
     char        port_type;
     const char *long_help;
     struct {
          int          len;
          harkd_init_f init;
          harkd_cmd_f  command;
          harkd_opr_f  clear;
          harkd_set_f  set,get;
     } fn;
};
extern const harkd_dev_itf_t **harkd_dev_itf_array;
const harkd_dev_itf_t *harkd_get_itf(const char *name);
```

```
void harkd_list_ports     (const harkd_dev_itf_t *device,char buffer[],int len);
int  harkd_port_is_openned(const char *portname);
void harkd_wait           (int ms);
```

```
#define HARKD_MAX_STRING  64
#define HARKD_MAX_TEXT    1024
```

```
/**
 * A structure to represent a device's. (harkd.c)
 */
```

```
struct harkd_dev_obj_s {
     hitz_t                *hitz;
     char                   name[HARKD_MAX_STRING];
     const harkd_dev_itf_t *itf;

     sp_port_t             *port;
     char                   portname[HARKD_MAX_STRING];
     harkd_dev_obj_t       *next;
     char                  *itf_data[1];
};
extern harkd_dev_obj_t *harkd_dev_obj_list;


harkd_dev_obj_t *harkd_new        (const char *name,const harkd_dev_itf_t *device,
                                   const char *port,const char *options[]);
harkd_dev_obj_t *harkd_get        (const char *name);
void             harkd_free       (harkd_dev_obj_t *harkd);
harkd_r          harkd_run        (harkd_dev_obj_t *harkd,const char *args[]);
harkd_r          harkd_var_set    (harkd_dev_obj_t *harkd,const char *var,double *val);
harkd_r          harkd_var_get    (harkd_dev_obj_t *harkd,const char *var,double *val);
void             harkd_close      (harkd_dev_obj_t *harkd);
int              harkd_is_open    (harkd_dev_obj_t *harkd);
int              harkd_port_is_openned(const char *portname);
#define          harkd_itf_data(H,T) ((T*) (H)->itf_data)
```

```
/**
 * (harkd-table.c)
 */
harkd_table_t *harkd_table_new        (const char *filename);
void           harkd_table_free       (harkd_table_t *t);
void           harkd_table_add        (harkd_table_t *t,double d);
void           harkd_table_add_string (harkd_table_t *t,const char *s);
void           harkd_table_add_formula(harkd_table_t *t,const char *s,...);
void           harkd_table_next  (harkd_table_t *t);
void           harkd_table_next_column(harkd_table_t *t);
int            harkd_table_column(harkd_table_t *t);
int            harkd_table_row   (harkd_table_t *t);
void           harkd_table_add_chart (harkd_table_t *t,const char *names,const char *yx,
...);
```

```
/* ----------------- SERIAL PORT ------------------------------- */
harkd_r    harkd_serial_ports    (char *o_ports,int o_len);
sp_port_t *harkd_serial_open     (harkd_dev_obj_t *harkd,const char *portname);
harkd_r    harkd_serial_write    (harkd_dev_obj_t *harkd,void *buffer,size_t len);
harkd_r    harkd_serial_read     (harkd_dev_obj_t *harkd,void *buffer,size_t len);
char      *harkd_serial_gets     (harkd_dev_obj_t *harkd,char *buffer,size_t len);
harkd_r    harkd_serial_puts     (harkd_dev_obj_t *harkd,char *buffer);
```

```
void       harkd_set_serial_error(harkd_dev_obj_t *harkd,const char *whence);
```

```
/* ------------------- TESTS ------------------------------------ */
typedef struct harkd_test_var_s harkd_test_var_t;
struct harkd_test_var_s {
     const char *name;
     const char *value;
     const char *help;
     const char *opt_choises[10];
};
```

```
typedef struct harkd_test_s     harkd_test_t;
struct harkd_test_s {
     const char *name;
     const char *description;
     const char *long_description;
     const harkd_test_var_t opts[20];
     harkd_r (*fun) (harkd_test_var_t *opts);
};
extern const harkd_test_t  **harkd_test_array;
harkd_r              harkd_test_run(const harkd_test_t *t,const char *opts[]);
#define FOREACH_TEST(T) FOREACH_LIST(harkd_test_t,T,harkd_test_list)
#define FOREACH_TEST_VARIABLES(V,VARS) for(harkd_test_var_t *V = VARS;V->name;V++)
```

```
extern hitz_module_t *HARKD_HITZ_MODULE;
```

```
/* ------------------ ERROR REPORTING ------------------------------------------- */
const char *harkd_strerror(void);
harkd_r     harkd_log(harkd_r type,const char *format,...);
hitz_t     *harkd_hitz(void);
#define harkd_error_invalid_variable(H,VAR) harkd_log(HARKD_ERR,"Invalid variable
‘%s‘.",VAR)
#define harkd_error_invalid_command(H,VAR) harkd_log(HARKD_ERR,"Invalid variable ‘%s‘.",
VAR)
#define harkd_error(H,...) harkd_log(HARKD_ERR,##__VA_ARGS__)
#define harkd_printf(...)  hitz_fprintf(harkd_hitz(),##__VA_ARGS__)

#endif
```

Source file: **harkdc/src/devices/checksum_flymake.h**

```
#ifndef _CHECKSUM_H_
#define _CHECKSUM_H_
#ifdef _WIN32
#  include <stdint.h>
typedef uint8_t u_int8_t;
typedef uint16_t u_int16_t;
typedef uint32_t u_int32_t;
#else
#  include <sys/types.h>
#endif
static u_int8_t array_checksum(u_int8_t *array,u_int8_t len) {
     u_int32_t chk = 0;
     for(int i=0;i<len;i++) {
          chk = (chk + array[i]); //  % 256
     }
     return chk % 256;
}
```

Source file: **harkdc/src/devices/protocols.h**

```
#ifndef _HARKD_PROTOCOLS_H_
#define _HARKD_PROTOCOLS_H_
#include "../harkd.h"
#ifdef _WIN32
#  include <stdint.h>
typedef uint8_t  u_int8_t;
typedef uint16_t u_int16_t;
typedef uint32_t u_int32_t;
#else
#  include <sys/types.h>
#endif
typedef struct __attribute__((__packed__)) {
     u_int8_t flag;
     u_int8_t addr;
     u_int8_t cmd;
     u_int8_t info[22];
     u_int8_t chksum;
} msg26_t;
harkd_r msg26_send(harkd_dev_obj_t *harkd,msg26_t *msg,u_int8_t addr);
harkd_r msg26_recv(harkd_dev_obj_t *harkd,msg26_t *msg);
void    msg26_init(msg26_t *msg,u_int8_t cmd);


#endif
```

Source file: **harkdc/src/devices/checksum.h**

```
#ifndef _CHECKSUM_H_
#define _CHECKSUM_H_
#ifdef _WIN32
#  include <stdint.h>
typedef uint8_t  u_int8_t;
typedef uint16_t u_int16_t;
typedef uint32_t u_int32_t;
#else
#  include <sys/types.h>
#endif
u_int8_t checksum8(u_int8_t *array,u_int8_t len);
#endif
```

Source file: **harkdc/src/devices/devices.h**

```
/* List of ‘const harkd_dev_itf_t‘ typed variables.*/
extern const harkd_dev_itf_t HARKD_DEVICE_ARRAY_371X;
extern const harkd_dev_itf_t HARKD_DEVICE_EXAMPLE;
extern const harkd_dev_itf_t HARKD_DEVICE_MPD3305D;
const harkd_dev_itf_t *harkd_device_list[] = {
    &HARKD_DEVICE_ARRAY_371X,
    &HARKD_DEVICE_EXAMPLE,
    &HARKD_DEVICE_MPD3305D,
    NULL
};
```

Source file: **harkdc/src/harkd-hitz.c**

```
#include "harkd.h"
#include <hitz/hitz.h>
#include <string.h>
#include <ctype.h>
#ifdef WIN32
#  include <windows.h>
#else
#  include <unistd.h>
#endif

static hitz_module_t harkd_hitz_module;
hitz_module_t *HARKD_HITZ_MODULE = &harkd_hitz_module;
/* ------------------------------------------------------------------------------------
---- */
int harkd_cmd_ports(hitz_t *h,hitz_module_t **module,const char *argv[]) {
     char buffer[512] = {0};
     if(harkd_serial_ports(buffer,sizeof(buffer)-1)==HARKD_OK) {
          hitz_fprintf(h,1,"Serial ports:\n%s",buffer);
     }
     return 1;
}
int harkd_cmd_interfaces(hitz_t *h,hitz_module_t **module,const char *argv[]) {
     HARKD_ARRAY_FOREACH(const harkd_dev_itf_t,itf,harkd_dev_itf_array) {
          if(argv[1]) {
               if(!strcasecmp(itf->name,argv[1])) {
                    harkd_printf(1,"## %s\n\n%s\n\n%s\n\n",itf->name,itf->help,itf-
>long_help);
               }
          } else {
```

```
                harkd_printf(1,"%-20s %s\n",itf->name,itf->help);
           }
      }
      return 1;
}
int harkd_cmd_new(hitz_t *h,hitz_module_t **module,const char *argv[]) {
      harkd_dev_obj_t *harkd = NULL; const harkd_dev_itf_t *def;
      if(!argv[1]) {
           harkd_printf(1,"new [NAME|list] [DRIVER|COMMAND|delete] [search]
[OPTIONS...]\n");
           return 0;
      }
      if(!argv[2]) {
           if(!strcasecmp(argv[1],"list")) {
                HARKD_LIST_FOREACH(harkd_dev_obj_t,d,harkd_dev_obj_list) {
                     hitz_fprintf(h,1,"%-20s %-20s %s\n",d->name,d->itf->name,d-
>portname);
                }
           } else if((harkd = harkd_get(argv[1]))) {
                const char *help[] = {"help",NULL};
                harkd_run(harkd,help);
           } else {
                harkd_error(NULL,"Please select an interface.");
                harkd_cmd_interfaces(h,module,argv+1);
           }
           return 0;
      }
      def = harkd_get_itf(argv[2]);
      if(def) {
           if(!argv[3]) {
                char buffer[512];
                harkd_list_ports(def,buffer,sizeof(buffer)-1);
                if(!buffer[0]) {
                     harkd_error(h,"No available ports.");
                } else {
                     hitz_fprintf(h,3,"Please select a port:\n");
                     hitz_fprintf(h,1,"%s",buffer);
                }
           } else if(!strcasecmp(argv[3],"search")) {
                harkd = harkd_new(argv[1],def,NULL,argv+4);
           } else {
                harkd = harkd_new(argv[1],def,argv[3],argv+4);
           }
      } else if(!(harkd = harkd_get(argv[1]))) {
           harkd_error(h,"Device `%s` not found.",argv[1]);
      } else if(!strcasecmp(argv[2],"delete")) {
           harkd_free(harkd);
      } else {
           harkd_run(harkd,argv+2);
      }
      return 1;
}
int harkd_cmd_table(hitz_t *h,hitz_module_t **module,const char *argv[]) {
      if(!argv[1]) {
           hitz_fprintf(h,1,"%s DEV/VAR[=VAL] ...\n",argv[0]);
           return 1;
      }
      int pl = 0;
      for(char **argp=(char**)argv+1;*argp;argp++) {
           char            *dev   = strtok(*argp,"/");
           harkd_dev_obj_t *harkd = harkd_get(dev);
```

```
          if(!harkd) {
               harkd_error("Device ‘%s‘ not found.",dev);
               continue;
          }
          double dval;
          char  *svar = strtok(NULL,"=");
          if(!svar) {
               harkd_error(h,"Specify a variable.");
               return 1;
          }
          for(char *p=svar;*p;p++) *p = toupper(*p);
          char  *sval = strtok(NULL,"\n");
          if(sval) {
               dval = atof(sval);
               if(harkd_var_set(harkd,svar,&dval)!=HARKD_OK) return 1;
          } else {
               if(harkd_var_get(harkd,svar,&dval)!=HARKD_OK) return 1;
               hitz_fprintf(h,1,"%10.4f,",dval); pl = 1;
          }
     }
     if(pl) hitz_fprintf(h,1,"\n");
     return 1;
}
int harkd_cmd_test(hitz_t *h,hitz_module_t **module,const char *argv[]) {
     HARKD_ARRAY_FOREACH(const harkd_test_t,t,harkd_test_array) {
          if(!argv[1]) {
               hitz_fprintf(h,1,"%-15s %s\n",t->name,t->description);
          } else if((strcasecmp(t->name,argv[1]))) {

          } else {
               harkd_test_run(t,argv+2);
          }
     }
     return 1;
}
static hitz_module_t harkd_hitz_module = {
     "harkd",
     NULL,
     {
          {"ports"      ,"List available ports."            ,harkd_cmd_ports},
          {"new"        ,"Open device."                     ,harkd_cmd_new},
          {"table"      ,"Get/set values from/to the device",harkd_cmd_table},
          {"interfaces" ,"Show available device drivers."   ,harkd_cmd_interfaces},
          {"test"       ,"Run a predefined test."           ,harkd_cmd_test},
          {NULL}
     }
};


void harkd_wait(int ms) {
#    ifdef _WIN32
     Sleep(ms);
#    else
     usleep(ms*1000);
#    endif
}
```

Source file: **harkdc/src/tests/example.c**

```
#include "../harkd.h"
#include <hitz/hitz.h>
harkd_r harkd_example_test (harkd_test_var_t *vars) {
     FOREACH_TEST_VARIABLES(v,vars) {
          harkd_printf(1,"%s=%s\n",v->name,v->value);
     }
     return HARKD_OK;
}
const harkd_test_t HARKD_EXAMPLE_TEST = {
     "example",
     "Simply prints the variables specified.",
     "Simply prints the variables specified.",
     {{"VAR1","VAL1"},
      {NULL  ,NULL  }},
     harkd_example_test
};
```

Source file: **harkdc/src/tests/dcdc.c**

```
#include "../harkd.h"
#include <string.h>
#include <stdlib.h>
#ifndef NL
#  define NL "\n"
#endif
harkd_r harkd_dcdc_test (harkd_test_var_t *vars) {
     harkd_dev_obj_t *l=NULL,*s=NULL; harkd_table_t *t = NULL;
     double Vin=0,Imax=0; int N=1,Tstep=0;
     double Istep=0,Isupply=0,Vsupply=0,Iload=0,Vload=0,Imaxin=5;
     const char
          *Iout   = "I1",
          *Vout   = "V1",
          *Output = "dcdc.xlsx";
     FOREACH_TEST_VARIABLES(v,vars) {
          if(!strcasecmp(v->name,"Vin")) {
               Vin = atof(v->value);
          } else if(!strcasecmp(v->name,"Imax")) {
               Imax = atof(v->value);
          } else if(!strcasecmp(v->name,"Imaxin")) {
               Imaxin = atof(v->value);
          } else if(!strcasecmp(v->name,"N")) {
               N = atoi(v->value);
          } else if(!strcasecmp(v->name,"Tstep")) {
               Tstep = atoi(v->value);
          } else if(!strcasecmp(v->name,"output")) {
               Output = v->value;
          } else if(!strcasecmp(v->name,"CH")) {
               int num = atoi(v->value);
               if(num==1) {
                    Iout = "I1";
                    Vout = "V1";
               } else if(num==2) {
                    Iout = "I2";
                    Vout = "V2";
               } else {
                    goto error;
               }
          } else {
               harkd_error(NULL,"Invalid variable ‘%s‘.",v->name);
               goto error;
          }
     }
     l = harkd_new("load"  ,harkd_get_itf("ARRAY-371X"),NULL,NULL);
     s = harkd_new("supply",harkd_get_itf("MPD-3305D") ,NULL,NULL);
     if(!l) { harkd_error(NULL,"Can't find a compatible load."  ); goto error; }
     if(!s) { harkd_error(NULL,"Can't find a compatible supply."); goto error; }
     t = harkd_table_new(Output);
     harkd_table_add_string(t,"I(in)");
     harkd_table_add_string(t,"V(in)");
     harkd_table_add_string(t,"I(out)");
     harkd_table_add_string(t,"V(out)");
```

```
     harkd_table_add_string(t,"P(in)");
     harkd_table_add_string(t,"P(out)");

     harkd_table_add_chart(t,"V(out);I(out)","Sheet1!$D$2:$D$%i;Sheet1!$C$2:$C$%i",N+1,
N+1);
     harkd_table_next(t);
     Istep = Imax/N*1;
     harkd_var_set(s,Iout,&Imaxin);
     harkd_var_set(s,Vout,&Vin);
     for(double I=0;I<Imax;I+=Istep) {
          harkd_var_set(l,"I",&I);
          harkd_wait(Tstep);
          harkd_var_get(l,"I",&Iload);
          harkd_var_get(l,"V",&Vload);
          harkd_var_get(s,Iout,&Isupply);
          harkd_var_get(s,Vout,&Vsupply);
          harkd_table_add(t,Isupply);
          harkd_table_add(t,Vsupply);
          harkd_table_add(t,Iload);
          harkd_table_add(t,Vload);
          /*
          int r = harkd_table_row(t)+1;
          harkd_table_add_formula(t,"=A%i*B%i",r,r);
          harkd_table_add_formula(t,"=C%i*D%i",r,r);
          */
          harkd_table_add(t,Isupply* Vsupply);
          harkd_table_add(t,Iload  * Vload);
          harkd_table_next(t);
     }
     if(s) harkd_free(s);
     if(l) harkd_free(l);
     if(t) harkd_table_free(t);
     return HARKD_OK;
error:
     if(l) harkd_free(l);
     if(s) harkd_free(s);
     return HARKD_ERR;
}
const harkd_test_t HARKD_DCDC_TEST = {
     "DC/DC",
     "Measure the efficiency of a DC/DC converter.",

     "This test measures the efficiency of a DC/DC converter" NL
     "using an ‘electronic load‘ and a ‘DC power supply‘."    NL
     ,
     {
          {"Vin"    ,"12"   ,"V (List separated by commas)"},
          {"Imax"   ,"2"    ,"A"},
          {"N"      ,"10"   ,"",
           {"10","20","40","80",NULL}},
          {"Tstep"  ,"2000"     ,"ms (Time between measurements)"},
          {"CH"     ,"1"        ,"Channel to use."},
          {"Output" ,"dcdc.xlsx","Output file."},
          {NULL}
     },
     harkd_dcdc_test
};
```

Source file: **harkdc/src/harkd-table.c**

```
#include "harkd.h"
#include <xlsxwriter.h>
#include <hitz/hitz.h>
#include <stdarg.h>
/* ------------------------------ TABLE ---------------------------------- */
struct harkd_table_s {
#    ifndef DISABLE_XLSX
     lxw_workbook  *workbook;
     lxw_worksheet *worksheet;
     lxw_chart     *chart;
#    endif
     int row,col,max_col;
};
harkd_table_t *harkd_table_new(const char *filename) {
     harkd_table_t *t = malloc(sizeof(*t));
#    ifndef DISABLE_XLSX
     t->workbook  = workbook_new(filename);
     t->worksheet = workbook_add_worksheet(t->workbook,NULL);
     t->chart     = NULL;
#    endif
     t->row = 0;
     t->col = 0;
     t->max_col = 0;
     return t;
}
void harkd_table_free(harkd_table_t *t) {
     if(t) {
#         ifndef DISABLE_XLSX
          if(t->chart) {
               worksheet_insert_chart(t->worksheet,0,t->max_col, t->chart);
          }
          workbook_close(t->workbook);
#         endif
          free(t);
     }
}
void harkd_table_add(harkd_table_t *t,double d) {
     harkd_printf(1,"%-15f",d);
#    ifndef DISABLE_XLSX
     worksheet_write_number(t->worksheet, t->row, t->col, d, NULL);
#    endif
     harkd_table_next_column(t);
}
void harkd_table_add_string(harkd_table_t *t,const char *s) {
```

```
     harkd_printf(1,"%-15s",s);
#    ifndef DISABLE_XLSX
     worksheet_write_string(t->worksheet, t->row, t->col, s, NULL);
#    endif
     harkd_table_next_column(t);
}
void harkd_table_add_formula(harkd_table_t *t,const char *format,...) {
#    ifndef DISABLE_XLSX
     char s[64];
     va_list args;
     va_start(args, format);
     vsprintf(s,format,args);
     va_end(args);
     harkd_printf(1,"%-15s",s);
     worksheet_write_formula(t->worksheet, t->row, t->col, s, NULL);
#    endif
     harkd_table_next_column(t);
}
void harkd_table_next(harkd_table_t *t) {
     t->row++;
     t->col = 0;
     harkd_printf(1,"\n");
     fflush(stdout);
}
void harkd_table_next_column(harkd_table_t *t) {
     harkd_printf(1,",");
     fflush(stdout);
     if((++t->col)>(t->max_col))
          t->max_col = t->col;
}

int  harkd_table_column(harkd_table_t *t) { return t->col; }
int  harkd_table_row   (harkd_table_t *t) { return t->row; }
void harkd_table_add_chart (harkd_table_t *t,const char *names,const char *yx,...) {
     char s[128];
     va_list args;
     va_start(args, yx);
     vsprintf(s,yx,args);
     va_end(args);
     const char *y = NULL,*x = NULL;
     y = strtok(s,";");
     if(y) x = strtok(NULL,";");
     if(!t->chart) {
          t->chart = workbook_add_chart(t->workbook, LXW_CHART_LINE);
     }
     /* Configure the chart. */
     chart_add_series(t->chart, x , y);
     if(names) {
          strcpy(s,names);
          if((y = strtok(s,";")))
               x = strtok(NULL,";");
          else
               x = NULL;
          if(y) chart_axis_set_name(t->chart->y_axis,y);
          if(x) chart_axis_set_name(t->chart->x_axis,x);
     }
}
```

Source file: **harkdc/src/devices/checksum.c**

```
#include "checksum.h"
u_int8_t checksum8(u_int8_t *array,u_int8_t len) {
     u_int8_t chk = 0;
     for(u_int8_t i=0;i<len;i++) {
          chk = (chk + array[i]);
     }
     return chk;
}
```

Source file: **harkdc/src/devices/mpd.c**

```
/**
 *
 *
 * > minicom -b 9600 -D /dev/ttyUSB0 -8
 */

#include "../harkd.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
harkd_r harkd_mpd_init(harkd_dev_obj_t *harkd,const char *port,const char *args[]) {
     char buffer[512];
     if(harkd_serial_open(harkd,port)==NULL)                  goto io_error;
     if(harkd_serial_puts(harkd,"\n*idn?\n")!=HARKD_OK)       goto io_error;
     if(harkd_serial_gets(harkd,buffer,sizeof(buffer))==NULL) goto another_device;
     if(harkd_serial_puts(harkd,"OUT1\n"   )!=HARKD_OK)       goto io_error;
     if(harkd_serial_puts(harkd,"TRACK0\n" )!=HARKD_OK)       goto io_error;
```

```
     if(strcasecmp("SN:V1.81",buffer))                        goto another_device;
     return HARKD_OK;
io_error:
     return HARKD_ERR;
another_device:
     return HARKD_ERR;
}
harkd_r harkd_mpd_command(harkd_dev_obj_t *harkd,const char *args[]) {
     return HARKD_OK;
}
harkd_r harkd_mpd_get (harkd_dev_obj_t *harkd,const char *var,double *val) {
     char buffer[64];
     if(!(var[0]=='V'||var[0]=='I')) goto variable_not_available;
     if(!(var[1]=='1'||var[1]=='2')) goto variable_not_available;
     sprintf(buffer,"%cOUT%c?\n",var[0],var[1]);
     if(harkd_serial_puts(harkd,buffer)!=HARKD_OK) goto io_error;
     if(harkd_serial_gets(harkd,buffer,sizeof(buffer))==NULL) goto io_error;
     *val = atof(buffer);
     return HARKD_OK;
variable_not_available:
     harkd_error_invalid_variable(harkd,var);
     return HARKD_ERR;
io_error:
     return HARKD_ERR;
}
harkd_r harkd_mpd_set (harkd_dev_obj_t *harkd,const char *var,double *val) {
     char buffer[64];
     if(!(var[0]=='V'||var[0]=='I')) goto variable_not_available;
     if(!(var[1]=='1'||var[1]=='2')) goto variable_not_available;
     sprintf(buffer,"%cSET%c:%.3f" "\n",var[0],var[1],*val);
     if(harkd_serial_puts(harkd,buffer)!=HARKD_OK) goto io_error;
     //harkd_wait(200);
     return HARKD_ERR;
variable_not_available:
     harkd_error_invalid_variable(harkd,var);
     return HARKD_ERR;
io_error:
     return HARKD_ERR;
}
harkd_r harkd_mpd_clear(harkd_dev_obj_t *harkd) {
     return harkd_serial_puts(harkd,
                              "OUT0" "\n");
}
const harkd_dev_itf_t HARKD_DEVICE_MPD3305D = {
     "MPD-3305D","30V 5A USB Interface Programmable Power Supply",'s',
     harkd_help_outputs("I1,I2,V1,V2")
     harkd_help_inputs("I1,I2,V1,V2")
     ,
     {
          0,
          harkd_mpd_init,
          harkd_mpd_command,
          harkd_mpd_clear,
          harkd_mpd_set,
          harkd_mpd_get
     }
};
```

Source file: **harkdc/src/devices/example.c**

```
#include "../harkd.h"
#include <string.h>
#include <stdlib.h>
#include <hitz/hitz.h>
typedef struct {
     double var1;
} harkd_example_device_t ;
harkd_r harkd_example_init(harkd_dev_obj_t *harkd,const char *port,const char *args[]) {
     harkd_example_device_t *udata = harkd_itf_data(harkd,harkd_example_device_t);
     if(args[0]) { udata->var1 = atof(args[0]); }
     harkd_printf(1,"Starting example device...\n");
     return HARKD_OK;
}
harkd_r harkd_example_clear(harkd_dev_obj_t *harkd) {
```

```
     harkd_printf(1,"Deleting example device...\n");
     return HARKD_OK;
}
harkd_r harkd_example_command(harkd_dev_obj_t *harkd,const char *args[]) {
     if(!strcasecmp(args[0],"hello")) {
          for(int i=1;args[i];i++) harkd_printf(1," %s",args[i]);
          harkd_printf(1,"\n");
          return HARKD_OK;
     } else {
          harkd_error_invalid_command(harkd,args[0]);
          return HARKD_ERR;
     }
}
harkd_r harkd_example_set (harkd_dev_obj_t *harkd,const char *var,double *val) {
     if(!strcasecmp(var,"var1")) {
          harkd_example_device_t *d = harkd_itf_data(harkd,harkd_example_device_t);
          d->var1 = *val;
          return HARKD_OK;
     } else {
          harkd_error_invalid_variable(harkd,var);
          return HARKD_ERR;
     }
}
harkd_r harkd_example_get (harkd_dev_obj_t *harkd,const char *var,double *val) {
     if(!strcasecmp(var,"var1")) {
          harkd_example_device_t *d = harkd_itf_data(harkd,harkd_example_device_t);
          *val = d->var1;
          return HARKD_OK;
     } else {
          harkd_error_invalid_variable(harkd,var);
          return HARKD_ERR;
     }
}




const harkd_dev_itf_t HARKD_DEVICE_EXAMPLE = {
     "example","An example device.",'n',
     harkd_help_paragraph(
          "This is an example virtual device. Driver developers"
          "can copy paste it's code as a template for new drivers.")
     harkd_help_inputs("VAR1")
     harkd_help_outputs("VAR1")
     harkd_help_commands()
     harkd_help_command("hello","Some text","This command simply echos the input")
     ,
     {

          sizeof(harkd_example_device_t),
          harkd_example_init,
          harkd_example_command,
          harkd_example_clear,
          harkd_example_set,
          harkd_example_get
     }
};
```

Source file: **harkdc/src/devices/protocols.c**

```
#include "protocols.h"
#include "checksum.h"
#include "../harkd.h"
#include <string.h>
harkd_r msg26_send(harkd_dev_obj_t *harkd,msg26_t *msg,u_int8_t addr) {
     msg->flag = 0xAA;
     msg->addr = addr;
     msg->chksum = checksum8((u_int8_t*)msg,25);
     /*
     for(int i=0;i<26;i++) {
          fprintf(stderr,"%02x:",((u_int8_t*)msg)[i]);
     }
     fprintf(stderr,"\n");
     */
     return harkd_serial_write(harkd,msg,sizeof(*msg));
}
harkd_r msg26_recv(harkd_dev_obj_t *harkd,msg26_t *msg) {
     while(1) {
          if(harkd_serial_read(harkd,msg,1)!=HARKD_OK) return HARKD_ERR;
          if(msg->flag == 0xAA) break;
     }
     harkd_r r = harkd_serial_read(harkd,&(msg->addr),sizeof(*msg)-1);
     if(r!=HARKD_OK) return r;
     u_int8_t chk = checksum8((u_int8_t*)msg,25);
     if(msg->chksum!=chk) {
          harkd_error(harkd,"371X: Checksum error.");
          return HARKD_ERR;
     }
     return HARKD_OK;
}
void    msg26_init(msg26_t *msg,u_int8_t cmd) {
     memset(msg,0,sizeof(*msg));
     msg->cmd = cmd;
}
```

Source file: **harkdc/src/devices/array.c**

```
/**
 *
 */
#include "../harkd.h"
#include "protocols.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MSG_371X_CMD_SET          0x90
#define MSG_371X_CMD_GET          0x91
#define MSG_371X_CMD_SET_ON_OFF   0x92
#define MSG_371X_CMD_SET_ODD_1_5  0x93
#define MSG_371X_CMD_SET_ODD_6_10 0x94
#define MSG_371X_CMD_START        0x95
#define MSG_371X_CMD_STOP         0x96

/* ------------------------------ Communication protocol ------------------------ */
typedef struct __attribute__((__packed__)) {
     u_int16_t IM;
     u_int16_t PM;
     u_int8_t  addr;
     u_int8_t  mod;
#    define MSG_371X_SET_CURRENT    0x01
#    define MSG_371X_SET_POWER      0x02
#    define MSG_371X_SET_RESISTANCE 0x03
     u_int16_t value;
} msg_371x_info_set_t;
typedef struct __attribute__((__packed__)) {
     u_int16_t I;
     u_int16_t V; u_int16_t V2;
     u_int16_t P;
     u_int16_t IM;
     u_int16_t PM;
     u_int16_t R;
     u_int8_t  os;
} msg_371x_info_get_t;



harkd_r msg_371x_send_ON(harkd_dev_obj_t *harkd,u_int8_t addr) {
     msg26_t msg; msg26_init(&msg,MSG_371X_CMD_SET_ON_OFF);
     msg.info[0] = 0b00000011;
     return msg26_send(harkd,&msg,addr);
}
harkd_r msg_371x_send_R(harkd_dev_obj_t *harkd,u_int8_t addr,double R,double IM,double
PM) {
     msg26_t msg; msg26_init(&msg,MSG_371X_CMD_SET);
     msg_371x_info_set_t *set = (msg_371x_info_set_t *) msg.info;
     set->IM    = (u_int16_t) ((IM) * 30000/30 );
     set->PM    = (u_int16_t) ((PM) * 2000/200 );
     set->mod   = MSG_371X_SET_RESISTANCE;
     set->value = (u_int16_t) ( (R) * 50000/500 );
     return msg26_send(harkd,&msg,addr);
}
harkd_r msg_371x_send_I(harkd_dev_obj_t *harkd,u_int8_t addr,double I,double IM,double
PM) {
     msg26_t msg; msg26_init(&msg,MSG_371X_CMD_SET);
     msg_371x_info_set_t *set = (msg_371x_info_set_t *) msg.info;
```

```
     set->IM    = IM * (30000/30 );
     set->PM    = PM * (2000/200 );
     set->mod = MSG_371X_SET_CURRENT;
     set->value = (u_int16_t) ( (I) * 30000/30 );
     return msg26_send(harkd,&msg,addr);
}
harkd_r msg_371x_send_P(harkd_dev_obj_t *harkd,u_int8_t addr,double P,double IM,double
PM) {
     msg26_t msg; msg26_init(&msg,MSG_371X_CMD_SET);
     msg_371x_info_set_t *set = (msg_371x_info_set_t *) msg.info;
     set->IM    = IM * (30000/30 );
     set->PM    = PM * (2000/200 );
     set->mod = MSG_371X_SET_POWER;
     set->value = (u_int16_t) ( (P) * 2000/200 );
     return msg26_send(harkd,&msg,addr);
}
harkd_r msg_371x_send_recv_get(harkd_dev_obj_t *harkd,u_int8_t addr,
                               double *I ,double *V ,double *P,
                               double *IM,double *PM,double *R) {
     harkd_r r;
     msg26_t msg; msg26_init(&msg,MSG_371X_CMD_GET);
     msg_371x_info_get_t *get = (msg_371x_info_get_t *) msg.info;
     r = msg26_send(harkd,&msg,addr);
     if(r!=HARKD_OK) return r;
     r = msg26_recv(harkd,&msg);
     if(r!=HARKD_OK) return r;
     if(I)  *I  = (double)get->I  * 30 /30000  ;
     if(V)  *V  = (double)get->V  * 360/360000 ;
     if(P)  *P  = (double)get->P  * 200 /2000  ;
     if(IM) *IM = (double)get->IM * 30  /30000 ;
     if(PM) *PM = (double)get->PM * 200 /2000  ;
     if(R)  *R  = (double)get->R  * 500/50000  ;
     return HARKD_OK;
}
```

```
/* ---------------------------- Device ----------------------------------- */
typedef struct {
     u_int8_t addr;
     double I,V,P,IM,PM,R;
} harkd_371x_t;
harkd_r harkd_array_init(harkd_dev_obj_t *harkd,const char *port,const char *args[]) {
     harkd_371x_t *d = harkd_itf_data(harkd,harkd_371x_t);
     d->addr = 0x0;
     d->IM   = 30;
     d->PM   = 200;
     if(args[0]) {
          if(args[1]) {
```

```
            d->addr = atoi(args[1]);
        }
    }
    if(harkd_serial_open(harkd,port)==NULL)                          goto io_error;
    if(msg_371x_send_ON(harkd,d->addr)!=HARKD_OK)                    goto io_error;
    if(msg_371x_send_I(harkd,d->addr,0,d->IM,d->PM)!=HARKD_OK)       goto io_error;
    if(msg_371x_send_recv_get
       (harkd,d->addr,
        &d->I,&d->V,&d->P,&d->IM,&d->PM,&d->R)!=HARKD_OK) goto another_device;

    return HARKD_OK;
io_error:
    return HARKD_ERR;
another_device:
    return HARKD_ERR;
}
harkd_r harkd_array_command(harkd_dev_obj_t *harkd,const char *args[]) {
    return HARKD_OK;
}
harkd_r harkd_array_get (harkd_dev_obj_t *harkd,const char *var,double *val) {
    harkd_371x_t *d = harkd_itf_data(harkd,harkd_371x_t);
    harkd_r r = msg_371x_send_recv_get(harkd,d->addr,&d->I,&d->V,&d->P,&d->IM,&d->PM,
&d->R);
    if(r!=HARKD_OK) return r;
    if(!strcasecmp(var,"I"))         { *val = d->I;
    } else if(!strcasecmp(var,"V"))  { *val = d->V;
    } else if(!strcasecmp(var,"P"))  { *val = d->P;
    } else if(!strcasecmp(var,"IM")) { *val = d->IM;
    } else if(!strcasecmp(var,"PM")) { *val = d->PM;
    } else if(!strcasecmp(var,"R"))  { *val = d->R;
    } else {
        harkd_error_invalid_variable(harkd,var);
        return HARKD_ERR;
    }
    return r;
}
harkd_r harkd_array_set (harkd_dev_obj_t *harkd,const char *var,double *val) {
    harkd_r r; harkd_371x_t *d = harkd_itf_data(harkd,harkd_371x_t);
    if(!strcasecmp(var,"R")) {
        r = msg_371x_send_R(harkd,d->addr,*val,d->IM,d->PM);
    } else if(!strcasecmp(var,"I")) {
        r = msg_371x_send_I(harkd,d->addr,*val,d->IM,d->PM);
    } else if(!strcasecmp(var,"P")) {
        r = msg_371x_send_P(harkd,d->addr,*val,d->IM,d->PM);
    } else if(!strcasecmp(var,"IM")) {
        d->IM = *val;
    } else if(!strcasecmp(var,"PM")) {
        d->PM = *val;
    } else {
        harkd_error_invalid_variable(harkd,var);
        return HARKD_ERR;
    }
    return r;
}


harkd_r harkd_array_clear(harkd_dev_obj_t *harkd) {
    return msg_371x_send_I(harkd,0x0,0,30,200);
}
```

```
const harkd_dev_itf_t HARKD_DEVICE_ARRAY_371X = {
     "ARRAY-371X",
     "Single input programmable DC electronic load.",
     's',
     harkd_help_inputs("R,I,P,IM,IP,V")
     harkd_help_outputs("R,I,P,IM,IP")
     ,
     {
          sizeof(harkd_371x_t),
          harkd_array_init,
          harkd_array_command,
          harkd_array_clear,
          harkd_array_set,
          harkd_array_get
     }
};
```

Source file: **harkdc/src/harkd-main.c**

```
#include "harkd.h"
#include <hitz/hitz.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <unistd.h>
#ifndef HARKD_COPYRIGHT
#  define HARKD_COPYRIGHT "Copyright (c) 2018, Harkaitz Agirre ezama, All rights
reserved."
#endif
static const harkd_test_t *test      = NULL;
static const char        **test_opts = NULL;
void *main_client (hitz_t *hitz) {

     if(test) {
          harkd_test_run(test,test_opts);
     } else {
          hitz_loop_and_clean(hitz);
     }

     return NULL;
}
```

```
int main_func(int argc,const char *argv[]) {
     const char *help =
          "Usage: harkdc [TEST ARGS...]"                          NL
          ""                                                      NL
          "HARKDC is a tool to manage some electronic instruments" NL
          "programatically with a PC."                            NL
          ""                                                      NL
          HARKD_COPYRIGHT                                         NL
          ""                                                      NL
          ;
     const char *wellcome =
          "  ___________                                                  " NL
          " | .^.^.^.^. | HARKD Terminal Interface                        " NL
          " |-----.-----|                                                 " NL
          " |HARKD: .oo | Type ‘help‘ to see a list of commands.          " NL
          " |-----’ .oo |                                                 " NL
          " |___________| Run the program with ‘-h‘ argument for runtime  " NL
          "               options.                                       " NL
          "                                                              " NL
          HARKD_COPYRIGHT NL
          ""                                                              NL
          ;
     hitz_t hitz; int ret = 1;
     hitz_init(&hitz,NULL,NULL);
     harkd_library_init(&hitz);
     if(argc>1) {
          if(!strcasecmp(argv[1],"-h")) {
               printf("%s",help);
          } else {
               test_opts = argv+2;
               HARKD_ARRAY_FOREACH(const harkd_test_t,t,harkd_test_array) {
                    if(!strcasecmp(t->name,argv[1])) { test = t; }
               }
               if(!test) {
                    HARKD_ARRAY_FOREACH(const harkd_test_t,t,harkd_test_array) {
                         hitz_fprintf(&hitz,1,"%-15s %s\n",t->name,t->description);
                    }
                    hitz_fprintf(&hitz,1,"Please select a valid test.");
                    goto err;
               }
               main_client (&hitz);
          }
     } else {
          hitz_fprintf(&hitz,HITZ_STDTTY,"%s" NL,wellcome);
          main_client (&hitz);

     }
     ret = 0;
err:
     harkd_library_clean();
     fflush(stdout);
     fflush(stderr);
     return ret;
}

int main(int argc,const char *argv[]) { return main_func(argc,argv); }
```

Source file: **harkdc/src/harkd.c**

```
#include <stdio.h>
#include <hitz/hitz.h>
#include <libserialport.h>
#ifndef DISABLE_XLSX
#  include <xlsxwriter.h>
#endif

#include <string.h>
#include <stdarg.h>

#include "harkd.h"




/* ------------------------------------------------------------ */
void harkd_list_ports(const harkd_dev_itf_t *device,char buffer[],int len) {
     buffer[0] = '\0';
     if(device->port_type=='s') {
```

```
          harkd_serial_ports(buffer,len);
     } else {
          strncat(buffer,"open\n",len);
     }
}



/* ------------------------------------------------------------ */
harkd_dev_obj_t *harkd_new(const char            *name,
                           const harkd_dev_itf_t *itf,
                           const char            *portname,
                           const char *args[]) {

     harkd_dev_obj_t *harkd = NULL;
     /* 1.- Check arguments. */
     if(!name)   { harkd_error(NULL,"Please specify a name."); return NULL; }
     if(!itf)    { harkd_error(NULL,"Invalid interface.");     return NULL; }


     /* 2- If the device exists, close. */
     HARKD_LIST_FOREACH(harkd_dev_obj_t,d,harkd_dev_obj_list) {
          if(!strcasecmp(name,d->name)) {
               *_nxt = d->next;
               harkd_free(d);
          }
     }

     /* 3.- Create object. */
     harkd = malloc(sizeof(*harkd)+itf->fn.len);
     memset(harkd,0,sizeof(*harkd)+itf->fn.len);
     strncpy(harkd->name,name,sizeof(harkd->name)-1);
     harkd->itf   = itf;
     harkd->portname[0] = '\0';

     harkd->port  = NULL;
     harkd->next  = NULL;

     /* 4.- Search a port. */
     char buffer[HARKD_MAX_TEXT] = {0};
     if(portname) {
          strncpy(buffer,portname,sizeof(buffer)-1);
     } else {
          harkd_list_ports(harkd->itf,buffer,sizeof(buffer)-1);
     }
     char *found   = NULL;
     const char *args2[] = {NULL};
     for(char *r,*s = strtok_r(buffer,"\n,",&r);s;s = strtok_r(NULL,"\n,",&r)) {
          harkd_log(1,"Trying to open `%s` ...",s);
          if(!harkd->itf->fn.init) {
               found = s;
               break;
          } else if(harkd->itf->fn.init(harkd,s,(args)?args:args2)>=0) {
               found = s;
               break;
          }
     }

     /* 5.- Can't search a valid port. */
     if(!found) {
```

```
          harkd_error(NULL,"Can't open any port.");
          goto error;
     }

     /* 6.- Add to list. */
     harkd_log(1,"Add [harkd_dev_obj_t *%p] to device objects list.",harkd);
     harkd->next = harkd_dev_obj_list;
     harkd_dev_obj_list = harkd;

     return harkd;
error:
     if(harkd) harkd_free(harkd);
     return NULL;
}
harkd_dev_obj_t *harkd_get (const char *name) {
     HARKD_LIST_FOREACH(harkd_dev_obj_t,d,harkd_dev_obj_list) {
          if(!strcasecmp(name,d->name)) {
               return d;
          }
     }
     return NULL;
}
void     harkd_free(harkd_dev_obj_t *harkd) {
     if(!harkd) return;
     if(harkd->itf->fn.clear)
          harkd->itf->fn.clear(harkd);
     harkd_close(harkd);
}
harkd_r harkd_run      (harkd_dev_obj_t *harkd,const char *args[]) {
     const char *d_args[] = {"info",NULL};
     if(harkd->itf->fn.command) {
          return harkd->itf->fn.command(harkd,(args)?args:d_args);
     } else {
          return HARKD_OK;
     }
}
harkd_r harkd_var_set  (harkd_dev_obj_t *harkd,const char *var,double *val) {
     if(harkd->itf->fn.set) {
          return harkd->itf->fn.set(harkd,var,val);
     } else {
          harkd_error(harkd,"Setting values is not supported for this device.");
          return HARKD_ERR;
     }
}
harkd_r harkd_var_get  (harkd_dev_obj_t *harkd,const char *var,double *val) {
     if(harkd->itf->fn.get) {
          return harkd->itf->fn.get(harkd,var,val);
     } else {
          harkd_error(harkd,"Getting values is not supported for this device.");
          return HARKD_ERR;
     }
}
void     harkd_close    (harkd_dev_obj_t *harkd) {
     if(harkd->port) {
          sp_close(harkd->port);
          sp_free_port(harkd->port);
          harkd->port = NULL;
     }
     harkd->portname[0] = '\0';
}
int      harkd_is_open  (harkd_dev_obj_t *harkd) {
```

```
     return (harkd->port)?1:0;
}
int      harkd_port_is_openned(const char *portname) {
     HARKD_LIST_FOREACH(harkd_dev_obj_t,d,harkd_dev_obj_list) {
          if(!strcasecmp(d->portname,portname)) return 1;
     }
     return 0;
}
```

```
/* ------------------------------------------------------------------------------
- */
harkd_r harkd_test_run(const harkd_test_t *t,const char *i_opts[]) {
     harkd_test_var_t opts[30]; int o;
     for(o=0;t->opts[o].name && o<29;o++) {
          opts[o].name  = t->opts[o].name;
          opts[o].value = (t->opts[o].value)?t->opts[o].value:"";
     }
     if(!i_opts) {} else if(!i_opts[0]) {
     } else if (!strcasecmp(i_opts[0],"help")) {
          harkd_printf(1,"## %s\n\n%s\n\n",t->name,t->long_description);
          harkd_printf(1,"+ Variables;\n\n");
          for(int j=0;t->opts[j].name;j++) {
               harkd_printf(1,"  + %-10s = %s\n",t->opts[j].name,t->opts[j].value);
          }
          harkd_printf(1,"\n");
          return HARKD_OK;
     } else {
          for(int k=0,j;i_opts[k];k++) {
               char *var = (char*)i_opts[k],*val="",*valn;
               if(!var) continue;
               if((valn = strchr(var,'='))) {
                    *valn='\0'; valn++; val = valn;
               }
               for(j=0;j<o;j++) {
                    if(!strcasecmp(var,opts[j].name)) {
                         opts[j].name  = var;
                         opts[j].value = val;
                         break;
                    }
               }
               if(j==o && o<29) {
                    opts[o].name  = var;
                    opts[o].value = val;
                    o++;
               }
          }
     }
     opts[o].name = NULL;
     return t->fun(opts);
}
```

Source file: **harkdc/src/harkd-core.c**

```
#include "harkd.h"
#include "devices/devices.h" /* harkd_device_list  */
#include "tests/tests.h"     /* harkd_tests        */
#include <hitz/hitz.h>
#include <hitz/hitz-commands.h>
#include <stdarg.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

hitz_t                 *hitz = NULL;
const harkd_dev_itf_t **harkd_dev_itf_array = harkd_device_list;
const harkd_test_t    **harkd_test_array    = harkd_tests;
harkd_dev_obj_t        *harkd_dev_obj_list  = NULL;
extern hitz_module_t   *HARKD_HITZ_MODULE;

/* ------------------------------------------------------------------------------
------------
 * ---- LIBRARY MANAGEMENT ------------------------------------------------------
------------
 * ------------------------------------------------------------------------------
------------ */
static int harkd_library_inited = 0;
static hitz_t s_hitz;
void harkd_library_init(hitz_t *h) {
     if(harkd_library_inited++) return;
     hitz_init(&s_hitz,NULL,NULL);
     hitz_add (&s_hitz,HARKD_HITZ_MODULE);
     hitz_add (&s_hitz,&HITZ_MODULE);
     if(h) {
          hitz = h;
          hitz_add (h,HARKD_HITZ_MODULE);
          hitz_add (h,&HITZ_MODULE);
     }
```

```
     return;
}
void harkd_library_clean(void) {
     if(--harkd_library_inited) return;
}
const harkd_dev_itf_t *harkd_get_itf_search(const char *name) {
     for(int i=0;harkd_dev_itf_array[i];i++) {
          return harkd_dev_itf_array[i];
     }
     return NULL;
}
hitz_t *harkd_hitz(void) {
     return (hitz)?hitz:&s_hitz;
}



const harkd_dev_itf_t *harkd_get_itf(const char *name) {
     HARKD_ARRAY_FOREACH(const harkd_dev_itf_t,i,harkd_dev_itf_array) {
          if(!strcasecmp(i->name,name)) {
               return i;
          }
     }
     return NULL;
}







static __thread char  harkd_errstr[512] = {0};
const char *harkd_strerror(void) { return  harkd_errstr; }
harkd_r harkd_log(harkd_r type,const char *format,...) {
     va_list args;
     va_start(args,format);
     if(type > 0) {
          if(getenv("harkd_DEBUG")) {
               fprintf(stderr,"harkd: [D] ");
               vfprintf(stderr,format, args);
               fprintf(stderr,"\n");
          }
     } else if(type == 0) {
          fprintf(stderr,"harkd: [I] ");
          vfprintf(stderr,format, args);
          fprintf(stderr,"\n");
     } else if(type < 0) {
          vsnprintf (harkd_errstr,sizeof(harkd_errstr)-1,format, args);
          if(getenv("harkd_DEBUG")) {
               fprintf(stderr,"harkd: [E] %s\n",harkd_errstr);
          }
          hitz_fprintf(harkd_hitz(),2,"harkd: error: %s\n",harkd_errstr);
     }
     va_end(args);
     return type;
}
```

Source file: **harkdc/src/harkd-serial.c**

```
#include "harkd.h"
#include <libserialport.h>
#include <string.h>
/* ------------------------------------------------------------------------------------
--- */
harkd_r harkd_serial_ports(char *o_ports,int o_len) {
     struct sp_port **port_list = NULL;
     o_ports[0] = '\0';
     if(sp_list_ports (&port_list)==SP_OK) {
          for(struct sp_port **port=port_list;*port;port++) {
               const char *p = sp_get_port_name (*port);
               HARKD_LIST_FOREACH(harkd_dev_obj_t,d,harkd_dev_obj_list) {
                    if(!strcasecmp(p,d->portname)) {
                         p = NULL; break;
                    }
               }
               if(p) {
                    strncat(o_ports,p,o_len-1);
                    strncat(o_ports,"\n",o_len-1);
               }
          }
          o_ports[o_len-1] = '\0';
          sp_free_port_list (port_list);
          return HARKD_OK;
     } else {
          char *err = sp_last_error_message();
          harkd_error(NULL,"%s",err);
          sp_free_error_message(err);
          return HARKD_ERR;
     }
}
sp_port_t *harkd_serial_open(harkd_dev_obj_t *harkd,const char *i_portname) {
     sp_port_t *port  = NULL;
     harkd_close(harkd);
     if(sp_get_port_by_name(i_portname,&port)!=SP_OK) {
          goto error;
     }
     // debugf("Port %p: Opening %s ...",port,i_portname);
     if(sp_open(port,SP_MODE_READ | SP_MODE_WRITE)!=SP_OK) {
          goto error;
     }
     if(sp_set_baudrate    (port, 9600               )!=SP_OK) goto error;
     if(sp_set_bits        (port, 8                  )!=SP_OK) goto error;
     if(sp_set_parity      (port, SP_PARITY_NONE     )!=SP_OK) goto error;
     if(sp_set_stopbits    (port, 1                  )!=SP_OK) goto error;

     if(sp_set_rts         (port, SP_RTS_OFF         )!=SP_OK) goto error;
     if(sp_set_cts         (port, SP_CTS_IGNORE      )!=SP_OK) goto error;
     if(sp_set_dtr         (port, SP_DTR_OFF         )!=SP_OK) goto error;
     if(sp_set_dsr         (port, SP_DSR_IGNORE      )!=SP_OK) goto error;
     if(sp_set_xon_xoff    (port, SP_XONXOFF_DISABLED)!=SP_OK) goto error;

     if(sp_set_flowcontrol (port, SP_FLOWCONTROL_NONE)!=SP_OK) goto error;
     if(sp_flush           (port, SP_BUF_BOTH        )!=SP_OK) goto error;
     harkd->port = port;
     strncpy(harkd->portname,i_portname,sizeof(harkd->portname)-1);
```

```
     return port;
error:
     harkd_set_serial_error(harkd,"Opening error.");
     harkd_close(harkd);
     return NULL;
}
harkd_r    harkd_serial_write    (harkd_dev_obj_t *harkd,void *buffer,size_t len) {
     sp_port_t *port = harkd->port;
     if(!port) return HARKD_ERR;
     if(sp_blocking_write (port,buffer,len,1000)!=len) {
          harkd_set_serial_error(harkd,"Writting error.");
          return HARKD_ERR;
     }
     if(sp_drain(port)!=SP_OK) {
          harkd_set_serial_error(harkd,"Writting error.");
          return HARKD_ERR;
     }
     return HARKD_OK;

}
harkd_r    harkd_serial_read     (harkd_dev_obj_t *harkd,void *buffer,size_t len) {
     sp_port_t *port = harkd->port; int r;
     if(!port) return HARKD_ERR;
     r = sp_blocking_read (port,buffer,len,1000);
     if(r==len) {
          return HARKD_OK;
     } else {
          if(r<0) harkd_set_serial_error(harkd,"Reading error");
          return HARKD_ERR;
     }
}
char      *harkd_serial_gets(harkd_dev_obj_t *harkd,char *buffer,size_t len) {
     for(size_t i=0;i<(len-1);i++) {
          if(harkd_serial_read(harkd,buffer+i,1)!=HARKD_OK) {
               return NULL;
          }
          if(buffer[i]=='\r'||buffer[i]=='\n') {
               if(i--) {
                    buffer[i] = '\0';
                    return buffer;
               }
          }
     }
     harkd_error(harkd,"Buffer overflow.");
     return NULL;
}


harkd_r   harkd_serial_puts(harkd_dev_obj_t *harkd,char *buffer) {
     harkd_log(1,"serial %s",buffer);
     return harkd_serial_write(harkd,buffer,strlen(buffer));
}
void       harkd_set_serial_error(harkd_dev_obj_t *harkd,const char *whence) {
     char *err = sp_last_error_message();
     harkd_error(harkd,"%s: Serial port `%s`: %s",whence,harkd->portname,err);
     sp_free_error_message(err);
}
```

2 Codigo fuente del programa HARKDW.

Source file: **harkdw/bin/harkdw-dcdc.tcl**

```
#!/usr/bin/env wish
lappend auto_path [file normalize [file dirname [info script]]/..]
package require harkdw
package require Tk
proc harkdw-dcdc { args } {
    global env tcl_platform
    set env(harkd_DEBUG) 1
    if { $tcl_platform(platform) eq {unix}} {
        set env(PATH) [file dirname [info nameofexecutable]]:$env(PATH)
    }
    harkd-show-interface -title "DC/DC Tester."
    harkdw-define-option-number 1 0 Vin    "Vin"   4      "(V)   Input voltage."
              0   20     0.1
    harkdw-define-option-number 2 0 CH     "CH"    1      "(1|2) Channel to use."
              1   2      1
    harkdw-define-option-number 3 0 Imaxin "Iin"   5      "(A)   Maximun input current."
             0.1 10     0.1
    harkdw-define-option-number 1 1 Imax   "Iout"  0.8    "(A)   Maximun output
current."             0.1 10     0.1
    harkdw-define-option-number 2 1 N      "N"     10     "(N)   Number of points."
              1   200    1
    harkdw-define-option-number 3 1 Tstep  "Tstep" 1000   "(ms)  Time to wait for each
measurement."  1   100000 1
}
if {[file tail [info script]] eq {harkdw-dcdc.tcl}} { harkdw-dcdc }
```

Source file: **harkdw/harkdw/pkgIndex.tcl**

```
# Tcl package index file, version 1.1
# This file is generated by the "pkg_mkIndex" command
# and sourced either when an application starts up or
# by a "package unknown" script.  It invokes the
# "package ifneeded" command to set up package-related
# information so that packages will be loaded automatically
# in response to "package require" commands.  When this
# script is sourced, the variable $dir must contain the
# full path name of this file's directory.

package ifneeded harkdw 0.1 [list source [file join $dir harkd.tcl]]\n[list source [file
join $dir harkdw.tcl]]\n[list source [file join $dir images.tcl]]
```

Source file: **harkdw/harkdw/harkd.tcl**

```
#l:
package provide harkdw 0.1
package require Tk
proc harkdc-set-handle { args } {
    global HARKD_HANDLE
    foreach {var val} $args {
        set HARKD_HANDLE($var) $val
    }
}
proc harkdc-stop { } {
    global HARKD_FP HARKD_HANDLE tcl_platform
    if {[info exists HARKD_FP]} {
        catch { close $HARKD_FP }
        catch { unset HARKD_FP }
        if {$tcl_platform(platform) eq {windows}} {
            catch { exec taskkill /f /im harkdc.exe }
        }
        if {[info exists HARKD_HANDLE(-stop)]} {
            eval $HARKD_HANDLE(-stop)
        }

    }
}
proc harkdc-isrunning { } {
    global HARKD_FP
    return [info exists HARKD_FP]
}
proc harkdc { args } {
    global HARKD_FP HARKD_HANDLE env
    harkdc-stop
    puts stderr "harkdc $args"
    fconfigure stderr -buffering none
    set env(UTERM_NO_STDERR) 1
    set HARKD_FP [open |[list harkdc {*}$args] rb]
    fconfigure $HARKD_FP -blocking 0
    fileevent $HARKD_FP readable {
        global HARKD_FP
        if {[gets $HARKD_FP line] < 0} {
            if {[eof $HARKD_FP]} {
                harkdc-stop
            }
        } else {
            puts stderr "Line: $line"
            if {[info exists HARKD_HANDLE(-line)]} {
                $HARKD_HANDLE(-line) $line
            }
        }
    }
}
proc harkd-open { url } {
    global tcl_platform
    if {$tcl_platform(platform) eq {windows}} {
        puts $url
        exec cmd.exe /C start $url $url &
    } else {
        catch { exec xdg-open $url }
    }
}
```

Source file: **harkdw/harkdw.tcl**

```
#1:
package provide harkdw 0.1
package require Tk
source [file dirname [info script]]/images.tcl
set HARKDW_URL "http://www.harkaitzv.dynu.com/harkd"
set EHU_URL    "https://www.ehu.eus/es/web/ingeniaritza-bilbo/"
set AUTHOR     "H. Agirre 12-11-2018"


set HELP_URL   [file native [file dirname [info nameofexecutable]]/harkd-manual.pdf]
if {[file exists $HELP_URL] == 0} {
    set HELP_URL "http://www.harkaitzv.dynu.com/files/harkd-manual.pdf"
}
set HARKD_GUI_OPTIONS(-title)      "DC/DC Tracer"
```

```
set HARKD_GUI_OPTIONS(-width)      800
set HARKD_GUI_OPTIONS(-height)     600
set HARKD_GUI_OPTIONS(-tool-opts)  [list -compound top -borderwidth 0 -padx 1]
set HARKD_GUI_OPTIONS(-title-cmd)  { }
set HARKD_GUI_OPTIONS(-test)       {DC/DC}
set HARKD_GUI_OPTIONS(-test-doc)   {test-dcdc.gif}
set HARKD_GUI_OPTIONS(-version)    2.0


font create Options -family Arial -size 9 -weight bold
font create Label   -family Arial -size 8



proc harkd-show-interface { args } {
    global HARKD_TEST_OPTIONS HARKD_GUI_OPTIONS AUTHOR
    set frame_options { -relief raised -borderwidth 1 -width 700}
    foreach {var val} $args {
        set o($var) $val
    }


##############################################################################################
    ## Set title.
    wm title . "HARKD - $HARKD_GUI_OPTIONS(-title)"
    wm minsize . $HARKD_GUI_OPTIONS(-width) $HARKD_GUI_OPTIONS(-height)
    wm maxsize . $HARKD_GUI_OPTIONS(-width) $HARKD_GUI_OPTIONS(-height)
    wm iconphoto . -default harkd.gif



##############################################################################################
    ## Configure toolbar.
    pack [frame .top -width 1000 {*}$frame_options] \
        -side top -fill both
    ## Create About buttons.
    button .top.about_harkd -text "" -image harkd.gif -width 70 -borderwidth 0 -command
{
        global HARKDW_URL
        harkd-open $HARKDW_URL
    }
    button .top.about_ehu -text "v$HARKD_GUI_OPTIONS(-version) $AUTHOR" -image ehu.gif \
        -compound top -borderwidth 0 -font Label -command {
        global EHU_URL
        harkd-open $EHU_URL
    }
    pack .top.about_harkd .top.about_ehu -side right
    ## Create Title logo and text.
    font create TitleFont -family Arial -size 23 -weight bold
    pack [button .top.title -text $HARKD_GUI_OPTIONS(-title) \
              -font TitleFont -foreground "#0f70b7" {*}$HARKD_GUI_OPTIONS(-tool-opts)] \
        -side left
    label .top.sep1 -text "|" ; pack .top.sep1 -side left
    ## Create controls.
    harkdc-set-handle -stop {
        .top.start configure -text "Restart" -image restart.gif
    }
    button .top.start -text " Start " -image start.gif {*}$HARKD_GUI_OPTIONS(-tool-opts)
\
        -command [string map [list %t $HARKD_GUI_OPTIONS(-test)] {
            global HARKD_TEST_OPTIONS
```

```
            if {[harkdc-isrunning]} {
                harkdc-stop
            } else {
                .body.c.output delete 1.0 end
                .top.start configure -text "  Stop " -image stop.gif
                set command [list %t]
                foreach name [array names HARKD_TEST_OPTIONS] {
                    lappend command $name=$HARKD_TEST_OPTIONS($name)
                }
                harkdc {*}$command
            }
        }]
    ## Output control.
    set HARKD_TEST_OPTIONS(output) output.xlsx
    button .top.open -text "  Open " -image spreadsheet.gif {*}$HARKD_GUI_OPTIONS(-tool-
opts) -command {
        global HARKD_TEST_OPTIONS
        if {[file exists $HARKD_TEST_OPTIONS(output)] == 0} {
            error "Please rerun the test."
        }
        harkd-open $HARKD_TEST_OPTIONS(output)
    }
    button .top.saveto -text "Save to" -image saveto.gif {*}$HARKD_GUI_OPTIONS(-tool-
opts) -command {
        global HARKD_TEST_OPTIONS
        set f [tk_getSaveFile              \
                    -defaultextension .xlsx \
                    -filetypes {
                        {Excel {.xlsx}}
                    } \
                    -initialfile $HARKD_TEST_OPTIONS(output)]
        if {[file exists $f]} { file delete $f }
    }
    button .top.help -text "  Help " -image help.gif {*}$HARKD_GUI_OPTIONS(-tool-opts) -
command {
        global HELP_URL
        harkd-open $HELP_URL
    }
    button .top.exit -text "  Exit " -image exit.gif {*}$HARKD_GUI_OPTIONS(-tool-opts) -
command {
        exit 0
    }
    pack .top.start .top.saveto .top.open .top.help .top.exit -side left
    ###############################################################################
    ## Body.
    # Create body.
    proc scrollpane_cfg {w wide high} {
        set newSR [list 0 0 $wide $high]
        if {![string equal [$w.c cget -scrollregion] $newSR]} {
            $w.c configure -scrollregion $newSR
        }
    }

    pack [frame     .body  {*}$frame_options] -side top -expand yes -fill both

    pack [canvas    .body.c    \
              -relief flat     \
             ]  \
        -side left -expand yes -fill both
```

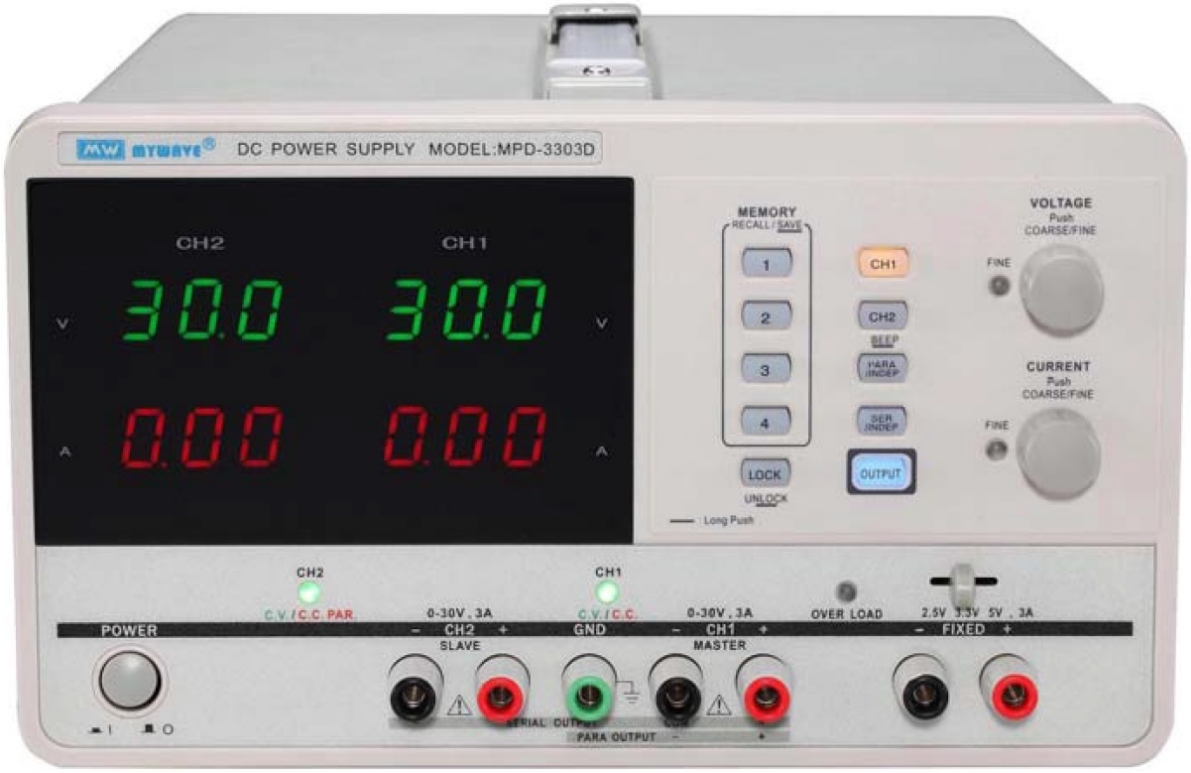
```
    # Layout body with grid.
    frame .body.c.manual {*}$frame_options
    frame .body.c.options {*}$frame_options
    font create LogFont -family Courier -size 9
    text  .body.c.output -font LogFont
    pack .body.c.manual .body.c.options .body.c.output -fill both


    #.text window create end -window .f3
    # Set image.
    catch {
        label .body.c.manual.image -image $HARKD_GUI_OPTIONS(-test-doc)
        pack  .body.c.manual.image
    }

    harkdc-set-handle -line harkdw-handle-input
}




proc harkdw-handle-input { line } {
    if {[regexp {^Error: (.*)} $line ign err]} {
        .body.c.output insert end $line\n
    } else {
        .body.c.output insert end $line\n
    }
}




##########################################################################
proc harkdw-define-option { r c name txt value descr widget args } {
    global HARKD_TEST_OPTIONS HARKD_GUI_OPTIONS
    set HARKD_TEST_OPTIONS($name) $value
    set fr .body.c.options.f$name
    # pack [frame $fr] -side top -fill both
    grid [frame $fr -width 400] -row $r -column $c -sticky W

    pack [label $fr.l -text "$txt: " -width 7 -anchor nw -font Options] -side left -padx
5
    pack [$widget $fr.e {*}$args] -side left
    pack [label $fr.d -text $descr -font Options] -side left -padx 0
}
proc harkdw-define-option-text { r c name txt value descr } {
    harkdw-define-option $r $c $name $txt $value $descr \
        entry \
        -textvariable HARKD_TEST_OPTIONS($name)
}
proc harkdw-define-option-number { r c name txt value descr min max step } {
    harkdw-define-option $r $c $name $txt $value $descr    \
        spinbox -from $min -to $max -increment $step -width 5 \
        -textvariable HARKD_TEST_OPTIONS($name)
}
```

Apendice C - Manuales de instrumentos.

Apéndice C

Manuales de los instrumentos

C.1 Fuente de alimentación MPD-3305D, Manual de usuario

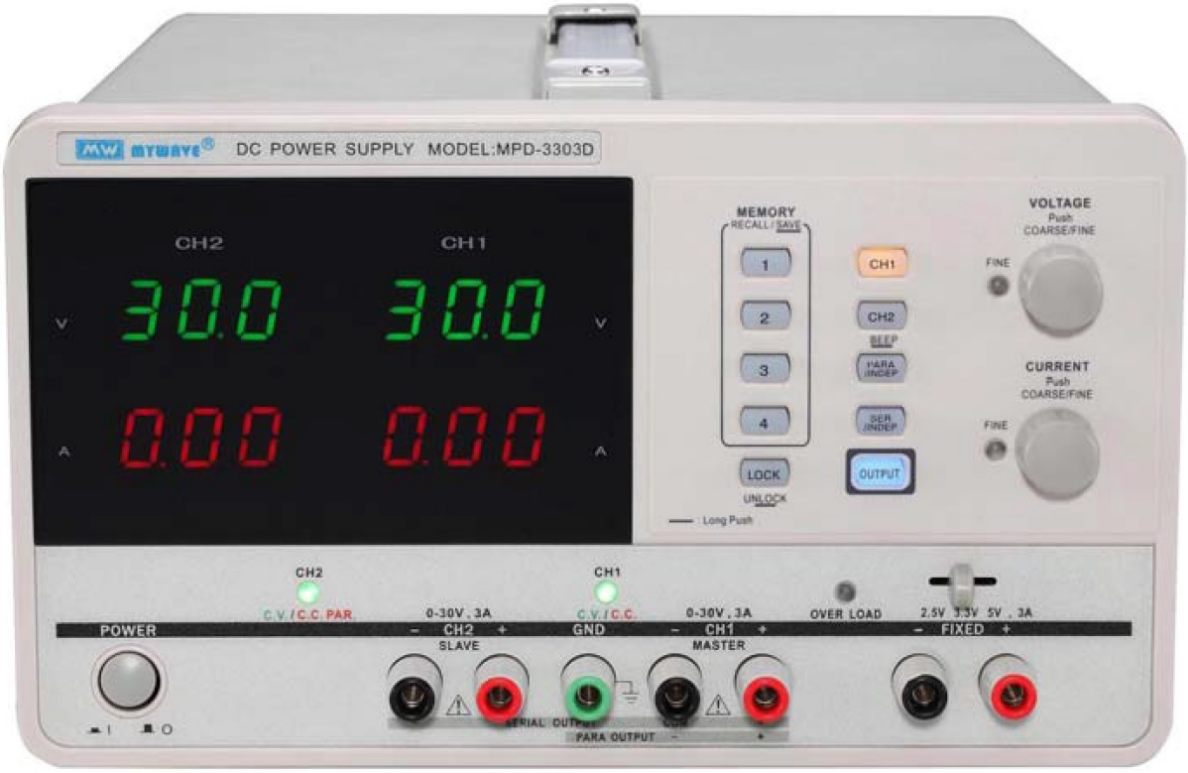


BENCHTOP INSTRUMENT

MYWAVE®

MPD-3305D

Multi-channel Programmable DC Power Supply Operation Manual



Part No. PPOM-010110E

CE

Contents

SAFETY INSTRUCTION II
1. OVERVIEW ..... - 1 -
1.1 Operation and Storage Environment ..... - 1 -
1.2 Introduction ..... - 1 -
1.3 Series Lineup/Main Features ..... - 2 -
1.4 Principle of Operation ..... - 3 -
1.5 Front Panel Overview ..... - 4 -
1.6 Rear Panel Overview ..... - 7 -
1.7 CV/CC Crossover Characteristics ..... - 8 -
2. SETUP ..... - 9 -
2.1 Power Up ..... - 9 -
2.2 Load Cable Connection ..... - 9 -
2.3 Output ON/OFF ..... - 10 -
2.4 Beep ON/OFF ..... - 10 -
2.5 Front Panel Lock ..... - 10 -
3. OPERATION ..... - 11 -
3.1 CH1/CH2 Independent Mode ..... - 11 -
3.2 CH3 Independent Mode ..... - 11 -
3.3 CH1/CH2 Tracking Series Mode ..... - 12 -
3.4 CH1/CH2 Tracking Parallel Mode ..... - 15 -
4. SAVE/RECALL SETUP ..... - 17 -
4.1 Save Setup ..... - 17 -
4.2 Recall Setup ..... - 17 -
5. REMOTE CONTROL (Only for MPD-3303S/3303D/3305D) ..... - 18 -
5.1 Remote Control Setup ..... - 18 -
5.2 Remote Connection Step ..... - 18 -
5.3 Command Syntax ..... - 19 -
5.4 Error Messages ..... - 19 -
5.5 Command List ..... - 19 -
5.6 Command Details ..... - 20 -
6. MAINTENANCE ..... - 23 -
6.1 Inspection ..... - 23 -
6.2 Fuse Replacement ..... - 23 -
6.3 Cleaning ..... - 23 -
7. FAQ ..... - 24 -
SPECIFICATIONS (MPD-3303S) ..... - 24 -
SPECIFICATIONS (MPD-3303/3305) ..... - 25 -
SPECIFICATIONS (MPD-3303D/3305D) ..... - 27 -
SPECIFICATIONS (MPD-3303C/3305C) ..... - 28 -

Use of Operation Manual

Please read through and understand this Operation Manual before operating the product. After reading, always keep the manual nearby so that you may refer to it as needed. When moving the product to another location, be sure to bring the manual as well.

Calibration notification

We notify that the instruments included in this manual are in compliance with the features and specifications as stated in this manual. Before shipment, the instrument has been calibrated in factory. The calibration procedures and standards are compliant to the national regulations and standards for electronic calibration.

Warranty

We guarantee that the instrument has been passed strict quality check. We warrant our instrument's mainframe and accessories in materials within the warranty period of one year. We guarantee the free spare parts for products which are approved defective in this period. To get repair service, please contact with your nearest sales and service office. We do not provide any other warranty items except the one being provided by this summary and the warranty statement. The warranty items include but not being subjected to the hinted guarantee items related to tradable characteristics and any particular purpose. We will not take any responsibility in cases regarding to indirect, particular and ensuing damage, such as modifications to the circuit and functions by the users, repairing or component replacement by the users, or damage during transportation.

For product improvement, the specifications are subject to change without prior notice.

SAFETY INSTRUCTION

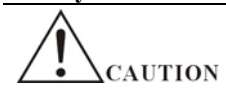
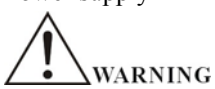
This chapter contains important safety instructions that you must follow when operating the instrument and when keeping it in storage. Read the following before any operation to insure your safety and to keep the best condition for the instrument.

Safety Symbols

The following safety symbols may appear in this manual or on the instrument:

	**WARNING**	Identifies conditions or practices that could result in injury or loss of life.
	**CAUTION**	Identifies conditions or practices that could result in damage to the instrument or to other properties.
	**DANGER**	High voltage
	**ATTENTION**	Refer to the manual
		Protective conductor terminal
		Earth (ground) terminal

Safety Guidelines

	● Before plugging into local AC mains, check and make sure that the output voltage is compatible to the load. (It is suggested to disconnect a load before plugging into local AC mains. ● Do not use this instrument near water. ● Do not operate or touch this instrument with wet hands. ● Do not open the casing of the instrument when it is connected to AC mains. ● The max.output voltage of the instrument may be over 60VDC, avoid touch the metal contact part of the output terminals. ● Do not use the instrument in an atmosphere which contains sulfuric acid mist or other substances which cause corrosion to metal. ● Do not use the instrument in a dusty place or a highly humid place as such will cause instrument reliability degradation and instrument failures. ● Install the instrument in a place where is free from vibration. ● Install the instrument in a place where the ambient temperature is in range of -10~70℃. Note that the instrument operation may become unstable if it is operated in an ambient temperature exceeding the range of 0~40℃
Power supply 	AC Input voltage: 110V/220V ±10%, 50/60Hz Connect the protective grounding conductor of the AC power cord to an earth ground to avoid electrical shock.
Fuse 	● Fuse type: 110V: T6.3A /250V, 220V: T3.15A/250V. ● Make sure the correct type of fuse is installed before power up. ● Replace the AC fuse with the same type and rating as the original fuse. ● Disconnect the power cord before fuse replacement. ● Make sure the cause of fuse blowout is fixed before fuse replacement.

Power cord for the United Kingdom

When using the power supply series in the United Kingdom, make sure the power cord meets the following safety instructions.

NOTE: This lead/appliance must only be wired by competent persons.

WARNING: THIS APPLIANCE MUST BE EATHED.

IMPORTANT: The wires in this lead are coloured in accordance with the following code:

Green/Yellow: Earth

Blue: Neutral

Brown: Live

As the colours of the wires in main leads may not correspond with the colours marking identified in your plug/appliance, proceed as follows:

- The wire which is coloured Green & Yellow must be connected to the Earth terminal marked with the letter E or by the earth symbol ⏚ or coloured Green or Green & Yellow.
- The wire which is coloured Blue must be connected to the terminal marked with the letter N or coloured Blue or Black.
- The wire which is coloured Brown must be connected to the terminal marked with the letter L or P or coloured Brown or Red.

If in doubt, consult the instructions provided with the equipment or contact the supplier.

This cable/appliance should be protected by a suitably rated and approved HBC mains fuse: refer to the rating information on the equipment and/or user instructions for details. As a guide, cable of 0.75mm^2 should be protected by a 3A or 5A fuse. Larger conductors would normally require 13A types, depending on the connection method used.

Any moulded mains connector that requires removal/replacement must be destroyed by removal of any fuse & fuse carrier and disposed of immediately, as a plug with bared wires is hazardous if a engaged in live socket. Any re-wiring must be carrier out in accordance with information detailed on this label.

1. OVERVIEW

This chapter describes the instrument in a nutshell, including its main features and front /rear panel introduction. After going through the overview, follow the SETUP chapter to properly power up and set operation environment.

1.1 Operation and Storage Environment

Operation Environment

Location: Indoor, no direct sunlight, dust free, almost non-conductive pollution (note below)

Relative Humidity: < 80%

Altitude: < 2000m

Temperature: 0℃ to 40℃

(Pollution Degree) EN 61010-1:2001 specifies the pollution degrees and their requirements as follows. The instrument falls under degree 2.

Pollution refers to "addition of foreign matter, solid, liquid, or gaseous (ionized gases), that may produce a reduction of dielectric strength or surface resistivity".

Pollution degree 1: No pollution or only dry, non-conductive pollution occurs. The pollution has no influence.

Pollution degree 2: Normally only non-conductive pollution occurs. Occasionally, however, a temporary conductivity caused by condensation must be expected.

Pollution degree 3: Conductive pollution occurs, or dry, nonconductive pollution occurs which becomes conductive due to condensation which is expected. In such conditions, equipment is normally protected against exposure to direct sunlight, precipitation, and full wind pressure, but neither temperature nor humidity is controlled.

Storage Environment

Location: Indoor

Relative Humidity: < 70%

Temperature: -10℃ to 70℃

1.2 Introduction

This series of regulated programmable DC power supply are light weight, adjustable, multifunctional work stations. They have three independent outputs: two with adjustable voltage level and one with fixed level selectable from 2.5V, 3.3V and 5V. The power supply can be used for logic circuits where various output voltage or current are needed, and for tracking mode definition systems where positive and negative voltages with good accuracy are required.

Independent /Tracking Series /Tracking Parallel

The three output modes of the power supply - independent, tracking series, and tracking parallel - can be selected by pressing the TRACKING key on the front panel. In the independent mode, the output voltage and current of each channel are controlled separately. The isolation degree, from output terminal to chassis or from output terminal to output terminal, is 300V. In the tracking modes, both the CH1 and CH2 outputs are automatically connected in series or parallel; no need to connect output leads. In the series mode, the output voltage is doubled; in the parallel mode, the output current is doubled.

Constant Voltage/Constant Current

Except for CH3, each output channel is completely transistorized and well-regulated, and works in constant voltage (CV) or constant current (CC) mode. Even at the maximum output current, a fully rated, continuously adjustable output voltage is provided. For a big load, the power supply can be used as a CV source; while for a small load, a CC source.

When in the CV mode (independent or tracking mode), output current (overload or short circuit) can be controlled via the front panel. When in the CC mode (independent mode only), the maximum (ceiling) output voltage can be controlled via the front panel. The power supply will automatically cross over from CV to CC operation when the output current reaches the target value. The power supply will automatically cross over from CC to CV when the output voltage reaches the target value. For more details about CV/CC mode operation, see page8.

Automatic Tracking Mode

The front panel display (CH1, CH2) shows the output voltage or current. When operating in the tracking mode, the power supply will automatically connect to the auto- tracking mode.

Dynamic Load

When used in audio production lines, the power supply will provide a continuous or dynamic load connector. When the connectors are connected to the position "ON", a stable DC current power will be provided for audio power amplifiers.

1.3 Series Lineup/Main Features

Series Lineup

Model	Output	Voltmeter	Ammeter	USB Interface
MPD-3303S	0~30V×2, 0~3A×2 Fixed 2.5V/3.3V/5V, 3A	5 digits LED	4 digits LED	√
MPD-3303	0~30V×2, 0~3A×2 Fixed 2.5V/3.3V/5V, 3A	3 digits LED	3 digits LED	_____
MPD-3305	0~30V×2, 0~5A×2 Fixed 2.5V/3.3V/5V, 3A	3 digits LED	3 digits LED	_____
MPD-3303D	0~30V×2, 0~3A×2 Fixed 2.5V/3.3V/5V, 3A	3 digits LED	3 digits LED	√
MPD-3305D	0~30V×2, 0~5A×2 Fixed 2.5V/3.3V/5V, 3A	3 digits LED	3 digits LED	√
MPD-3303C	0~30V×2, 0~3A×2 Fixed 2.5V/3.3V/5V, 3A	3 digits LED	3 digits LED	_____
MPD-3305C	0~30V×2, 0~5A×2 Fixed 2.5V/3.3V/5V, 3A	3 digits LED	3 digits LED	_____

Main Features

Performance	➢ Low ripple & noise, intelligent cooling fan ➢ Compact design, light weight
Operation	➢ Constant voltage/constant current operation ➢ Tracking series/tracking parallel operation ➢ Output ON/OFF control

	➢ 3 outputs: 30V/3A(5A)×2, 2.5V/3.3V/5V/3A×1 ➢ Digital panel control ➢ Panel lock function ➢ 4 programming presets for voltage and current save/recall ➢ Coarse and fine control for voltage and current ➢ Software calibration ➢ Beeper output
Protection	➢ Over load protection ➢ Reverse polarity protection
Interface	➢ USB interface for remote PC control (only for models with USB interface)

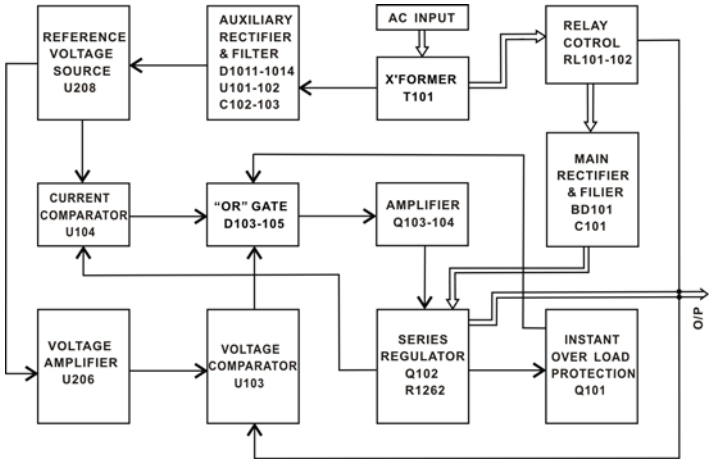
1.4 Principle of Operation

The power supply consists of the following:

- AC input circuit
- Transformer
- Bias power supply including rectifier, filter, pre-regulator and reference voltage source
- Main regulator circuit including the main rectifier and filter, series regulator, current comparator, voltage comparator, reference voltage amplifier, remote device and relay control circuit

The block diagram below shows the circuit arrangement. The single phase input power is connected to the transformer through the input circuit. Details of each part are described as below.

Overview



Rectifier	The auxiliary rectifiers D1011~D1014 provide bias voltage filtered by the capacitors C102 and C103, for the pre-regulators U101 and U102. They provide a regulated voltage for other modules.
Main rectifier	The main rectifier is a full wave bridge rectifier. It provides the power after the rectifier is filtered by the capacitor C101, and then regulated via a series-wound regulator, which is finally delivered to the output terminal.
Current limiter	U104 acts as a current limiter. When the current is over predetermined rating, U104 is activated

	and decreases the current. U208 provides a reference voltage. U206 is the inverter amplifier. U103 is a comparator amplifier which compares reference voltage and feedback voltage, and then delivers to Q103, Q104, which then calibrates the output voltage.
Overload	When the unit is overload, Q107 activates to control the current magnitude of Q104, to limit the output current. The relay control circuit controls the power dissipation in the series-wound regulated circuit.

1.5 Front Panel Overview

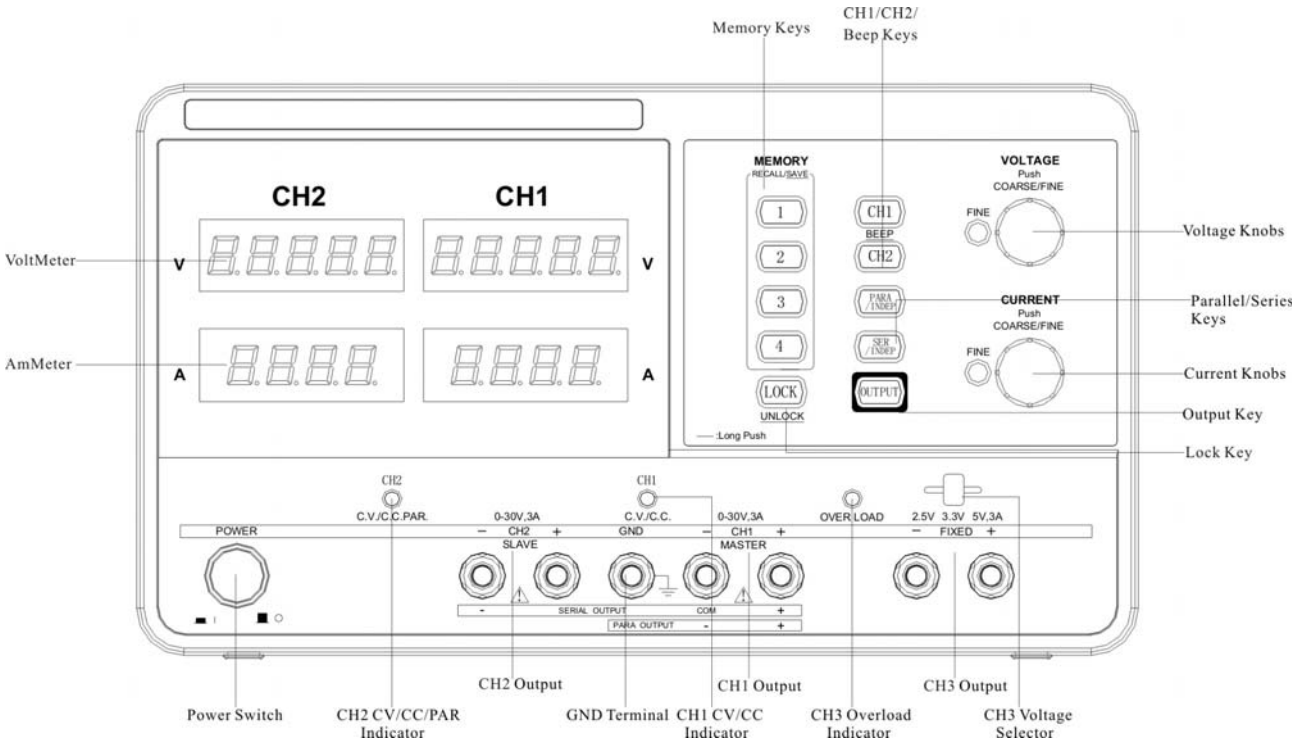


Fig.1.5-1 Front panel of MPD-3303S

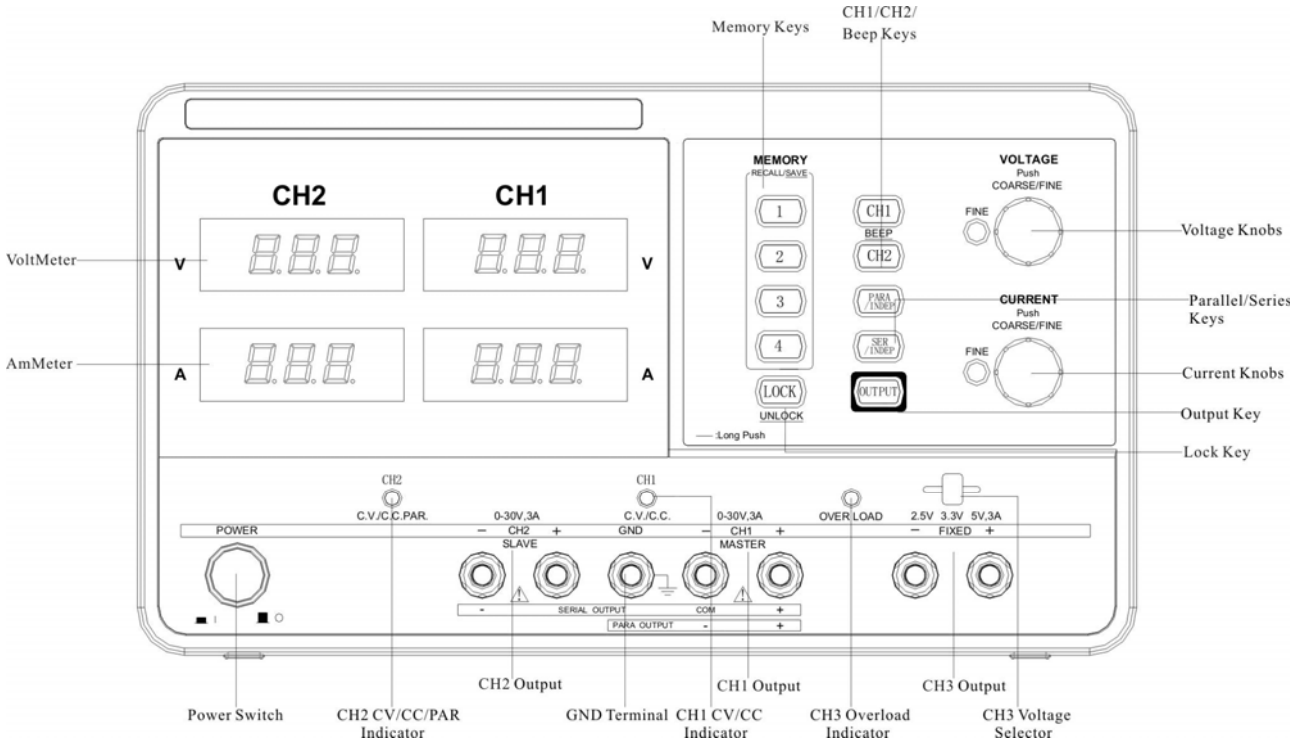
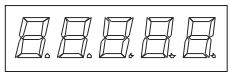
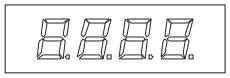
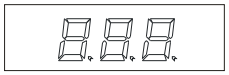
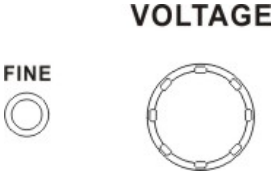
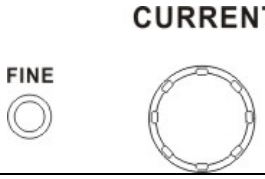
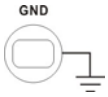


Fig.1.5-2 Front panel of MPD-3303/3305/3303C/3305C/3303D/3305D

Display		
Voltmeter	Displays CH1 or CH2 output voltage	
`	 V	 V
	For 1mV, 1mA models (5 digits)	For 10mV, 10mA /100mV, 10mA models (3 digits)
Ammeter	Display CH1 or CH2 output current	
	 A	 A
	For 1mV, 1mA models (4 digits)	For 10mV, 10mA /100mV, 10mA models (3 digits)

Control Panel		
Memory keys		Saves or recalls panel settings. Max.4 sets for programming preset. Refer to page 17 for details.
CH1/CH2 beep keys		Selects the output channel for level adjustment. Refer to page 11 for level setting details Pressing and holding CH2 key enables beep sound. Refer to page 10 for details.
Parallel/Series keys		Activates Tracking Parallel operation or Tracking Series operation. Refer to page 12 for details.
Lock key		Locks or unlocks the front panel settings. Refer to page 10 for details.
Output key		Turns the output on or off.
Voltage knobs		Adjusts the output voltage level for CH1 or CH2. Pressing the knob switches for coarse and fine level setting. When in fine adjustment, the FINE indicator lights on.
Current knobs		Adjusts the output current level for CH1 or CH2. Pressing the knob switches coarse and fine level setting. When in fine adjustment, the FINE indicator lights on.

Terminals		
Power switch		Turns on ▬ or off ▪ the main power. Refer to page 9 for power up sequence.
GND terminal		Accepts a grounding wire.
CH1 output		Outputs CH1 voltage and current.
CH1 CV/CC indicator		Indicates CH1 constant voltage or constant current operation mode.
CH2 output		Outputs CH2 voltage and current.
CH2 CV/CC/ PAR indicator		Indicates CH2 constant voltage, constant current or tracking parallel operation mode.
CH3 output		Outputs CH3 voltage and current.
CH3 overload indicator		Indicates when CH3 output current is overloaded.
CH3 voltage selector		Selects CH3 output voltage from 2.5V, 3.3V, 5V.

1.6 Rear Panel Overview

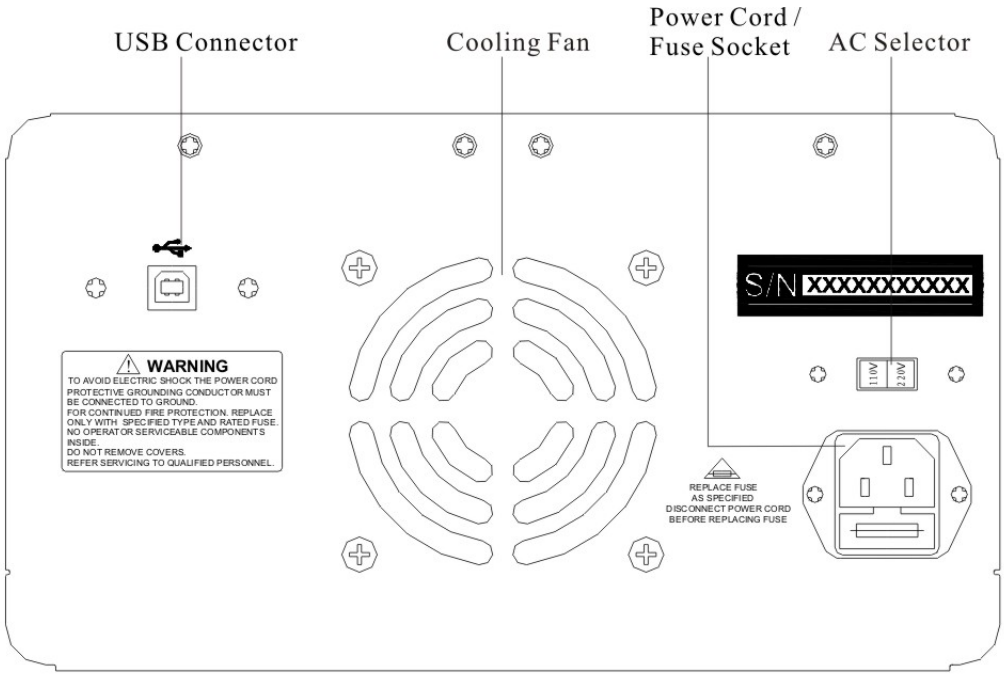


Fig.1.6-1 Rear panel of MPD-3303S/3303D/3305D

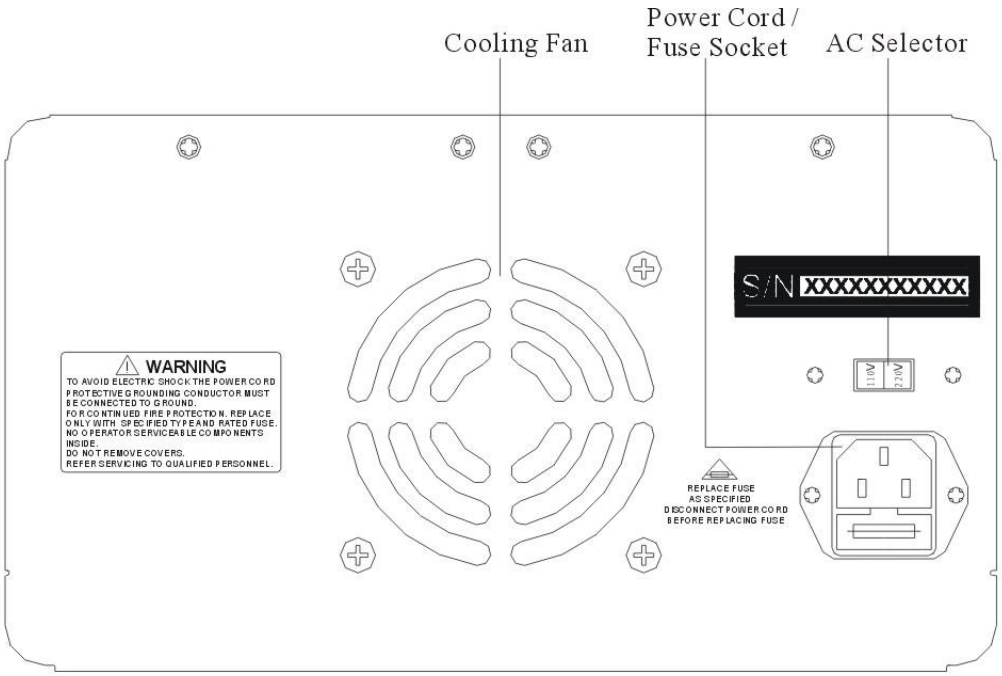
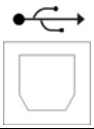
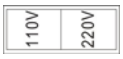
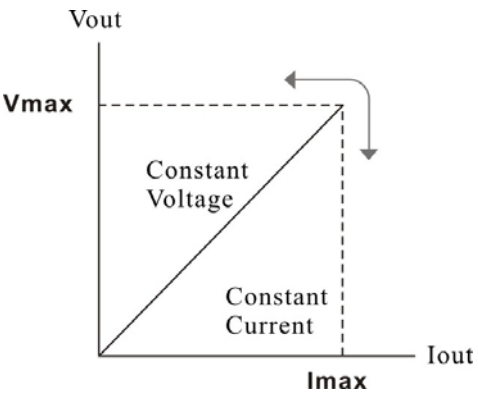


Fig.1.6-2 Rear panel of MPD3303/3305/3303C/3305C

USB connector		Accepts a USB slave connector for command-based remote control (page 18). For models with USB interface.
Power cord/fuse socket		The power cord socket accepts the AC mains. Refer to page 9 for power up details. The fuse holder contains the AC main fuse. Refer to page 23 for details of fuse replacement.
AC line voltage selector		Selects AC line voltage from 110V/220V.

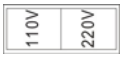
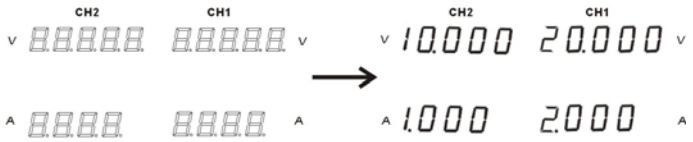
1.7 CV/CC Crossover Characteristics

Background	The instrument automatically switches between constant voltage mode (CV) and constant current mode (CC), according to load condition.
CV mode	When the current level is smaller than the output setting, the instrument operates in Constant Voltage mode. The indicator on the front panel turns green (C.V.) The Voltage level is kept at the setting and the Current level fluctuates according to the load condition until it reaches the output current setting.
CC mode	When the current level reaches the output setting, the instrument starts operating in Constant Current mode. The indicator on the front panel turns red (C.C.) The Current level is kept at the setting but the Voltage level becomes lower than the setting, in order to suppress the output power level from overload. When the current level becomes lower than the setting, the instrument goes back to the Constant Voltage mode.
Diagram	

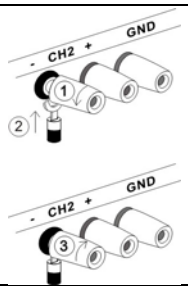
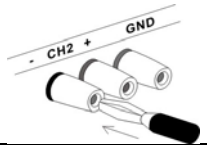
2. SETUP

This chapter describes how to properly power up and configure the power supply series before operation.

2.1 Power Up

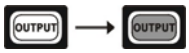
Select AC line voltage	Before powering up the power supply, select the AC input voltage from the rear panel.	
Connect AC power cord	Connect the AC power cord to the rear panel socket.	
Power on	Press the power switch to turn on the power. The display shows the initialization screen, followed by the last recalled settings.	 
Power off	Press the power switch again to turn off the power.	

2.2 Load Cable Connection

Standard accessory	1. Turn the terminal counterclockwise and loosen the screw. 2. Insert the cable terminal. 3. Turn the terminal clockwise and tighten the screw.	
Banana plug	Insert the plug into the socket.	
Wire type	When using load cables other than the attached, make sure they have enough current capacity for minimizing cable loss and load line impedance. Voltage drop across a wire should not excess 0.5V. The following list is the wire current rating at 450A/cm^2.	

Wire size (AWG)	Max. current (A)
20	2.5
18	4
16	6
14	10
12	16

2.3 Output ON/OFF

Panel operation	Pressing the Output key turns on all CH 1/2/3 outputs.  The key LED also turns on. Pressing the Output key again turns off the output and the key LED.
Automatic output off	Any of the following actions during output on automatically turns it off. They might involve sudden and harmful change in the output level. ➢ Change the operation mode between independent / tracking series / tracking parallel ➢ Recalling other setups from the memory ➢ Storing the setup into the memory

2.4 Beep ON/OFF

Panel operation	By default, the beeper sound is enabled. To turn off the beep, press the beep key  for 2 seconds. A beep sound comes out and the beeper setting will be turned off. To enable the beeper, press the beep key again for 2 seconds.
List of beeper	The following operations go with a beep sound when the beeper setting is on. Power on INDEP – SER – PAR mode switching Setup save/recall Voltage/current knob, fine/coarse knob Output on/off Panel lock/unlock CH1/CH2 output level knob Voltage/current level reaching minimum (zero) level

2.5 Front Panel Lock

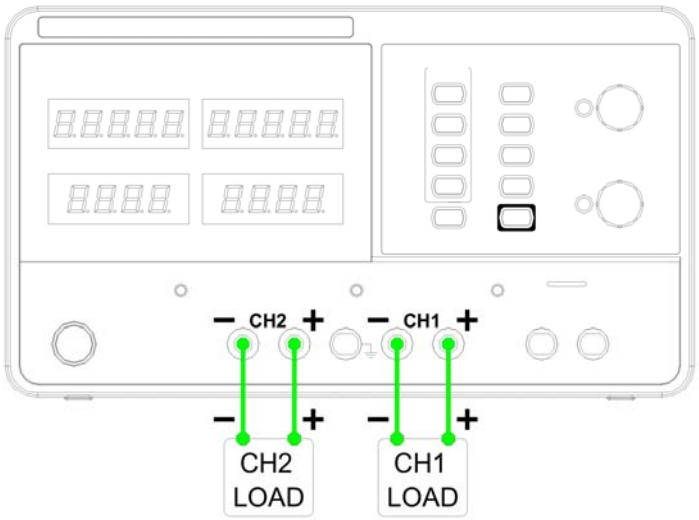
Panel operation	Press the LOCK key to lock the front panel key operation. The key LED turns on. To unlock, press the LOCK key for 2 seconds. The key LED also turns off. 
Note	The OUTPUT key is not affected by the lock operation. 

3. OPERATION

3.1 CH1/CH2 Independent Mode

Background/ Connection

CH1 and CH2 outputs work independent of each other and are separately controlled.



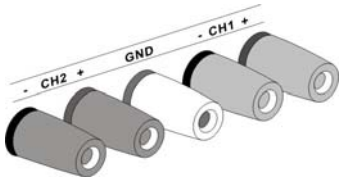
Output rating

0~30V/0~3A for each channel (I≤3A)
0~30V/0~5A for each channel (I>3A)

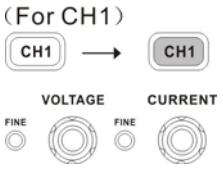
Panel operation

1. Make sure the PARA INDEP and SERIES INDEP keys are turned off (the key LEDs are off)

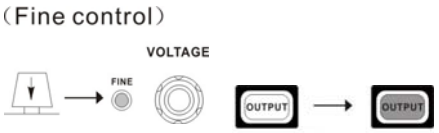
2. Connect the load to the front panel terminals, CH1 +/-, CH2 +/-.



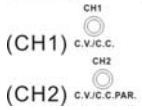
3. Set the CH1 output voltage and current. Press the CH1 switch (LED turns on) and use the Voltage and Current knob. By default, the Voltage and Current knob work in the coarse mode. To activate the fine mode, press the knob and turn on the FINE LED.



Coarse: 0.1V or 0.1A @ rotation click.
Fine: the smallest digit @ rotation click.



4. Repeat the above settings for CH2 channel.

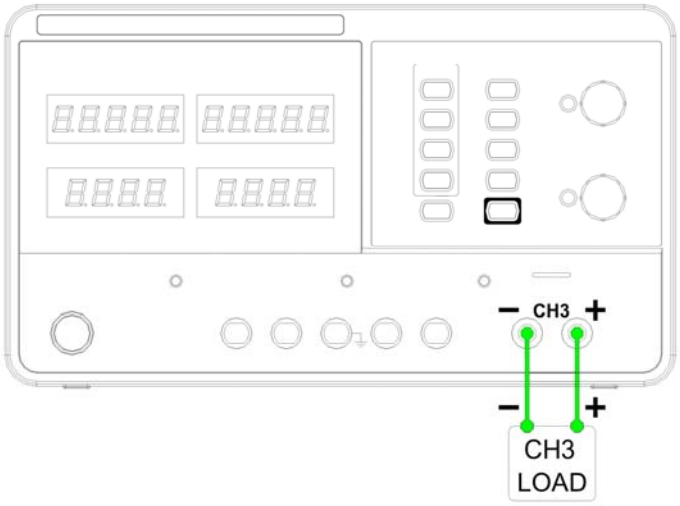


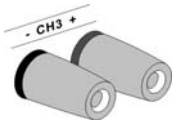
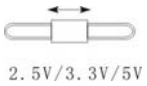
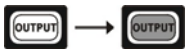
5. To turn on the output, press the output key. The key LED turns on and the CH1 /CH2 indicator shows the output mode, CV or CC.

3.2 CH3 Independent Mode

Background/ Connection

The CH3 rating is 2.5V/3.3V/5V, maximum 3A. It works independently from CH1 and CH2, regardless of their modes.



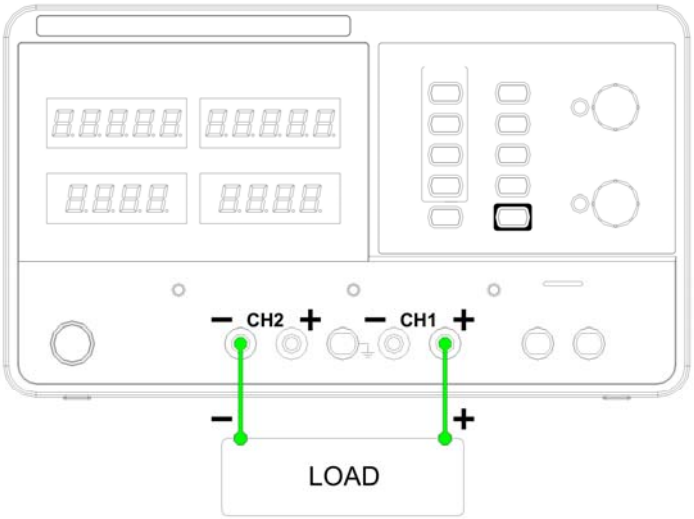
Output rating	Fixed 2.5V/3.3V/5V, 3A
No tracking Series/Parallel mode	CH3 does not have tracking series/parallel mode. Also, CH3 output is not affected by CH1 and 2 modes.
Panel operation	1. Connect the load to the front panel CH3 +/- terminal.  2. Select the output voltage from 2.5V, 3.3V and 5V, using the CH3 voltage selector switch.  3. To turn on the output, press the output key. The key LED turns on. 
CV to CC	When the output Current level exceeds 3A, the overload indicator turns red and CH3 operation mode switches from Constant Voltage to Constant Current.  Note: “overload” in this case does not mean an abnormal operation.

3.3 CH1/CH2 Tracking Series Mode

Background	Tracking series operation doubles the Voltage capacity of the power supply series by internally connecting CH1 (Master) and CH2 (Slave) in series and combining the output to a single channel. CH1 (Master) controls the combined Voltage output level. The following describes two types of configurations depending on the common ground usage.

Tracking series without common terminal

Connection

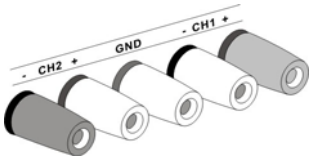


Output rating 0~60V/0~3A (I≤3A)
0~60V/0~5A (I>3A)

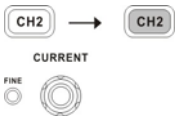
Panel operation

1. Press the SER/INDEP key to activate the tracking series mode. The key LED turns on.

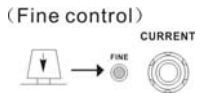
2. Connect the load to the front panel terminals, CH1+ & CH2-. (Single supply).



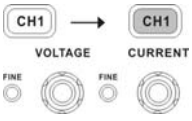
3. Press the CH2 switch (LED turns on) and use the Current knob to set the CH2 output current to the maximum level. By default, the Voltage and Current knob work in the coarse mode. To activate the fine mode, press the knob and turn on the FINE LED.



Coarse: 0.1V or 0.1A @ rotation click.
Fine: the smallest digit @ rotation click.

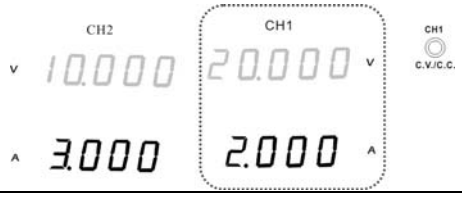


4. Press the CH1 switch (LED turns on) and use the Voltage and Current knob to set the output voltage and current level.



5. To turn on the output, press the output key. The key LED turns on.

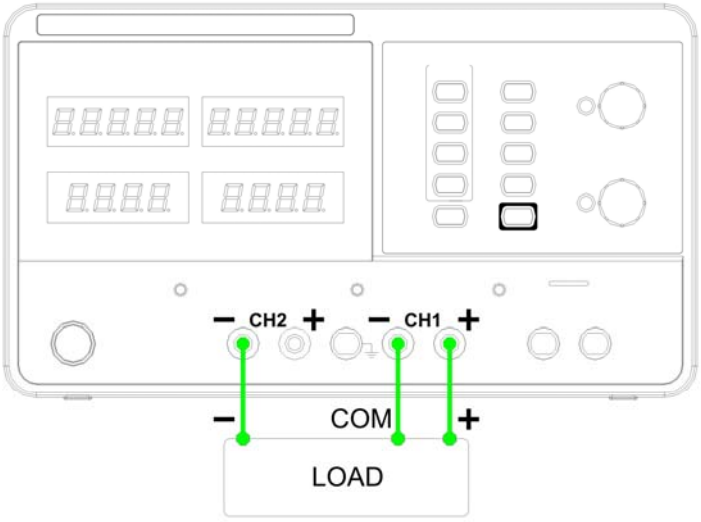
6. Refer to the CH1 (Master) meter and indicator for the output setting level and CV/CC status.



Voltage level	Double the reading on the CH1 Voltage meter. In the above case, the actual output is 20.0 x 2= 40.0V.
Current level	CH1 meter reading shows the output Current. In the above case, 2.000A. (CH2 Current control must be in the Maximum position=3.0A(I≤3A) or 5.0A(I>3A)).

Tracking series with common terminal

Connection

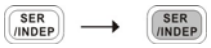


Output rating

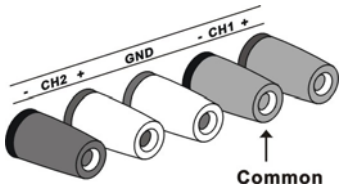
0~30V/0~3A for CH1~COM (I≤3A) 0~30V/0~5A for CH1~COM (I>3A)
0~-30V/0~3A for CH2~COM (I≤3A) 0~-30V/0~5A for CH2~COM (I>3A)

Panel operation

1. Press the SER/INDEP key to activate the tracking series mode. The key LED turns on.



2. Connect the load to the front panel terminals, CH1+ & CH2-. Use the CH1 (-) terminal as the common line connection.

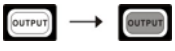


3. Press the CH1 switch (LED turns on) and use the Voltage knob to set the master & slave output voltage (the same level for both channels). By default, the Voltage and Current knob work in the coarse mode. To activate the fine mode, press the knob and turn on the FINE LED.

Coarse: 0.1V or 0.1A @ rotation click.
Fine: the smallest digit @ rotation click.

4. Use the current knob to set the master output current.

5. To turn on the output, press the output key. The key LED turns on.

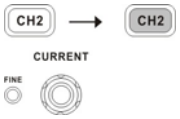


6. For the master (CH1) output level and CV/CC status, refer to the CH1 meter and indicator.

Master (CH1) voltage level: CH1 meter reading shows the output voltage. In the above case, 20.0V.

Master (CH1) current level: CH1 meter reading shows the output current. In the above case, 2.000A.

7. Press the CH2 switch (LED turns on) and use the Current knob to set the slave output current.



8. For the slave (CH2) output level and CV/CC status, refer to the CH1/2 meter and CH2 indicator.

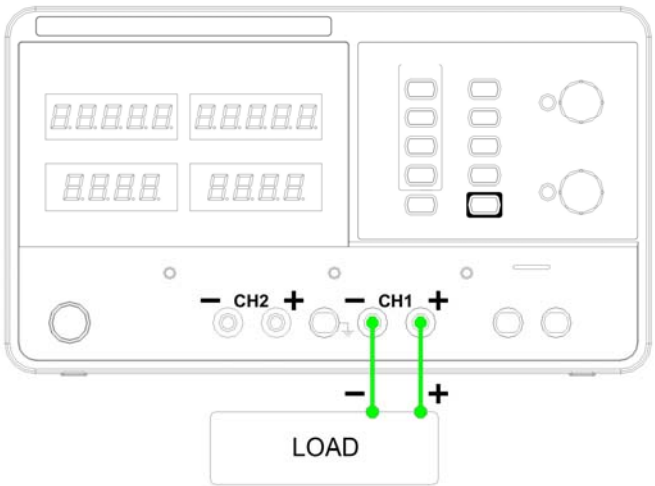
Slave (CH2) voltage level: CH1 meter reading shows the output voltage. In the above case, 20.0V.

Slave (CH2) current level: CH2 meter reading shows the output current. In the above case, 3.000A.

3.4 CH1/CH2 Tracking Parallel Mode

Background/ connection

Tracking parallel operation doubles the current capacity of the power supply series by internally connecting CH1 and CH2 in parallel and combining the output to a single channel. CH1 controls the combined output.

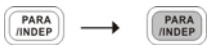


Output rating

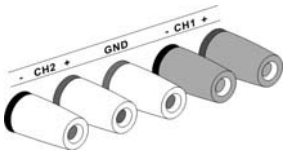
0~30V/0~6A (I≤3A)
0~30V/0~10A (I>3A)

Panel operation

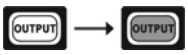
1. Press the PAR/INDEP key to activate the tracking parallel mode. The key LED turns on.



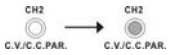
2. Connect the load to the CH1 +/- terminals.



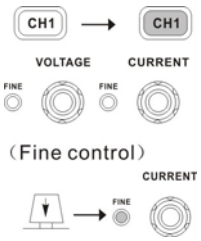
3. To turn on the output, press the output key. The key LED turns on.



4. The CH2 indicator turns red, indicating tracking parallel (PAR) mode.



5. Press the CH1 switch (LED turns on) and use the Voltage and Current knob to set the output voltage and current. The CH2 output control is disabled. By default, the Voltage and Current knob work in the coarse mode. To activate the fine mode, press the knob and turn on the FINE LED.

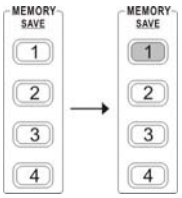


6. For the output level and CV/CC status, refer to the CH1 meter and indicator.

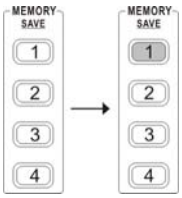
Voltage level: The CH1 meter reading shows the output voltage. In the above case, 20.0V.
Current level: Double the amount of CH1 current meter reading. In the above case,
2.0A x 2 = 4.0A.

4. SAVE/RECALL SETUP

4.1 Save Setup

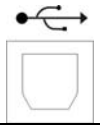
Background	The front panel settings can be stored into one of the four internal memories.
Programming contents	The following list shows the programming setting contents: ➢ Independent / tracking series / tracking parallel mode ➢ CH1/CH2 knob selection ➢ Fine/coarse knob editing mode ➢ Beep on/off ➢ Output voltage/current level The following settings are always saved as "off": ➢ Output on/off ➢ Front panel lock on/off
Panel operation	Press one of the 1~4 Memory keys for 2 seconds, for example memory 1. The panel settings will be saved in memory 1 by long push to this key and the key LED turns on. When the panel settings are modified, the LED turns off. 
Note	When the setting is stored, the output automatically turns off.

4.2 Recall Setup

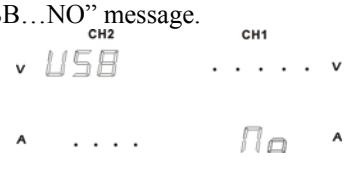
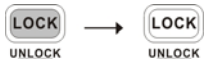
Background	The front panel settings can be recalled from one of the four internal memories.
Programming contents	The following list shows the programming setting contents: ➢ Independent / tracking series / tracking parallel mode ➢ CH1/CH2 knob selection ➢ Fine/coarse knob editing mode ➢ Beep on/off ➢ Output voltage/current level The following settings are always recalled as "off": ➢ Output on/off ➢ Front panel lock on/off
Panel operation	Press one of the 1~4 Memory keys, for example memory 1. The panel settings saved in memory 1 will be recalled by pressing this key. The key LED turns on. When the panel settings are modified, the LED turns off. 
Note	When a setting is recalled, the output automatically turns off.

5. REMOTE CONTROL (Only for MPD-3303S/3303D/3305D)

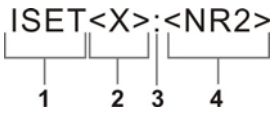
5.1 Remote Control Setup

Background	The power supply is capable of being remotely controlled via the USB connection.
Interface	 USB slave port, rear panel
COM setting	Set up the COM port inside the PC according to the following list: Baud rate: 9600 Parity bit: None Data bit: 8 Stop bit: 1 Data flow control: None
Functionality check	Run this query command via the terminal application such as MTTTY (Multi-threaded TTY). *IDN? This should return the identification information: series number, firmware version.

5.2 Remote Connection Step

Enter the remote control mode	1. Connect the USB cable to the slave port. 2. The connection will be automatically established, and the front panel shows "USB…YES" message.  3. The power supply also automatically enters the lock state (the Lock key will be activated). 
Exit the remote control mode	1. Disconnect the USB cable from the rear. 2. The display shows "USB…NO" message.  3. Unlock the power supply by pressing the Lock key until it turns off backlight.  4. The power supply goes back to local operation mode.

5.3 Command Syntax

Command format		1: command header 2: output channel 3: separator 4: parameter

Parameter	Type	Description	Example
	<Boolean>	Boolean logic	0 (off), 1 (on)
	<NR1>	Integers	0, 1, 2, 3
	<NR2>	Decimal numbers	0.1, 3.14, 8.5

Output channel	1 (CH1) or 2 (CH2)
Note	Commands must be capital letters

5.4 Error Messages

The following error messages might appear when the power supply cannot accept the command.

Message contents		Descriptions
a	Program mnemonic too long	The command length must be 15 characters or less.
b	Invalid character	Invalid characters, such as symbols, are entered. Example: VOUT#
c	Missing parameter	The parameter is missing from the command. Example: VSET: (should have a number)
d	Data out of range	The entered value exceeds the specification. Example: VSET:33 (should be≤32V)
e	Command not allowed	The entered command is not allowed in the circumstance. Example: trying to set CH2 output while in the tracking mode.
f	Undefined header	The entered command does not exist, or the syntax is wrong.

5.5 Command List

Detailed descriptions of each command starts from the next page.

The “HELP” command shows all the following commands and their meanings, except for the HELP command itself.

Command	Meanings
ISET<X>:<NR2>	Sets the output current
ISET<X>?	Retunes the output current setting
VSET<X>:<NR2>	Sets the output voltage
VSET<X>?	Returns the output voltage setting
IOUT<X>?	Returns the actual output current
VOUT<X>?	Returns the actual output voltage
TRACK<NR1>	Selects the operation mode
BEEP<BOOLEAN>	Turn on or off the beep
OUT<BOOLEAN>	Turn on or off the output
STATUS?	Returns the MODEL status

*IDN?	Returns the MODEL identification
RCL<NR1>	Recalls a panel setting
SAVE<NR1>	Saves the panel setting
HELP?	Shows the command list
ERR?	Returns the instrument error messages

5.6 Command Details

Command	**ISET<X>:<NR2>**
Description	Sets the output current.
Panel operation	Refer to page 11
Response time	Min.70ms
Example	ISET1:2.234 Sets the CH1 output current to 2.234A. (MPD-3303S) ISET1:2.23 Sets the CH1 output current to 2.23A. (MPD-3303D/3305D)

Command	**ISET<X>?**
Description	Returns the output current setting
Response time	Min.70ms
Example	ISET1? Returns CH1 output current setting.

Command	**VSET<X>:<NR2>**
Description	Sets the output voltage.
Panel operation	Refer to page 11
Response time	Min.70ms
Example	VSET1:20.345 Sets the CH1 voltage to 20.345V. (MPD-3303S) VSET1:20.3 Sets the CH1 voltage to 20.3V. (MPD-3303D/3305D)

Command	**VSET<X>?**
Description	Returns the output voltage setting.
Response time	Min.70ms
Example	VSET1? Returns the CH1 voltage setting.

Command	**IOUT<X>?**
Description	Returns the actual output current.
Response time	Min.70ms
Example	IOUT1? Returns the CH1 output current.

Command	**VOUT<X>?**
Description	Returns the actual output voltage.
Response time	Min.70ms
Example	VOUT1? Returns the CH1 output voltage.

Command	TRACK<NR1>
Description	Selects the operation mode: INDEP, tracking SER, tracking PAR
Panel operation	Refer to page 12
NR1	0: Independent 1: Tracking series 2: Tracking parallel
Response time	Min.70ms
Example	TRACK0 Selects the independent mode.

Command	BEEP<Boolean>
Description	Turns on or off the beeper.
Panel operation	Refer to page 10
Response time	Min.70ms
Example	BEEP1 Turns on the beeper.

Command	OUT<Boolean>
Description	Turns on or off the output.
Panel operation	Refer to page 10
Response time	Min.70ms
Example	OUT1 Turns on the output.

Command	STATUS?
Description	Returns the MODEL status.
Response time	Min.400ms
Contents	8 bits in the following format. (Refer to table on the right.)

Bit	Item	Description
0	CH1	0=CC mode, 1=CV mode
1	CH2	0=CC mode, 1=CV mode
2, 3	Tracking	01=Independent, 11=Tracking series，10=Tracking parallel
4	Beep	0=Off, 1=On
5	N/A	N/A
6	Output	0=Off, 1=On
7	N/A	N/A

Command	SAV<NR1>
Description	Saves the panel setting.
Panel operation	Refer to page 17
NR1	1~4: Memory 1 to 4
Reponses time	Min.70ms
Example	SAV1 Stores the panel setting into memory 1.

Command	RCL<NR1>
Description	Recalls a panel setting.
Panel operation	Refer to page 17
NR1	1~4: Memory 1 to 4
Reponses time	Min.70ms
Example	RCL1 Recalls the panel setting stored in memory 1.

Command	*IDN?
Description	Returns the MODEL identification
Reponses time	Min.300ms
Contents	Series number, firmware version

Command	HELP?
Description	Shows the command list.
Reponses time	Min.1000ms
Contents	Refer to the following table.

Command	ERR?
Description	Checks the error status of the instrument and returns the nearest error massage
Reponses time	Min.70ms
Contents	See page 19 for list of error messages.

Contents for Command HELP	
ISET<x>:<NR2>	Sets the value of current.
VSET<x>:<NR2>	Sets the value of voltage. X: 1=CH1, 2=CH2.
ISET<x>?	Return the value of current.
VSET<x>?	Return the value of voltage.
IOUT<x>?	Returns actual output current.
VOUT<x>?	Returns actual output voltage.
TRACK<NR1>	Sets the output of the power supply working on independent or tracking mode. NR1: 0=INDE, 1=SER, 2=PARA.
BEEP<Boolean>	Sets the BEEP state on or off.
OUT<Boolean>	Sets the output state on or off
STATUS?	Returns the power supply state.
bit0:(CH1)0=CC,1=CV	
bit1:(CH2)0=CC,1=CV	
bit23:(TRACK)01=INDEP, 11=SER,10=PAR	
bit4:(BEEP)0=OFF,1=ON	
bit6:(OUT)0=OFF,1=ON	
*IDN?	Returns instrument identification.
RCL<NR0>	Recall the setting data from the memory which previous saved.
SAV<NR0>	Saves the setting data to memory.
NR0: 1=Memory1, 2=Memory2, 3=Memory3, 4=Memory4;	
ERR? Returns instrument error messages.	

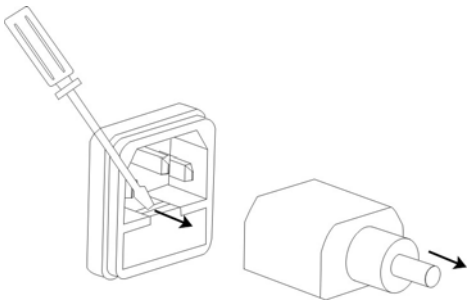
6. MAINTENANCE

6.1 Inspection

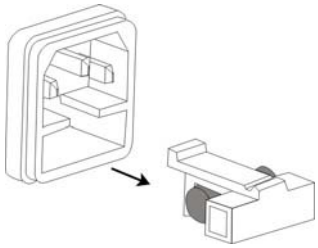
- Inspect the instrument at regular intervals so that it maintains its initial performance for a long time.
- Check the input power cord for damage of the vinyl cover and overheating of the plug and cord stopper. Check the terminal screws and binding posts for loosening.
- Remove dust from the inside of the casing and ventilation holes of the cover by using a compressed air of the exhaust air of a vacuum cleaner.

6.2 Fuse Replacement

Steps 1. Take off the power cord and remove the fuse socket using a minus driver.



2. Replace the fuse in the holder.



Fuse rating 110V: T6.3A/250V
220V: T3.15A/250V

6.3 Cleaning

- Before cleaning, disconnect the AC mains.
- To clean the power supply, use a soft cloth dampened in a solution of mild detergent and water. Do not spray cleaner directly onto the instrument, since it may leak into the cabinet and cause damage.
- Do not use chemicals containing benzene, toluene, xylene, acetone, or similar solvents.
- Do not use abrasive cleaners on any portion of the instrument.

7. FAQ

Q1: I pressed the panel lock key but the output still turns on/off.

A1: The output key is not affected by the panel lock key operation, for ensuring safety.

Q2: The CH3 overload indicator turned on - is this an error?

A2: No, it simply means that the CH3 output current reached the maximum 3.0A and the operation mode turned from CV (constant voltage) to CC (constant current). You can continue using the power supply, although reducing the output load is recommended.

Q3: The specifications do not match the real accuracies.

A3: Make sure that the instrument is powered on for at least 30 minutes, within ambient temperature of +20℃~+30℃

Q4: The internal memory is not recording the panel setting correctly - the output should be on.

A4: The output is always stored or recalled as "off" to ensure safety.

SPECIFICATIONS (MPD-3303S)

Output ratings

CH1/CH2 independent: 0~30V, 0~3A

CH1/CH2 series: 0~60V, 0~3A

CH1/CH2 parallel: 0~30V, 0~6A

CH3: 2.5V/3.3V/5V, 3A

Constant voltage operation

Line regulation: ≤0.01%+3mV

Load regulation: ≤0.01%+3mV (I≤3A) / ≤0.02%+5mV (I>3A)

Recovery time: ≤100us (50% load change, minimum load 0.5A)

Ripple & Noise: ≤1mV rms (I≤3A) (5Hz~1MHz) / ≤2mV rms (I>3A) (5Hz~1MHz)

Temp.co-efficient: ≤300ppm/℃

Constant current operation

Line regulation: ≤0.2%+3mA

Load regulation: ≤0.2%+3mA (I≤3A) / ≤0.2%+5mA (I>3A)

Ripple & Noise: ≤3mA rms (I≤3A) / ≤6mA rms (I>3A)

Tracking parallel operation

Line regulation: ≤0.01%+3mV

Load regulation: ≤0.01%+5mV (I≤3A) / ≤0.02%+10mV (I>3A)

Tracking series operation

Line regulation: ≤0.01%+5mV

Load regulation: ≤300mV

Tracking error: ≤0.5%+10mV of the master (No load. With load, add load regulation≤300mV)

CH3 output

Line regulation: ≤25mV

Load regulation: ≤25mV

Ripple & Noise: ≤2mV rms

Output voltage: 2.5V, 3.3V, 5V, ±8%

Output current: 3A

Display

Ammeter: 3.200A full scale, 4 digits 0.4" LED display

Voltmeter: 32.000V full scale, 5 digits 0.4" LED display

Voltmeter resolution: 1mV

Ammeter resolution: 1mA

Programming accuracy: ±(0.03% of reading + 10mV) (0~30V)

(25±5℃) ±(0.3% of reading + 10mA (I≤3A)

Readback accuracy: ±(0.03% of reading + 10mV) (0~30V)

(25±5℃) ±(0.3% of reading + 10mA (I≤3A)

Protection: Over load and reverse polarity protections

Insulation: Between base and output terminal≥20MΩ/500VDC

Between base and power cord≥30MΩ/500VDC

Operation environment: Indoor use

Altitude: ≤2000m

Ambient temperature: 0~40℃

Relative humidity: ≤80%

Installation category: II

Pollution degree: 2

Storage environment: Ambient temperature: -10~70℃

Relative humidity: ≤70%

Power source: AC 110V/220V±10%, 50/60Hz

Accessories: User manual×1, power cord×1, USB cable, USB interface software CD

Dimensions: 310(D)*250(W)*150(H)mm

Weight: 7.5kg

SPECIFICATIONS (MPD-3303/3305)

Output ratings

CH1/CH2 independent: 0~30V, 0~3A (I≤3A) / 0~30V, 0~5A (I>3A)

CH1/CH2 series: 0~60V, 0~3A (I≤3A) / 0~60V, 0~5A (I>3A)

CH1/CH2 parallel: 0~30V, 0~6A (I≤3A) / 0~30V, 0~10A (I>3A)

CH3: 2.5V/3.3V/5V, 3A

Constant voltage operation

Line regulation: ≤0.01%+3mV

Load regulation: ≤0.01%+3mV (I≤3A) / ≤0.02%+5mV (I>3A)

Recovery time: ≤100us (50% load change, minimum load 0.5A)

Ripple & Noise: ≤1mV rms (I≤3A) (5Hz~1MHz) / ≤2mV rms (I>3A) (5Hz~1MHz)
Temp.co-efficient: ≤300ppm/℃

Constant current operation
Line regulation: ≤0.2%+3mA
Load regulation: ≤0.2%+3mA (I≤3A) / ≤0.2%+5mA (I>3A)
Ripple & Noise: ≤3mA rms (I≤3A) / ≤6mA rms (I>3A)

Tracking parallel operation
Line regulation: ≤0.01%+3mV
Load regulation: ≤0.01%+5mV (I≤3A) / ≤0.02%+10mV (I>3A)

Tracking series operation
Line regulation: ≤0.01%+5mV
Load regulation: ≤300mV
Tracking error: ≤0.5%+50mV of the master (No load. With load, add load regulation≤300mV)

CH3 output
Line regulation: ≤25mV
Load regulation: ≤25mV
Ripple & Noise: ≤2mV rms
Output voltage: 2.5V, 3.3V, 5V, ±8%
Output current: 3A

Display
Ammeter: 3.20A full scale, 3 digits 0.5” LED display (MPD-3303)
5.10A full scale, 3 digits 0.5” LED display (MPD-3305)
Voltmeter: 32.0V full scale, 3 digits 0.5” LED display
Voltmeter resolution: 10mV(0~9.99V), 100mV(10~30V)
Ammeter resolution: 10mA
Programming accuracy: ±(0.2% of reading + 3digits) (0~9.99V)
(25±5℃) ±(0.5% of reading + 2digits) (10~30V)
±(0.5% of reading + 2digits) (I≤3A)
±(0.5% of reading +5digits) (I>3A)

Readback accuracy: ±(0.2% of reading + 3digits) (0~9.99V)
(25±5℃) ±(0.5% of reading + 2digits) (10~30V)
±(0.5% of reading + 3digits) (I≤3A)
±(0.5% of reading +5digits) (I>3A)

Protection: Over load and reverse polarity protections
Insulation: Between base and output terminal≥20MΩ/500VDC
Between base and power cord≥30MΩ/500VDC
Operation environment: Indoor use
Altitude: ≤2000m
Ambient temperature: 0~40℃

Relative humidity: ≤80%
Installation category: II
Pollution degree: 2

Storage environment: Ambient temperature: -10~70℃
Relative humidity: ≤70%

Power source: AC 110V/220V±10%, 50/60Hz

Accessories: User manual×1, power cord×1

Dimensions: 310(D)*250(W)*150(H)mm

Weight: 7.5kg (MPD-3303) / 10kg (MPD-3305)

SPECIFICATIONS (MPD-3303D/3305D)

Output ratings

CH1/CH2 independent: 0~30V, 0~3A (I≤3A) / 0~30V, 0~5A (I>3A)

CH1/CH2 series: 0~60V, 0~3A (I≤3A) / 0~60V, 0~5A (I>3A)

CH1/CH2 parallel: 0~30V, 0~6A (I≤3A) / 0~30V, 0~10A (I>3A)

CH3: 2.5V/3.3V/5V, 3A

Constant voltage operation

Line regulation: ≤0.01%+3mV

Load regulation: ≤0.01%+3mV (I≤3A) / ≤0.02%+5mV (I>3A)

Recovery time: ≤100us (50% load change, minimum load 0.5A)

Ripple & Noise: ≤1mV rms (I≤3A) (5Hz~1MHz) / ≤2mV rms (I>3A) (5Hz~1MHz)

Temp.co-efficient: ≤300ppm/℃

Constant current operation

Line regulation: ≤0.2%+3mA

Load regulation: ≤0.2%+3mA (I≤3A) / ≤0.2%+5mA (I>3A)

Ripple & Noise: ≤3mA rms (I≤3A) / ≤6mA rms (I>3A)

Tracking parallel operation

Line regulation: ≤0.01%+3mV

Load regulation: ≤0.01%+5mV (I≤3A) / ≤0.02%+10mV (I>3A)

Tracking series operation

Line regulation: ≤0.01%+5mV

Load regulation: ≤300mV

Tracking error: ≤0.5%+50mV of the master (No load. With load, add load regulation≤300mV)

CH3 output

Line regulation: ≤25mV

Load regulation: ≤25mV

Ripple & Noise: ≤2mV rms

Output voltage: 2.5V, 3.3V, 5V, ±8%

Output current: 3A

Display

Ammeter: 3.20A full scale, 3 digits 0.5” LED display (MPD-3303D)

5.10A full scale, 3 digits 0.5” LED display (MPD-3305D)

Voltmeter: 32.0V full scale, 3 digits 0.5” LED display

Voltmeter resolution: 100mV

Ammeter resolution: 10mA

Programming accuracy: ±(0.5% of reading + 2digits)(0~30V) ,

(25±5℃) ±(0.5% of reading + 2digits) (I≤3A)

±(0.5% of reading + 5digits) (I>3A)

Readback accuracy: ±(0.5% of reading + 2digits)(0~30V) ,

(25±5℃) ±(0.5% of reading + 3digits) (I≤3A)

±(0.5% of reading + 5digits) (I>3A)

Protection: Over load and reverse polarity protections

Insulation: Between base and output terminal≥20MΩ/500VDC

Between base and power cord≥30MΩ/500VDC

Operation environment: Indoor use

Altitude: ≤2000m

Ambient temperature: 0~40℃

Relative humidity: ≤80%

Installation category: II

Pollution degree: 2

Storage environment: Ambient temperature: -10~70℃

Relative humidity: ≤70%

Power source: AC 110V/220V±10%, 50/60Hz

Accessories: User manual×1, power cord×1, USB cable, USB interface software CD

Dimensions: 310(D)*250(W)*150(H)mm

Weight: 7.5kg (MPD-3303D) / 10kg (MPD-3305D)

SPECIFICATIONS (MPD-3303C/3305C)

Output ratings

CH1/CH2 independent: 0~30V, 0~3A (I≤3A) / 0~30V, 0~5A (I>3A)

CH1/CH2 series: 0~60V, 0~3A (I≤3A) / 0~60V, 0~5A (I>3A)

CH1/CH2 parallel: 0~30V, 0~6A (I≤3A) / 0~30V, 0~10A (I>3A)

CH3: 2.5V/3.3V/5V, 3A

Constant voltage operation

Line regulation: ≤0.01%+3mV

Load regulation: ≤0.01%+3mV (I≤3A) / ≤0.02%+5mV (I>3A)

Recovery time: ≤100us (50% load change, minimum load 0.5A)

Ripple & Noise: ≤1mV rms (I≤3A) (5Hz~1MHz) / ≤2mV rms (I>3A) (5Hz~1MHz)

Temp.co-efficient: ≤300ppm/℃

Constant current operation

Line regulation: ≤0.2%+3mA
Load regulation: ≤0.2%+3mA (I≤3A) / ≤0.2%+5mA (I>3A)
Ripple & Noise: ≤3mA rms (I≤3A) / ≤6mA rms (I>3A)

Tracking parallel operation
Line regulation: ≤0.01%+3mV
Load regulation: ≤0.01%+5mV (I≤3A) / ≤0.02%+10mV (I>3A)

Tracking series operation
Line regulation: ≤0.01%+5mV
Load regulation: ≤300mV
Tracking error: ≤0.5%+100mV of the master (No load. With load, add load regulation≤300mV)

CH3 output
Line regulation: ≤25mV
Load regulation: ≤25mV
Ripple & Noise: ≤2mV rms
Output voltage: 2.5V, 3.3V, 5V, ±8%
Output current: 3A

Display
Ammeter: 3.20A full scale, 3 digits 0.5" LED display (MPD-3303C)
5.10A full scale, 3 digits 0.5" LED display (MPD-3305C)
Voltmeter: 32.0V full scale, 3 digits 0.5" LED display
Voltmeter resolution: 100mV
Ammeter resolution: 10mA
Programming accuracy: ±(0.5% of reading + 2digits)(0~30V) ,
(25±5℃) ±(0.5% of reading + 2digits) (I≤3A)
±(0.5% of reading + 5digits) (I>3A)

Readback accuracy: ±(0.5% of reading + 2digits)(0~30V) ,
(25±5℃) ±(0.5% of reading + 3digits) (I≤3A)
±(0.5% of reading + 5digits) (I>3A)

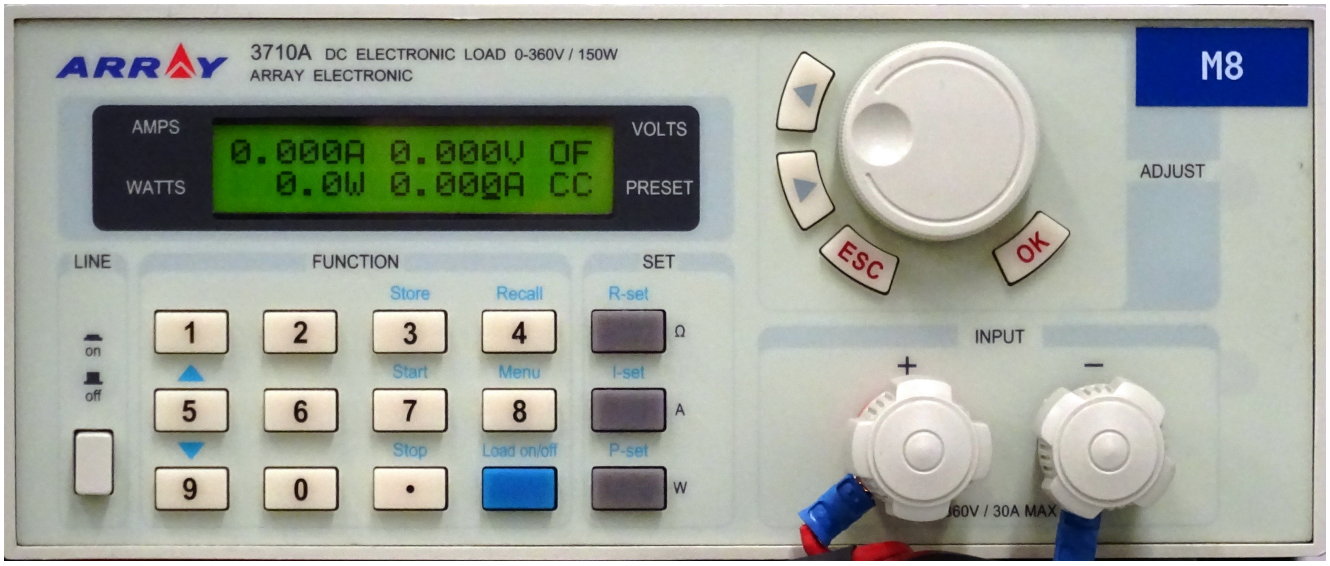
Protection: Over load and reverse polarity protections
Insulation: Between base and output terminal≥20MΩ/500VDC
Between base and power cord≥30MΩ/500VDC
Operation environment: Indoor use
Altitude: ≤2000m
Ambient temperature: 0~40℃
Relative humidity: ≤80%
Installation category: II
Pollution degree: 2
Storage environment: Ambient temperature: -10~70℃
Relative humidity: ≤70%
Power source: AC 110V/220V±10%, 50/60Hz

Accessories: User manual×1, power cord×1, USB cable, USB interface software CD
Dimensions: 310(D)*250(W)*150(H)mm
Weight: 7.5kg (MPD-3303C) / 10kg (MPD-3305C)

Apéndice C

Manuales de los instrumentos

C.2 Carga electrónica Array 3710A, Manual de usuario



PROGRAMMABLE DC ELECTRONIC LOAD

3700 SERIES

user's manual

VERSION 2.0 2006 .03

SAFETY GUIDE

This manual contains the precautions necessary to ensure your personal safety as well as for protection for the products and the connected equipment. These precautions are highlighted with a triangle "**WARNING**" symbol in this manual and are marked according to the danger levels as follows:

Danger:

It indicates that if appropriate precautions are not taken, serious incidents of personal injuries and deaths or significant damages or losses to the properties will be caused.

Caution:

It indicates that if appropriate precautions are not taken, injuries or losses of properties will take place.

Note:

Remind you to pay particular attention to the important information related to the products, disposal of products or the specific part of documents.

Warning:

Only qualified personnel are allowed to debug and operate this equipment. The qualified personnel are specified as those personnel who carry out commissioning, grounding and apply the volume identification to the circuits, equipment and systems according to the available safety practices and standards.

Note:

Only when this product is transported, stored assembled and installed in a proper way and operated and maintained according to the recommendations, can it implement the functions properly and reliably.

The following general safety precautions must be observed during all phases of operation, service, and repair of this instrument. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the instrument. The manufacturer assumes no liability for the customer's failure to comply with these requirements.

Attention

I. Safety

1. Users should operate according to this manual.

2. There is high voltage inside the instrument, please avoid touching it directly.

3. Please read the user manual carefully before you use the instrument to assure your safety.

4. Ground the Instrument

This product is provided with a protective earth terminal. To minimize shock hazard, the instrument chassis and cabinet must be connected to an electrical ground. The instrument must be connected to the AC power supply mains through a three-conductor power cable, with the third wire firmly connected to an electrical ground (safety ground) at the power outlet.

5.Keep Away From Live Circuits

Operating personnel must not remove instrument covers. Component replacement and internal adjustments must be made by qualified service person. Do not split the components when power cable connected.

Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power, discharge circuits and remove external voltage sources before touching components.

6.Do Not Substitute Parts or Modify Instrument

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification to the instrument. Return the instrument to a qualified dealer for service and repair to ensure that safety features are maintained.

II. Connecting the power line

1. Inspect the power selection switch on the back panel, to assure if the selected voltage is the same as environment power voltage. If not, please refer to the notice around the power plug.

2. Before connecting the power supply, please be sure that the switch on the front panel should be in the off position.

3. Connect the power cable to the AC in the power supply and the 3 connector plug. Please be sure that the AC power supply is grounded.

4. Press the switch on the front panel to switch on the instrument and you can start to use it.

III. Fuse

1. The fuse is at the position of power input part on the back panel, for the purpose of avoiding failure brought by using wrong voltage. Please pay attention to the following items when you change the voltage input and the fuse.

2. Before changing the voltage and fuse, please be sure that the AC power is switching off, and be sure that there is no other equipment connected to this instrument.

3. Put a screwdriver into the fuse mounting, press it and the fuse mounting will pop up.

4. Pull out the fuse, change it according to the label beside the power input plug.

Warning: To avoid damaging the instrument, please be sure to change a suitable type fuse.

Programmable DC Electronic Load

3700 SERIES

Table of Contents

PART1:3700 Series Electronic Load

Chapter 1 General Introduction……..……..…1

1.1 General Introduction……..…1

1.2 Specification……..……1

1.3 Features……..……2

1.4 Dimension and Structure……2

1.4.1 Dimension……...2

1.4.2 Structure…..……3

1.4.2.1 Front view……3

1.4.2.2 Back view……4

Chapter 2 Operation……..5

2.1 Connect the power supply to the electronic load……..…5

2.2 Main functions……..……5

2.2.1 Constant Current Mode……..…5

2.2.2 Constant Power Mode……..…7

2.2.3 Constant Resistance Mode……..…7

2.2.4 Store a program……...8

2.2.5 Recall a program……9

2.2.6 Start a program……9

2.2.7 Stop a program……9

2.2.8 Load On/Off……9

2.3 Menu operation……...9

2.3.1 Setting the maximum current……10

2.3.2 Setting the maximum power……10

2.3.3 Setting Minimum Input Voltage……...10

2.3.4 Setting Baud Rate……10

2.3.5 Setting Address (0~254)……...10

2.3.6 Enable / disable the Rotary knob……..……11

2.3.7 Program……..…11

2.3.8 Save option……12

2.3.9 Lock the keyboard……..……13

2.3.10 Load Default……..…13

2.3.11 Exit……..……13

PART 2: Electronic Load Software……14
Chapter 1 Software Installation……15
1.1 Connect the electronic load to PC……15
1.2 Installation……15
Chapter 2 Introduction……19
2.1 Running the program……19
2.2 The Eload List……20
2.3 The Eload Mode List……20
2.4 Control Switch……20
2.5 The Tool Buttons……21
2.6 Keypad and Rotary dial……21
2.7 The Amp, Power, Resistance, Voltage Meter……22
2.8 Dynamic Curve……22
2.9 Setting……23
2.10 Programming Window……24
2.11 Eload Setting……24
2.12 Data Report……26
2.13 Use Electronicload software control your load……27
Chapter 3 Uninstall ElectronicLoad Software……28
3.1 Uninstall the ElectronicLoad Software……28

Chapter 1 General Introduction

1.1 General Introduction

Array 3700 series electronic load is a single input programmable DC electronic load. It provides a convenient way to test DC power supplies and batteries. It offers constant current mode, constant resistance mode and constant power mode. Program operation and control by PC is also available. The backlight LCD, numerical keypad and rotary knob make it much easier to use. It is an essential instrument for designing, testing and manufacturing of many suitable products.

1.2 Specifications

Model	3710A	3711A
Number of Input	1	1
Input Voltage	0~360V	0~360V
Input Power	0~150W	0~300W
Input Current	0~30A	0~30A
Voltage Accuracy	0.000-3.999V: ± 0.2%+3mV 4.00-35.999V ± 0.2%+30mV 36.0-360.0V:±0.2%+300mV	
Current Accuracy	0.000-2.999A: ±0.2%+3mA 3.00-30.00: ±0.2%+30mA	
Max Resolution	Voltage: 1mV Current: 1mA	
Minimum Conductive Resistance	< 0.08 Ω	
Ripple	< 10mVpp	
Communication Interface	RS232/RS485 /USB*	
Software	Free application software ElectronicLoad, ActiveX control support VC++ / VB / DELPHI / LABVIEW	
Program Memory	9 programs, 10 steps each	
Protective Mode	Over voltage/over current/over power/over heat/polarity-reverse	
AC Input	110/220V AC switch able (60/50HZ)	
Weight	5.0kg	
Accessories	Software, users manual, AC power cable, handlebars	
Optional Parts	3311 isolated TTL to RS232 adaptor 3312 isolated TTL to USB adaptor 3313 isolated RS232 to RS485 adaptor 3314 isolated TTL to RS485 adaptor Mounting rack	

1.3 Features

1. LCD display with backlight.
2. High resolution measurement.
3. CC/CP/CR mode.
4. Number keypad and rotary knob.
5. Multifunction menu.
6. Over voltage / over current / over power / over temperature / polarity reversion protection.
7. 10 steps program.
8. Compact and light-weight.
9. Can be used in parallel connection.
10. Can be controlled by PC.

1.4 Dimensions and Structure

1.4.1 Dimensions (unit: mm)

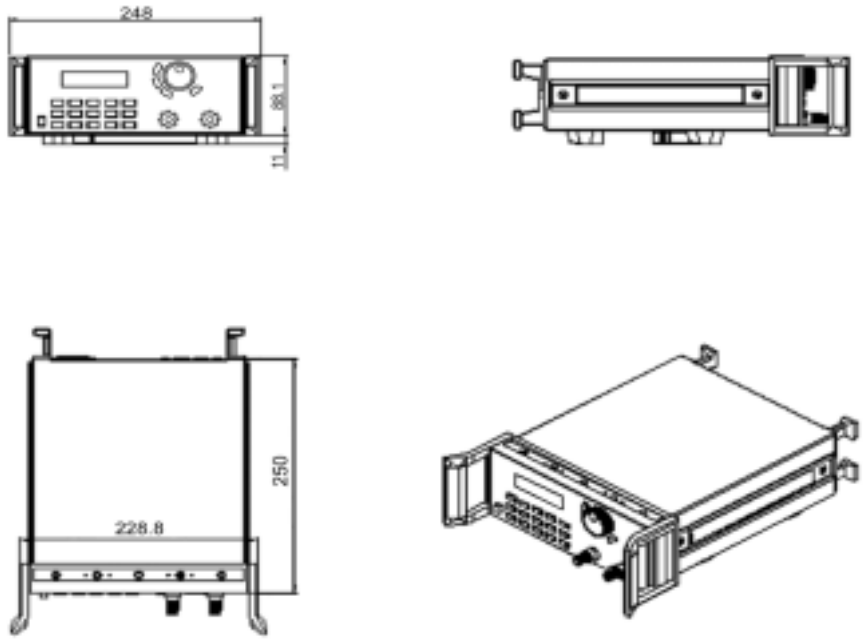


Fig.1-1 Dimension of 3700 Series Programmable DC electronic load

1.4.2 Structure

1.4.2.1 Front view

Front panel is for users to operate, and there is one LCD display, one number keypad and one rotary knob. Please see the following pictures.

Fig.1-2 Front view of 3700 Series DC electronic load

1. LCD display

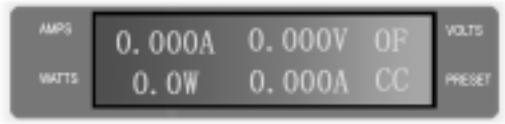


Fig.1-3 LCD Display of 3700 Series electronic load

The 1st line:
Current, voltage, On/Off state

The 2nd line:

Power, load setting value, working mode.

CC, CP, CR represent constant current, constant power, constant resistance mode.

2. Arrangement of the Keyboard

In general, the keyboard will perform the function of the black words 1st. But in the 2nd mode, it will perform the function of the words in blue.

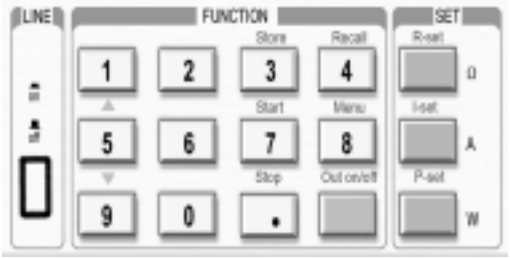


Fig.1-4 Keyboard of 3700 Series DC electronic load

0~9: The number keys

Load on/off: Load ON/OFF switch

R-set: Setting resistance value

I-set: Setting current value

P-set: Setting power value

Store: Store current program

Recall: Recall a saved program

Start: Start a program

Stop: Stop running program

Menu: Menu function operation

▲: The up moving key

▼: The down moving key

3. Rotary knob and function keys

Fig.1-5 Rotary knob and function keys

◀ : The left moving key

▶ : The right moving key

ESC: Escape key

OK: Confirmation key

Rotary knob: The rotation dial

1.4.2.2 Back view

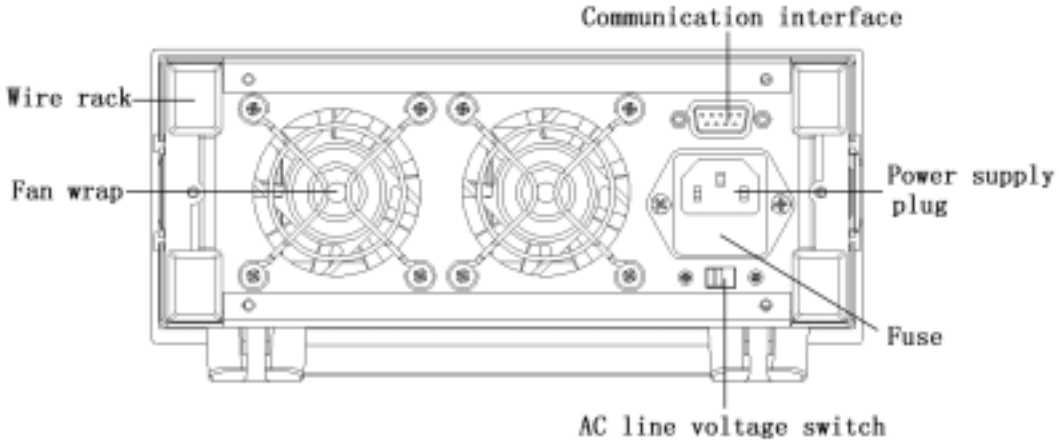


Fig.1-6 Back view

The fuse can be changed easily by using a small flat screw driver. Please use a fuse within the range of 0.3-0.5A.

Chapter 2 Operation

2.1 Connect the power supply to the electronic load

Fig.2-1 Connect the power supply to the electronic load

2.2 Main functions

1. Constant current mode
2. Constant power mode
3. Constant resistance mode
4. Store a program
5. Recall a program
6. Start a program
7. Stop a program
8. Load On/Off

2.2.1 Constant Current Mode

In constant current mode, the electronic load will sink a constant current in accordance with the set value regardless of input voltage. The current value can be set with the keyboard and/or rotary knob. Please see the following operation procedure:

Constant current setting procedure:

Procedure	Operation details	LCD display
Step 1	Press “I-set”	Input Password
Step 2	Enter the password (if the keyboard isn’t locked, please go to step4)	Input Password ****
Step 3	Press the “OK” button, the original set value will be displayed on the LCD. (It will return to step 2 if your password is wrong)	Set Curr = 2.00A New=
Step 4	Enter the new current value with the number keypad or rotate the dial to adjust the value	Set Curr = 2.00A New= 3.00
Step 5	Press “I-set” button for confirmation	Set Curr =3.00A New= 3.00
Step 6	Press ESC to go back to the main menu	
You can exit the current set operation at any point by pressing the ESC key		

For example, set constant current to 3.12A

1. Setting by number keyboard

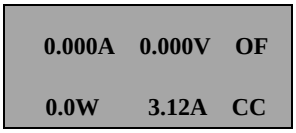
Step1. Press “ I-set ” button,

Step2. Enter the password by using the number keyboard (if the keyboard isn’t locked, please go to step4),

Step3. Press “ OK ” button (if the password is wrong, please go to step2 for reentering),

Step4. Press “ 3 ”, “ **.** ”, “ 1 ” and “ 2 ” button to enter the current value,

Step5. Press “ I-set ” button to confirm the current value.



2. Setting by Rotary Knob

(1) If the keyboard isn’t locked, press “I-set” button, then directly rotate the “Rotary knob”, the current value will be continually changed from the previous value according to the rotation. At the beginning, the cursor will be shown on the last number of the value which is indicated on the LCD, you can move the cursor to the first number, second number etc by using “ ◀ ” and“ ▶ ” buttons, and then rotate to change each number, and let it stay at 3.12 A.

(2) If the keyboard is locked

Step1. Press " I-set " button,

Step2. Enter the password by using the number keyboard,

Step3. Press OK button (if the password is wrong, please go to step2 for reentering),

Step4. Rotate the Rotary to change the value，the operation is the same as item (1)

Step5. Press " I-set " button to confirm the current value.

2.2.2 Constant Power Mode

In constant power mode, the load's current will increase automatically if the input voltage is decreased or vice versa, so that the electronic load will sink a constant power in accordance with the set value. The power value can be set with the keyboard and/or rotary knob. Please see the following operation procedure:

Constant power setting procedure:

Procedure	Operation details	LCD display
Step 1	Press "P-set"	Input Password
Step 2	Enter the password (if the keyboard isn't locked, please go to step4)	Input Password ****
Step 3	Press the "OK " button, the original set value will be displayed on the LCD. (It will return to step 2 if your password is wrong)	Set Power = 20.0W New=
Step 4	Enter the new power value with the number keypad or rotate the dial to adjust the value	Set Power = 20.0W New= 30.0
Step 5	Press "P-set" button for confirmation	Set Power = 30.0W New= 30.0
Step 6	Press ESC to go back to the main menu	
You can exit the power set operation at any point by pressing the ESC key		

2.2.3 Constant Resistance Mode

In constant resistance mode, the electronic load will sink a current linearly proportional to the input voltage in accordance with the set value. The resistance value can be set with the keyboard and/or rotary knob. Please see the following operation procedure:

Constant resistance setting procedure:

Procedure	Operation details	LCD display
Step 1	Press “R-set” button	Input Password
Step 2	Enter the password (if the keyboard isn’t locked, please go to step4)	Input Password ****
Step 3	Press the “OK ” button, the original set value will be displayed on the LCD. (It will return to step 2 if your password is wrong)	Set Resis = 200.0Ω New=
Step 4	Enter the new resistance value with the number keypad or rotate the dial to adjust the value	Set Resis = 200.0Ω New= 50.00
Step 5	Press “R-set” button for confirmation	Set Resis = 50.00Ω New= 50.00
Step 6	Press ESC to go back to the main menu	
You can exit the resistance set operation at any point by pressing the ESC key		

2.2.4 Store a program

The 3700 Series electronic load provides a program mode for dynamic testing. The user can program the 3700 series electronic load and define the load values (current, power, resistance) and duration of every step. For more details about the programming operation, please refer to 2.3.7.

User’s program may be stored in the 3700’s nonvolatile memory for later use. Up to 9 programs can be stored in the 3700. To store a program, please follow this operation procedure:

Procedure	**Operation details**	**LCD display**
Step1	Press “ Store ” button	SAVE 1
Step 2	Enter the program number (1 to 9) by using the number key or the rotary knob	SAVE *
Step 3	Press the “ OK ” button for confirmation. It will return to Step 2 if the number is out of the range.	
You can exit the store operation at any point by pressing the ESC key		

2.2.5 Recall a program

To recall a program stored in the 3700, please follow this operation procedure:

Procedure	Operation details	LCD display
Step1	Press " Recall " button	RECALL 1
Step 2	Enter the program number (1 to 9) by using the number key or the rotary knob	RECALL *
Step 3	Press the " OK " button for confirmation. It will return to Step 2 if the number is out of the range.	
You can exit the recall operation at any point by pressing the ESC key		

2.2.6 Start a program

Press "Start" button to run the current program.

2.2.7 Stop a program

Press "Stop" button to stop the running program.

2.2.8 Load On/Off

The load's input can be toggled on/off through the Load On/Off key.

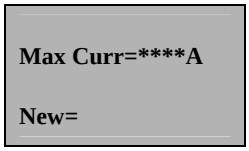
2.3 Menu operation

The Menu operation offers some setting and adjusting functions. The operation and functions are as following.

Procedure	Operation details	LCD display
Step1	Press " Menu " button	
Step 2	The LCD displays the menu functions. Use the UP and DOWN keys to select a function, then press " OK " for confirmation	Maximum Current Maximum Power Min Input Volt Band Rate Address Knob Enable Program Save Option Key Lock Load Default Exit
You can exit the menu operation at any point by pressing the ESC key		

2.3.1 Setting the maximum current

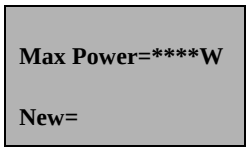
When you select the Maximum Current function, the LCD will display:



Set the maximum current value by using the number keyboard or the rotary knob, then press "OK" for confirmation.

2.3.2 Setting the maximum power

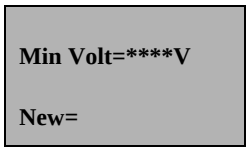
When you select the Maximum Power function, the LCD will display:



Set the maximum power value by using the number keyboard or the rotary knob, then press "OK" for confirmation.

2.3.3 Setting Minimum Input Voltage

This function is used to set the minimum input voltage, when the input voltage is lower than this value, the 3700 electronic load will automatically stop. This function is used to discharge batteries. When the battery's voltage drops down to the set value, the load switches off automatically, keeping the battery from over-discharge. When you select Min Input Volt function, the LCD will display:

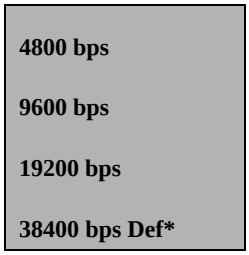


Set the minimum input voltage by using the number keyboard or the rotary knob, then press "OK" for confirmation.

2.3.4 Setting Baud Rate

This function is used to set the communication baud rate.

When you select the Baud Rate function, the LCD will display:

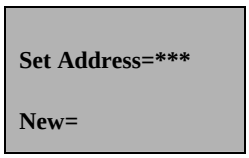


Select the Baud Rate by using the UP and DOWN keys or the rotary knob, then press "OK" for confirmation.

2.3.5 Setting Address (0~254)

This function is used to set the address of the load. To communicate with the computer or other equipment, the 3700 load must be assigned an address.

When you select Address set function, the LCD will display:



Set the address by using the number keyboard or the rotary knob, and then press "OK" for confirmation. The valid address range is from 0 to 254.

2.3.6 Enable/disable the Rotary knob

This function is to enable or disable the rotary knob.

Enable Def. *
Disable

Use the UP and DOWN keys or rotary knob to select, and press “OK” to confirm.

2.3.7 Program

3700 Series electronic load provides a program mode for dynamic testing. The user can program the 3700 series electronic load and define the load values (current, power, resistance) and duration of each program step. To program the 3700, please follow these 4 steps:

1、 Select one working mode: constant current, constant power, constant resistance;

2、 Set the length of the program;

3、 Set the load value (current, power, or resistance value) and duration time of each step;

4、 Select program running mode: one time or repeat.

For example, you need to set the following program:

Constant Current mode:

Step1 1A for 2 seconds

Step2 2A for 5 seconds

Step3 3A for 10 seconds

Repeat running step1 to step3.

Programming procedure is showed below:

1、 Select Program Set from menu operation, the LCD will display:

Constant Curr
Constant Power
Constant Resis

Use the UP and DOWN keys or rotary knob to select Constant Current mode, and press “OK” to confirm.

2、 LCD will display:

Step Number=*
New=

Input the program length “3” and press “OK” to confirm.

3、 LCD will display:

Step1 Set=0.00
New=

Input the current value “1” A, then press “OK” to confirm.

4、 LCD will display:

Step Time=1s
New=

Input the time value “2”s, then press “OK” to confirm.

5、 LCD will display:

Step 2 Set=0.00
New=

Input the current value “2” A, then press “OK” to confirm.

6、 LCD will display:

Step Time=1s
New=

Input the time value “5”s, then press “OK” to confirm.

7、 LCD will display:

Step 3 Set=0.00
New=

Input the current value “3” A, then press “OK” to confirm

8、 LCD will display:

Step Time=1s
New=

Input the time value “10”s, then press “OK” to confirm.

9、 LCD will display:

One Time
Repeat

Select Repeat, then press “OK” to confirm.

The programming process is completed. Press “ESC” to return to the main menu, press “Start” to run the program. If you want to store this program for later use, please see 2.2.4

2.3.8 Save Option

This function is used to save the last load setting (current, power and resistance) before switching off the 3700. When this option has been set, every time the user switches on the 3700, the previous load setting will be restored automatically.

Select the Save Option function, the LCD will display:

Save I,P,R
Don’t Save Def*

Use the UP and DOWN keys or the rotary knob to change the selection, then press “OK” to confirm. Select Save I, P, R, and load setting will be saved before the unit is switched off, and will be restored automatically when switched on the next time. Select Don’t Save def* represent not to save current load setting.

2.3.9 Lock the keyboard

This function is used to lock the keyboard, keep the load setting from unauthorized change. Only the person with correct password is allowed to operate the 3700.

Select the Key Lock function, the LCD will display:

Set Password

Enter a 4 numbers password, then press “OK ” to confirm.

2.3.10 Load Default

This function is used to restore all the parameters to the factory setting.

Select Load Default function, LCD will display:

Don’t Load
Load Default

Select Load Default, and press “OK” to restore to the factory setting.

2.3.11 Exit

Select “Exit” to exit the menu operation.

ElectronicLoad Software

ElectronicLoad is a free software which allow the user to monitor and control the Array 3700 series electronic load.

Chapter 1 Software Installation

1.1 Connect the electronic load to PC

With the optional communication adapter, you can connect the 3700 load to your computer through RS-232, USB or RS-485 port. Connect the 3700 to a RS-232 port with the 3311 adapter, to a USB port with the 3312, to a RS-485 with the 3314. If your computer doesn't have an RS-485 port, you may choose the 3313 RS-232=>RS-485 adapter.

Fig.1-1 Connect the electronic load to PC

1.2 Installation

1.2.1 Insert the Array software CD into the CD-ROM drive, double-click My Computer => Array CD => Eload => Setup.exe to start the installation, the Fig. 1-2 will be displayed.

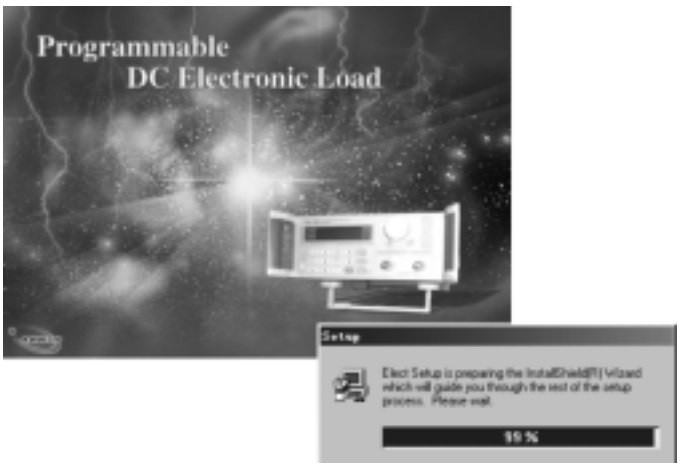


Fig.1-2 The Initialization of the Installation

1.2.2 The Welcome dialog box displays, press “ NEXT ” to continue

Fig.1-3 The Welcome dialog box

1.2.3 The License Agreement dialog box displays. Click Yes to accept the License Agreement and proceed with the installation process.

Note: If you do not want to accept the License Agreement, click No to abort the installation process.

Fig.1-4 The License Agreement dialog box

1.2.4 The User Information dialog box displays, Enter your name and your company’s name, then click Next.

Fig.1-5 The User Information dialog box

1.2.5 The destination folder dialog box displays. The default folder is " C:\Program Files\Array\ELoad ". You may click "Browse" to select other installation path.

Fig.1-6 The destination folder dialog box

1.2.6 The installation option dialog box displays. Select "Typical" to install all components or select "Custom" to select installation components.

Fig.1-7 The installation option dialog box

1.2.7 The Select Program Folder dialog box displays. Specify a program folder (or leave the default folder), then click Next.

Fig.1-8 The Select Program Folder dialog box

1.2.8 Copy the files to your hard disk.

Fig.1-9 Copy Files

Software installation is complete when finished.

Chapter 2 Introduction

2.1 Running the program

2.1.1 To run the ElectronicLoad program, select “Start=>Program=>Array=>Eload” and click “E3710A” or E3711A” to run the program.

Fig.2-1 Start the program

2.1.2 The ElectronicLoad startup pattern

Fig.2-2 ElectronicLoad Startup pattern

2.1.3 The ElectronicLoad Main Window

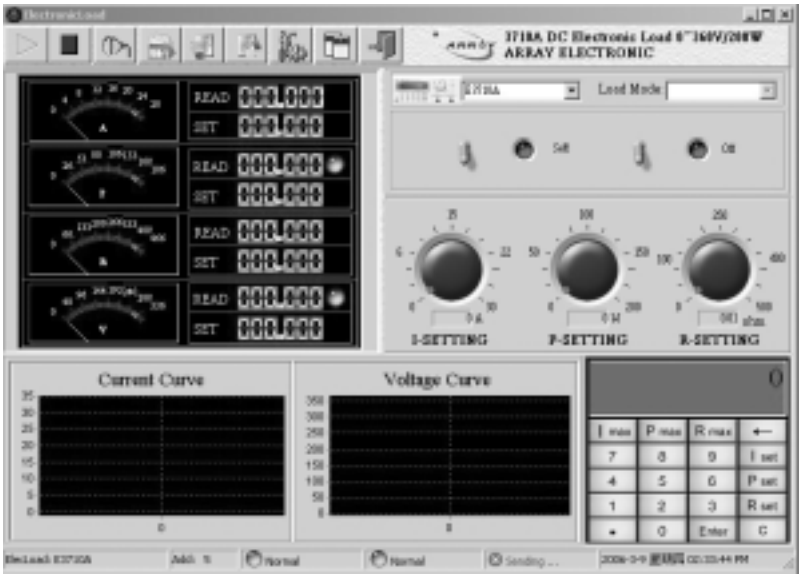


Fig.2-3 The ElectronicLoad Main Window

2.2 The ELoad List

The Eload list lists all registered electronic load on the computer. To make the ElectronicLoad control your load, choose the load you want to use from the Eload list.

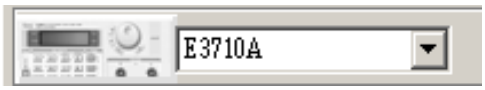


Fig.2-4 The Eload List

2.3 The Load Mode List

From the Load Mode list, you can select Load Mode: Constant Current, Constant Power, Constant Resistance.

Fig.2-5 The Load Mode List

2.4 Control Switch

The Load On/Off switch is to turn on/off the electronic load; the Self/PC Control switch is to set who control the electronic load: the PC or the load itself.

Fig.2-6 Control switch

2.5 The Tool Buttons

On the top left of the main window, there are 9 tool buttons:

 "Start": Start communication

 "Stop": Stop communication

 "Setting": setting COM Port and recording interval

 "Program": Setting a Program

 "Run Program": Run a Program

 "Stop Program": Stop Running Program

 "Eload Setting": Setting load parameters

 "Report": Data report

 "Quit": Quit the ElectronicLoad program

2.6 Keypad and Rotary dial:

Fig.2-7 Current, Power, Resistance setting dial

To quickly modify a parameter (Current, Power, Resistance), put the mouse on the red point of the dial and rotate. To accurately set a parameter, use the keypad.

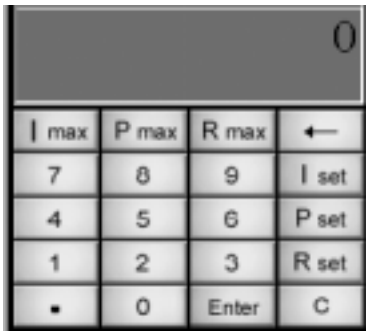


Fig.2-8 Keypad

Set Current: Press “I set”, input the current value (0~30A), press “ENTER” to confirm.

Set Power: Press “P set”, input the power value (3710A: 0~150W，3711A: 0~300W), press “ENTER” to confirm.

Set Resistance: Press “R set”, input the resistance value (0~500Ω), press “ENTER” to confirm.

I Max, P Max, R Max: show the max value of each parameter.

2.7 The Amp, Power, Resistance, Voltage Meter

The Amp, Power, Resistance, Voltage Meter show the setting and read back value of current, power, resistance and voltage.

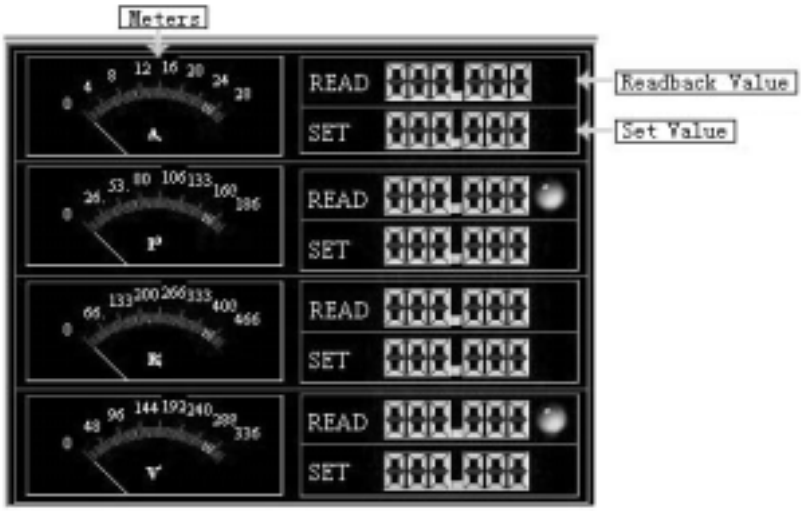


Fig.2-9 The Amp, Power, Resistance, Voltage Meter

2.8 Dynamic Curve

These curve show the dynamic states of voltage and current.

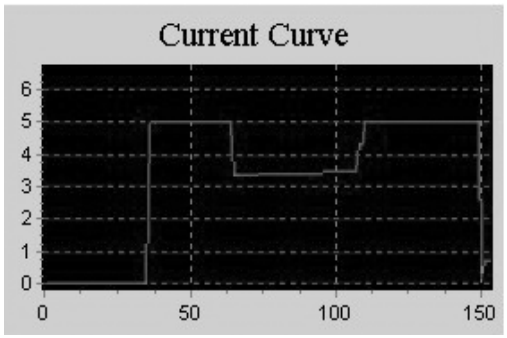


Fig.2-10 The Voltage Curve

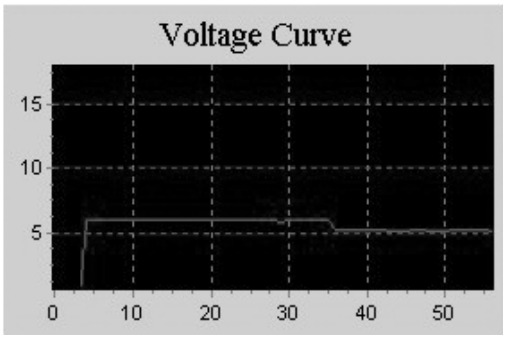


Fig.2-11 The Current Curve

You may drag your mouse on the curve to zoom in/out.

2.9 Setting

2.9.1 Setting Com port and baud rate

Click the button, select the correct Com port and baud rate. The baud rate setting should be the same as the connected electronic load.

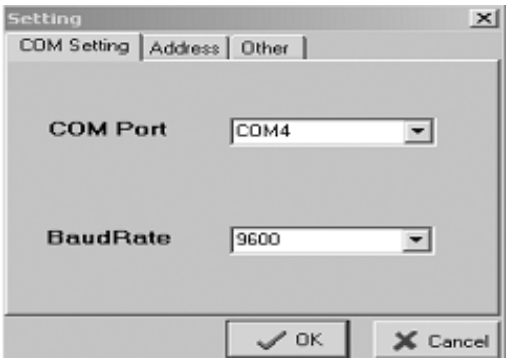


Fig.2-12 Setting Window

2.9.2 Setting electronic load address

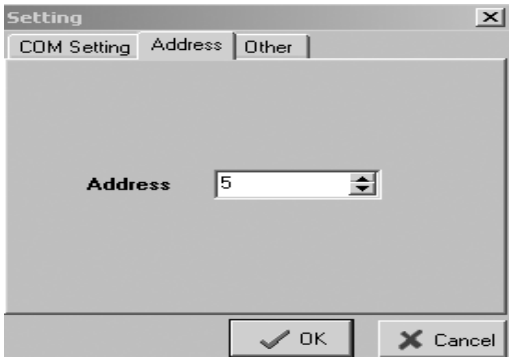


Fig.2-13 Address Setting window

Click “Address” to go to the Address setting window. Set an address that is the same as the connected electronic load. Press “OK” to confirm.

2.9.3 Setting Recording Interval

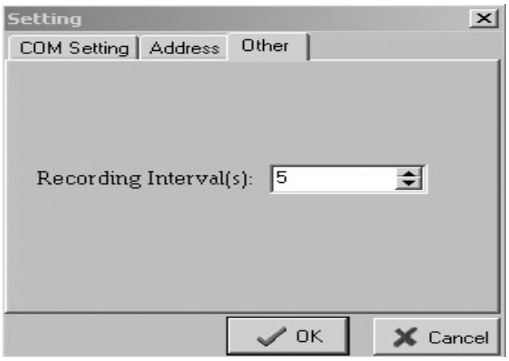


Fig.2-14 Setting Recording Interval window

2.10 Programming Window:

ElectronicLoad allows user to make an own program to control the 3700 load. User can select the load mode, and define load value (current, power, resistance) and duration of each program step.

Click the  button, the Program window will be displayed:

Fig.2-15 Programming window

Select an “Electronic Load Mode”; enter the load value and the duration of each program step.

The maximum program length is 10 steps. But user may decided only run part of them. Use the “Run Steps” push bar to decide how many steps you want to run.

Program State is used to define whether repeat to run this program.

When finishing, press “OK” to store the program and go back to the main window.

Press  button to run the program; press  button to stop the program.

2.11 Eload setting

Click the  button, the Eload setting window will be displayed.

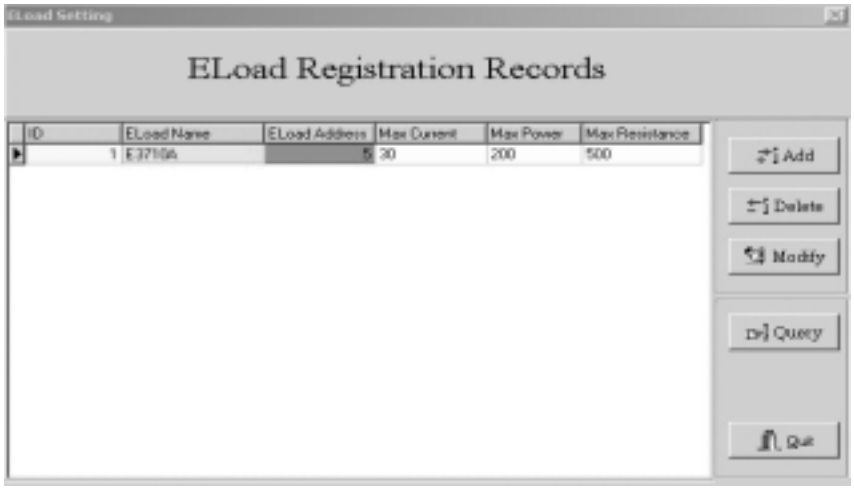


Fig.2-16 The Eload Setting window

The Eload Registration Records show the parameters of all registered load.

To add a new load, click the Add button.

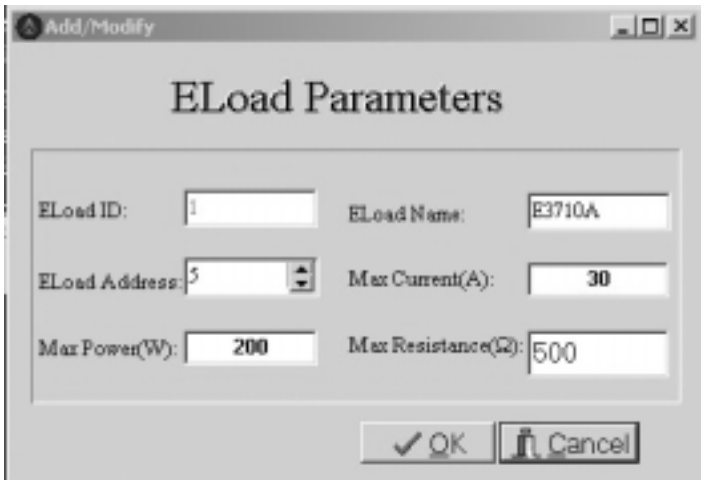


Fig.2-17 Add new load window

Enter the Eload Name, Address and the maximum value of current, power, resistance, then press "OK" to confirm. Note: the Eload ID is assigned by the software.

The Parameter Explanation

Parameter	Explanation	Range
Eload ID	ID	
Eload Name	Name	ASCII
Eload Address	Address	0-254
Max Current	Maximum Current	0-30A
Max Power	Maximum Power	0-150W /300W
Max Resistance	Maximum Resistance	0-500Ω

To delete a registered load, select the load from the Eload Registration Records, then click the Delete button.

To modify a load's parameters, select the load from the Eload Registration Records, then click the Modify button.

To query a load, click the Query button, then enter the load name you want to query.

Attention: after the Add, Modify, Delete operation, the program must restart so that the modifications may take effect. The following dialog box will be shown, click "Yes" to close the program, then restart the program from the Start menu.

Fig2-18 After Add, Modify, Delete operation, the program must restart

2.12 Data Report

Click the  button, the data report window will be shown:

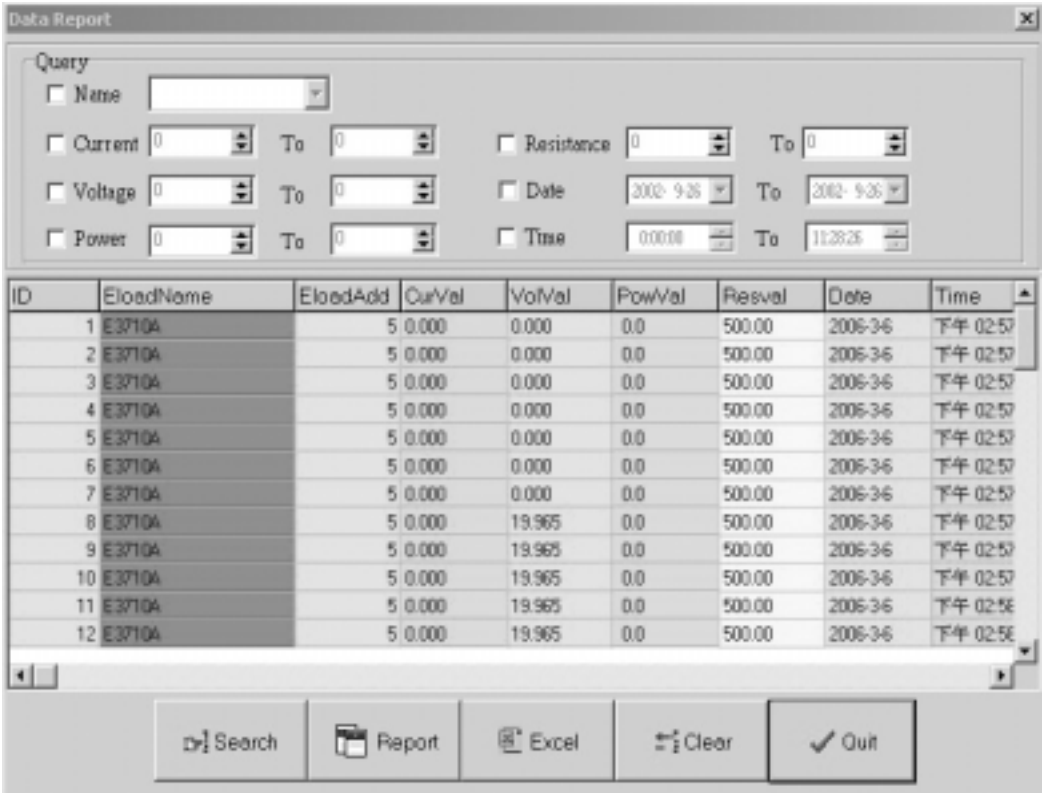


Fig2-19 The Data Report window

To search, set the query condition, and then click the “Search” button.

Click the “Report” button to watch the report.

Click the “Excel” button to export the report in Excel format.

To delete historic data, click the “Clear” button.

Quit: quit to the main window.

2.13 Use ElectronicLoad software control your load

Start the ElectronicLoad program, from the Eload List, select the load you want to use. If the desired load does not appear on the Eload List, use Eload Setting to add the desired load.

Click the  Setting button, ensure that the Com port, baud rate and address is the same as that of the load..

Click the  Start button to start communication. Using the Control Mode switch, set the load in PC control mode; You may now control the load from the computer.

Prior to quitting the ElectronicLoad program, do remember to return the setting of your load to the Self control mode, failing which, you will be unable to control your load using it's own keyboard.

In the event of communications failure, please check the Com port, baud rate and Address settings. Also ensure that the communication cable is firmly and correctly connected.

Chapter 3 Uninstall ElectronicLoad Software

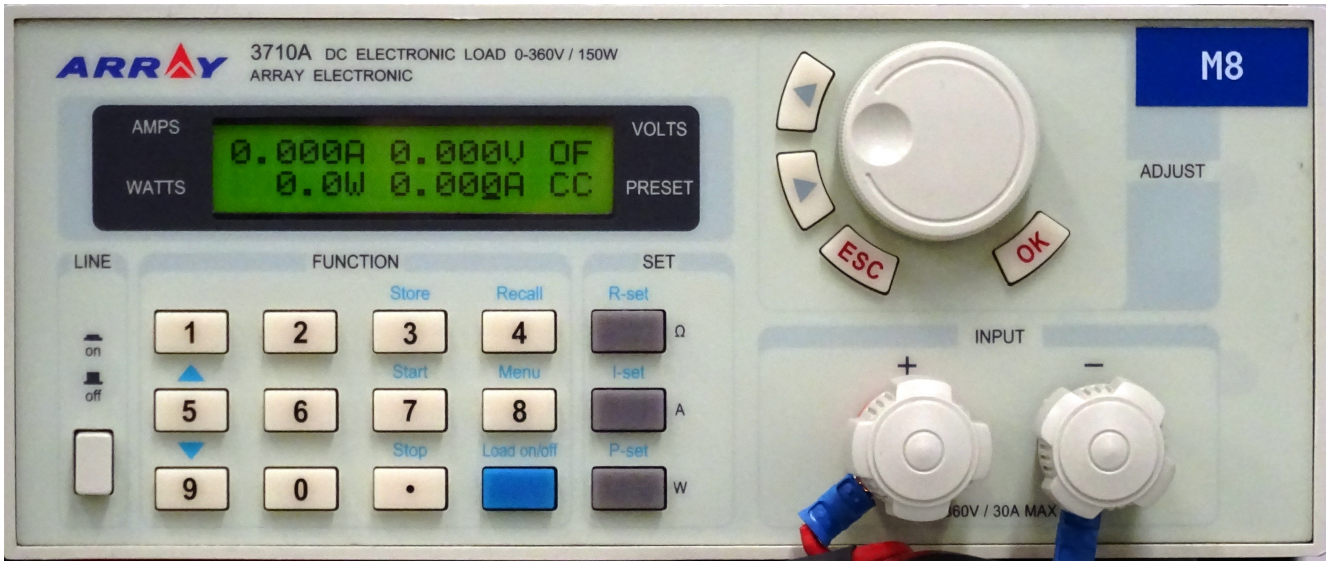
3.1 Uninstall the ElectronicLoad Software

From the Start Menu, select “Program=>Array=>Eload=>Uninstall Eload” to unstall the software. Remember to quit the program before uninstalling.

Apéndice C

Manuales de los instrumentos

C.3 Carga electrónica Array ·710A, Protocolo de comunicaciones



Control Protocol for Electronic Load Instrument

A. Default Serial Communications Port Settings

1) Baud Rate: 9600
2) Data Bits: 8
3) Stop Bits: 1
4) Handshake: None

B. Frame Format (applies to both transmitted and received data)

The frame length is 26 bytes with the following format:

AAh	Address	Command	Relative information: Bit 4 - 25	Checksum

Description of frame bytes:

1) The first byte of the frame is always AAh
2) The second byte is the instrument address (00h to FEh as set using front panel menu)
3) The third byte is the instrument control Command (90h to A0h)
   These are the possible commands:
   1) 90h------Set max current, max power and Set-value.
   2) 91h------Read current, Voltage, power and instrument's state
   3) 92h------To control the ON/OFF state of the load
   4) 93h------Programmed test sequence, define odd step1-5
   5) 94h------Programmed test sequence, define odd step 6-10
   6) 95h------Start programmed test sequence
   7) 96h------Stop programmed test sequence

4) Bytes 4 through 25 are the instrument data being sent or received
   1) the Voltage range of 0-360V is represented by an integer in the range of 0-360000.
   2) the current range of 0-30A is expressed as an integer in the range of 0-30000.
   3) the power range of 0-200W is expressed as an integer in the range of 0-2000.
   4) the resistance range of 0-500Ω is expressed as an integer in the range of 0-50000.

5) Byte 26 is the Checksum obtained by adding the values of the previous 25 bytes.

For example, this is the command you would use to query the state of the instrument …
AA 01 91 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 3C

C. Command Descriptions

90h, Set load operating parameters and maximum limits

Byte 1	AAh (frame start)
Byte 2	Address (00h - FEh)
Byte 3	90h (command)
Byte 4	Low byte of the maximum current
Byte 5	High byte of the maximum current
Byte 6	Low byte of the maximum power
Byte 7	High byte of the maximum power
Byte 8	New address of the Load (change address)
Byte 9	Type of set-value; 01h = current, 02h = power, 03h =resistance
Byte 10	Low byte of set-value
Byte 11	High byte of set-value
Byte 12 - 25	System Reserved
Byte 26	Checksum

The set-values for current, power and resistance are all expressed by two bytes. The Low byte is sent first. For example, the set-value 3589h is specified by the following sequence:

89h	35h

91h, Read current, voltage, power and resistance of the instrument

Byte 1	AAh (frame start)
Byte 2	Address (00h—FEh)
Byte 3	91h (command)
Byte 4	Low byte of the current
Byte 5	High byte of the current
Byte 6	Low byte of the low character of the voltage
Byte 7	High byte of the low character of the voltage
Byte 8	Low byte of the high character of the voltage
Byte 9	High byte of the high character of the voltage
Byte 10	Low byte of the power
Byte 11	High byte of the power
Byte 12	Low byte of the max current
Byte 13	High byte of the max current
Byte 14	Low byte of the max power
Byte 15	High byte of the max power
Byte 16	Low byte of the resistance value
Byte 17	High byte of the resistance value
Byte 18	Output state of the electronic load
Byte 19 - 25	System reserved
Byte 26	Checksum

The output state of the load is revealed by the individual bits of byte18:

From High to Low

b7	b6	b5	b4	b3	b2	b1	b0

b0: 0 = local (front panel) control ; 1 = remote (PC) control
b1: 0 = load OFF; 1 = load ON.
b2: 0 = correct polarity detected; 1 = wrong polarity detected
b3: 0 = temperature in acceptable range; 1 = excessive temperature

b4: 0 = Voltage acceptable; 1 = excessive Voltage.
b5: 0 = power acceptable; 1 = excessive power

Note: values of bytes 4 through 18 in command string will be ignored by instrument (suggest setting these to 00h); reply from instrument will have valid data in bytes 4 through 18.

92h, Activate or de-activate load and set local/remote control

Byte 1	AAh
Byte 2	Address (00h-FEh)
Byte 3	92h (command)
Byte 4	State of the electronic load
Byte 5- 25	System Reserved
Byte 26	Checksum

The desired state of the load is specified by the individual bits of byte4:

From High to Low

b7	b6	b5	b4	b3	b2	b1	b0

b0: 0 = load OFF; 1 = load ON
b1: 0 = go to local mode (front panel control); 1 = go to remote control (PC in control)

93h, Define programmed test sequence, steps 1-5

Byte 1	AAh
Byte 2	Address (00h—FEh)
Byte 3	93h (command)
Byte 4	Type of setting; 01h = current, 02h = power, 03h = resistance
Byte 5	Total number of program steps (1-10)
Byte 6	Low byte of step 1 setting
Byte 7	High byte of step 1 setting
Byte 8	Low byte of step 1 duration (seconds)
Byte 9	High byte of step 1 duration
Byte 10	Low byte of step 2 setting
Byte 11	High byte of step 2 setting
Byte 12	Low byte of step 2 duration
Byte 13	High byte of step 2 duration
Byte 14	Low byte of step 3 setting
Byte 15	High byte of step 3 setting
Byte 16	Low byte of step 3 duration
Byte 17	High byte of step 3 duration
Byte 18	Low byte of step 4 setting
Byte 19	High byte of step 4 setting
Byte 20	Low byte of step 4 duration
Byte 21	High byte of step 4 duration
Byte 22	Low byte of step 5 setting
Byte 23	High byte of step 5 setting
Byte 24	Low byte of step 5 duration
Byte 25	High byte of step 5 duration
Byte 26	Checksum

94h, Define programmed test sequence, steps 6-10

Byte 1	AAh
Byte 2	Address (00h—FEh)
Byte 3	94h (command)
Byte 4	Low byte of step 6 setting
Byte 5	High byte of step 6 setting
Byte 6	Low byte of step 6 duration
Byte 7	High byte of step 6 duration
Byte 8	Low byte of step 7 setting
Byte 9	High byte of step 7 setting
Byte 10	Low byte of step 7 duration
Byte 11	High byte of step 7 duration
Byte 12	Low byte of step 8 setting
Byte 13	High byte of step 8 setting
Byte 14	Low byte of step 8 duration
Byte 15	High byte of step 8 duration
Byte 16	Low byte of step 9 setting
Byte 17	High byte of step 9 setting
Byte 18	Low byte of step 9 duration
Byte 19	High byte of step 9 duration
Byte 20	Low byte of step 10 setting
Byte 21	High byte of step 10 setting
Byte 22	Low byte of step 10 duration
Byte 23	High byte of step 10 duration
Byte 24	Program mode (00h = run once; 01h = repeat)
Byte 25	System Reserved
Byte 26	Checksum

95h, Start programmed test sequence

Byte 1	AAh
Byte 2	Address (00h-FEh)
Byte 3	95h (command)
Byte 4 - 25	System Reserved
Byte 26	Checksum

96h, Stop programmed test sequence

Byte 1	AAH
Byte 2	Address (00h-FEh)
Byte 3	96h (command)
Byte 4 - 25	System Reserved
Byte 26	Checksum