DAC unit «DAC-40-USB»

Manual

1. Purpose

Digital-to-analog converter unit «DAC-40-USB» is intended to provide multi-channel voltage output. It is controlled from a personal computer via a USB port. Its primary purpose is to drive deformable mirrors produced by OKO Technologies.

2. Specifications

	V 1.0	**V 2.0**
Analog outputs	40	40
Output range	0 – 5.5 V	0 – 5.0 V
Output resolution	12 bits (4096 levels)	
Range of adjustment of the maximum output voltage	2.5 – 5.5 V	2.6 – 5.0 V
Ohmic load, each channel	≥100 kΩ	≥100 kΩ
Load capacitance, each channel	≤500 pF	≤500 pF

Synchronous output for all channels.
Power is provided via the USB port.

3. General design

«DAC-40-USB» is designed as a PCB with two double-row angle connectors BH-20R (male) and a B-type USB connector; it is mounted in a compact housing. Pins of the output connectors are labeled according to the

numbering order of the output channels. The maximum output voltage value can be adjusted (all channels simultaneously) by a variable resistor, whose slot is sunk in a hole on the front side of the unit (Fig. 2).

To provide access to the PCB (Fig.1, 2) one should disconnect upper and lower decks of the housing by simultaneous depression of their side surfaces. To close the housing, the decks must be pushed vertically one to another till latched.

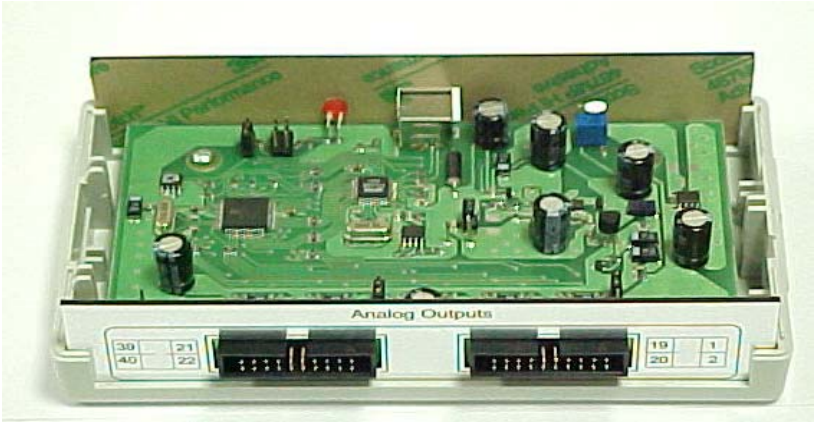

Fig. 1. «DAC-40-USB» view from the output connectors' side

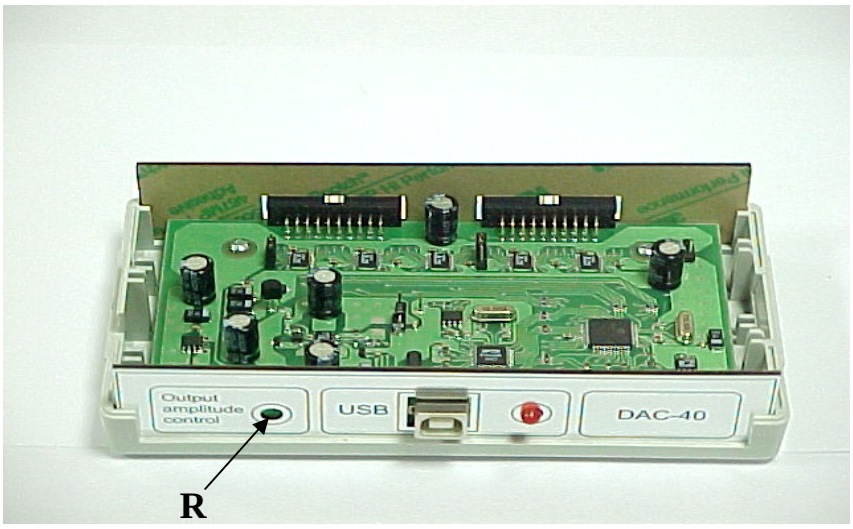

Fig. 2. «DAC-40-USB» view from the USB connector side

3.1. Jumper settings for version 1.0

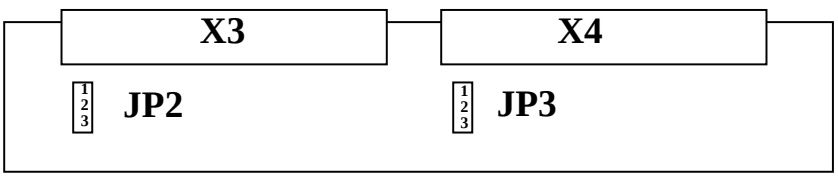

Fig. 3. PCB layout and numbering of jumpers' contacts for V.1.0

Jumpers JP2 and JP3 are mounted on the PCB (Fig. 3). By connecting contacts 2 and 3 of jumpers JP2 and JP3 pin #1 of connector X3 (channel 1) and/or pin #1 of X4 (channel 21), correspondingly, can be connected to the ground (see Table 1). It should be kept in mind that when pin #1 of X3 (pin #1 of X4) is connected to the ground, the 1st (or 21st, correspondingly) channel of the DAC is disconnected from the connector.

Table 1

Jumper settings and output connectors commutation

JP2	Pin 1 of X3		JP3	Pin 1 of X4
1–2	Channel #1 output		1–2	Channel #21 output
2–3	Ground		2–3	Ground

3.2. Jumper settings for version 2.0

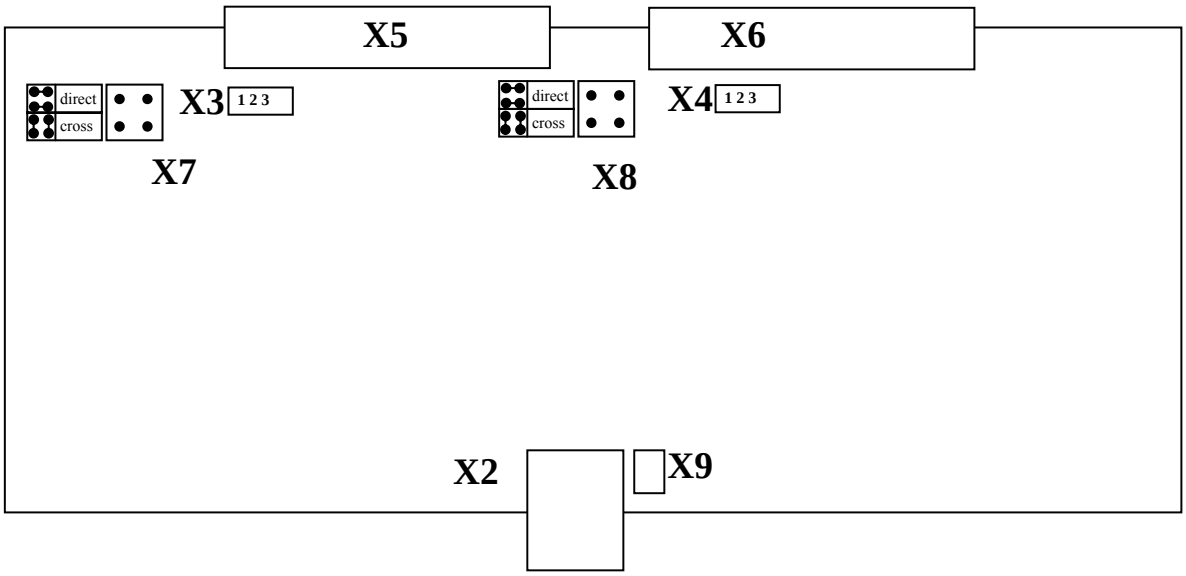

Fig. 4. PCB layout and numbering of jumpers' contacts for V.2.0

Jumpers X3 and X4 are mounted on the PCB (Fig. 4). By connecting contacts 2 and 3 of jumpers X3 and X4 pin #1 of connector X5 (channel 1) and/or pin #1 of X6 (channel 21), correspondingly, can be connected to the ground (see Table 2). It should be kept in mind that when pin #1 of X3 (pin #1 of X4) is connected to the ground, the 1st (or 21st, correspondingly) channel of the DAC is disconnected from the connector.

Jumpers X7 and X8 allow swapping pins 1 and 2 of the output connectors X5 and X6 (see Table 2 for details).

Jumper X9 can be set to connect the outer case of the USB connector X2 to ground.

Table 2

Jumper settings and output connectors commutation

X3	X7	Pin 1 of X5	Pin 2 of X5		X4	X8	Pin 1 of X6	Pin 2 of X6
1–2	direct	Ch #1 output	Ch #2 output		1–2	direct	Ch #21 output	Ch #22 output
2–3		Ground	Ch #2 output		2–3		Ground	Ch #22 output
1–2	cross	Ch #2 output	Ch #1 output		1–2	cross	Ch #22 output	Ch #21 output
2–3		Ch #2 output	Ground		2–3		Ch #22 output	Ground

4. Getting started

Before using the unit, it is necessary to check settings of jumpers or appropriately configure those according to specifications of connected device(s) as described above.

To put the unit into operation following steps must be done.

- Plug connecting cables into the output connectors of the unit.
- Connect the unit to the USB port of your PC using the USB cable. The LED lit on the USB port face in 2-2.5 seconds indicates normal operation of the unit.
- Install FDTI Direct Driver for Windows 98/2000/ME/XP from the folder **/Drivers** of the supplied software CD. Updated drivers can be downloaded from www.ftdichip.com. For version 2.0 you have to use the driver version 2.00.00 or higher.
- Run **TEST_DAC40.EXE**, move the pointer of the output voltage into U_{max} position, then adjust the output voltage to the required value by rotation of the spindle of the variable resistor *R*. The level of the output voltage must be measured by a voltmeter connected to one of the output pins with respect to the grounded pin.

5. The program interface

5.1. General

Data transfer between PC and DAC via USB bus is managed by the interface chip from FT245 series. To provide necessary speed of data transfer between PC and DAC it is necessary to use FDTI Direct Driver. The corresponding program interface is implemented in the library **FTD2XX.DLL**.

5.2. Application of the library **FTD2XX.DLL**

To provide access to the DAC, the following functions of **FTD2XX.DLL** are implemented:

- **FT_ListDevices** – allows to detect the number of connected devices, their serial numbers and specifications;
- **FT_Open** – opens of the device for data transfer;
- **FT_Write** – send a packet of control data from the PC to the device;
- **FT_Close** – closes the device.

Specifications of these functions are given in the programmer's manual of the library **FTD2XX** (file **D2XXPG33.pdf** in the folder **/Driver** of the software CD). Examples of their use are given in Section 5.4.

5.3. Data interchange via USB

Control of the unit from an external program is organized by sending data packets to the module. The data packet has 129 bytes size. An example of forming of the data packet from an array of 16-bit values corresponding to output voltages of the DAC unit is given in Appendix 1, function **MakePacket**.

When the data package is received and control words loaded into DAC chips, synchronous setting of signal levels at the outputs takes place, and the micro-controller enters the standby mode, waiting for the next data package. In this mode, zero bytes transferred into the module via the USB bus are ignored. In order to put the unit into the standby mode, it is necessary to fulfill transfer of data package consisting of 128 zero bytes. It is recommended to perform this operation when initializing the unit's control cycles.

5.4. **MakePacket** function usage

To facilitate programming of the unit, one may use the function **MakePacket**. It transforms an array consisting of 40 two-byte words, each of them coding an output voltage at the corresponding channel, into the data packet, which can be transferred via USB. Text of the function and an example of its usage are given in Appendix 1.

When connecting two or more DACs «DAC-40-USB» to the computer it is necessary to define their system numbers by means of **FT_ListDevices** function. A detailed description of this function is given in file **D2XXPG33.pdf**, folder **/Driver** in the software CD. An example of usage of function **FT_ListDevices** is given in Appendix 2.

For synchronous control of the units one may use the function **MakePacket**. It is implied that before this all devices «DAC-40-USB» are opened by means of the function **FT_Open**, with preliminary defined system numbers of these modules.

7. Calibration of the unit

Using the DAC unit it is necessary to keep in mind that the minimum voltage generated by any output channel corresponds, in general case, to non-zero control code. It is due to specifics of operation of the DAC chip. User can set the minimum value of this parameter for each channel by adjustment of the maximum output voltage $U_{\max}$. To do this, run **TEST_DAC40.EXE** in «Custom» mode and connect a digital voltmeter with accuracy not worse than 0.5 mV to the output of the channel to be tested. Detect the minimum control code, whose unitary increment changes the output voltage to approximately $U_{\max} / 4095$.

Appendix 1.

```
#include "ftd2xx.h"
void    MakePacket(WORD *buf,BYTE *packet);// Function of data package for DAC formation
void dac_setup(void)
// Sample:   To set voltages with level codes 0,100,200, … 4000
//           at the DAC outputs  1,2,3, …, 40

{
WORD   buf[40];                 // Buffer of DAC channels
BYTE   packet[130];             // DAC data package
FT_HANDLE  DAC;                 // DAC descriptor

FT_STATUS fs = FT_Open(0,&DAC);  // Open a device with system # 0(1,2,etc)
if(fs==FT_OK)                    // If the device is opened successfully
       {
       unsigned long BR;
       for(int i=0;i<40;i++)     // Fill up the buffer of channels with codes of
              buf[i]=i*100;             // voltage levels
       MakePacket(buf,packet);       // Transform buffer of the channels
                                     // into data package
       FT_Write(DAC,packet,130,&BR); // Transfer data package into the DAC
       FT_Close(DAC);               // Close the device
       }
else                                {// Error handling … }
} // End dac_setup()

//---------------------------------------------------------------------------
static BYTE DAC_CHANEL_TABLE[40]= // Table of DAC channels
    {
/*DAC-> 0   1   2   3   4    |
        --------------------+  OUTPUT    */
```

```
        7, 15, 23, 31, 39, //|  A
        6, 14, 22, 30, 38, //|  B
        5, 13, 21, 29, 37, //|  C
        4, 12, 20, 28, 36, //|  D
        3, 11, 19, 27, 35, //|  E
        2, 10, 18, 26, 34, //|  F
        1,  9, 17, 25, 33, //|  G
        0,  8, 16, 24, 32  //|  H
    };
//---------------------------------------------------------------------------
void MakePacket(WORD *buf,BYTE *packet)
/*          Form a data packet from the buffer of channels:
   buf -    an input array consisting of forty 16-digit words, which code
            voltage levels of the outputs ##1-40
  packet – the resulting 129-byte output array to be transferred into the unit via
            USB bus
*/
{
 BYTE *p=packet+1;
 for(int i=0,s=0;i<8;i++,s+=5)
    {
       // Form address parts of control words for five DAC chips
    *(p++)=0;
    *(p++)=(i&4)?0x1f:0;
    *(p++)=(i&2)?0x1f:0;
    *(p++)=(i&1)?0x1f:0;

        // form control codes from the array of voltages according to the table
    for(int j=0,mask=0x800;j<12;j++,mask>>=1)
        *(p++)=
            ((buf[DAC_CHANEL_TABLE[s+0]]&mask)?0x01:0) |
            ((buf[DAC_CHANEL_TABLE[s+1]]&mask)?0x02:0) |
            ((buf[DAC_CHANEL_TABLE[s+2]]&mask)?0x04:0) |
            ((buf[DAC_CHANEL_TABLE[s+3]]&mask)?0x08:0) |
            ((buf[DAC_CHANEL_TABLE[s+4]]&mask)?0x10:0) ;
    }
  packet[0]   = 0xff; // non-zero starting byte
}
```

Appendix 2.

Sample code shows how to get the number of devices currently connected

```
FT_STATUS ftStatus;
DWORD numDevs;

ftStatus = FT_ListDevices(&numDevs,NULL,FT_LIST_NUMBER_ONLY);
if (ftStatus == FT_OK) {
    // FT_ListDevices OK, number of devices connected is in numDevs
}
else {
    // FT_ListDevices failed
}
```

This sample shows how to get the serial number of the first device found. Note that indexes are zero-based. If more than one device is connected, incrementing devIndex will get the serial number of each connected device in turn.

```
FT_STATUS ftStatus;
DWORD devIndex = 0;
char Buffer[16];

ftStatus =
FT_ListDevices((PVOID)devIndex,Buffer,FT_LIST_BY_INDEX|FT_OPEN_BY_SERIAL_NUMBER);
if (FT_SUCCESS(ftStatus)) {
```

```
    // FT_ListDevices OK, serial number is in Buffer
}
else {
    // FT_ListDevices failed
}
```

This sample shows how to get the product descriptions of all the devices currently connected.

```
FT_STATUS ftStatus;
char *BufPtrs[3];         // pointer to array of 3 pointers
char Buffer1[64];         // buffer for the product description of first device found
char Buffer2[64];         // buffer for the product description of second device
DWORD numDevs;            // initialize the array of pointers

BufPtrs[0] = Buffer1;
BufPtrs[1] = Buffer2;
BufPtrs[2] = NULL;        // last entry should be NULL

ftStatus = FT_ListDevices(BufPtrs,&numDevs,FT_LIST_ALL|FT_OPEN_BY_DESCRIPTION);
if (FT_SUCCESS(ftStatus)) {
       // FT_ListDevices OK, product descriptions are in Buffer1 and Buffer2, and
       // numDevs contains the number of devices connected
}
else {
       // FT_ListDevices failed
}
```

8. Using with 37-channel micromachined membrane deformable mirror

«DAC-40-USB» can be used to drive a 37-channed micromachined membrane deformable mirror (MMDM) produced by OKO Technologies. Two amplifier boards and four cables from OKO Technology are required for this purpose. Assembling of the system is described below.

- Disconnect «DAC-40-USB» unit from your computer. In order to provide ground to the mirror and connect it to the ground of the amplifier boards, jumpers X3 and X4 of the unit should be set to the position 2-3. **If the mirror is supplied with a full set of control electronics, the jumpers should be already properly configured.**
- Connect the amplifier boards (or amplifier units) to «DAC-40-USB» unit using 20 pins-to-26 pins cables, observing the order of numbering, e.g., the amplifier board number 1 should be connected to X5 output connector, whereas the board number 2 should be connected to X6 connector.
- Supply power to the amplifier boards (units) as specified in the mirror manual.
- Connect «DAC-40-USB» to the computer using a USB cable. Turn on the low voltage power supply, then the high voltage one.
- The folder **/mmdm37ch_vc** of the software CD contains simple command-line utilities for control of the mirror, which are compiled with Visual C++ 6.0. Enter this folder from the command line and type "**am_set 4095**"; it will set the maximum voltage level at the output of the unit.
- Measuring the voltage between the ground pin (pin 20) and any control pin at the output of any amplifier board, adjust it by turning the variable resistor ***R*** until the maximum supply voltage for this mirror is achieved. **If the mirror is supplied with a full set of control electronics, the resistor *R* is already adjusted.**
- Turn off the high voltage power supply. Connect the mirror to the amplifier boards using 20 pins-to-20 pins cables, also observing the numbering. Fix the cables to the optical table.
- Turn on the high voltage power supply. Now you can use the mirror.

The following utilities are provided to drive the mirror.

- **"rotate"** sets the maximum value (4095) to all channels of the mirror, one by one.
- "**am_set *N***" sets the same voltage ***N*** to all actuators, where ***N*** is in the range 0…4095.
- "**19_set**" sets 4095 to 20-37 channels of OKO 37ch mirror and 0 to other channels.
- "**set_channel *N***" sets 4095 to ***N***-th channel and 0 to other channels.
- **"smiley37"** generate a smiley-like shape on the mirror.

If you need to recompile the utilities with Visual C++, copy the whole folder **/mmdm37ch_vc**, enter it in the command line and type **“nmake”**.

9. Using with other deformable mirrors

The CD suipplied with «DAC-40-USB» also contains sample control programs for other deformable mirrors from OKO Technologies. These programs include “**am_set**”, “**rotate**” and “**set_channel**” utilities similar to those described in Section 8. These are written in Visual C++ and accompanied by a make file for recompilation. The samples can be found in the following directories.

mmdm79ch_40mm_vc - 79-channel 40 mm MMDM connected to two USB units;

mmdm_lin19ch_vc – 19-channel linear MMDM connected to one USB unit;

pdm109ch_50mm_vc - 109-channel 50 mm PDM connected to three USB units;

piezo19ch_vc – 19-channel 30 mm PDM connected to one USB unit;

piezo37ch_50mm_vc – 37-channel 50 mm PDM connected to one USB unit;

piezo37ch_2005_vc – 37-channel 30 mm PDM (design of 2005) connected to one USB unit;

piezo37ch_vc – 37-channel 30 mm PDM (old design) connected to one USB unit;

piezo_lin20ch_vc – 20-channel liner PDM connected to one USB unit.

For hardware connections, please refer to the manual of the deformable mirror.