USB 96 channel digital output card

Product Datasheet 24

Features

- USB, hot pluggable, 96 channel digital output card
- Pin compatible with NIDAQ DIO96, PXI-6508, PCI-6509, PCI-DIO96 & DAQPAD6508 digital I/O cards
- Example code downloads available for: LabView, VB, VC, C#, JAVA, Agilent VEE & Delphi. Uses simple ASCII/Hex text command strings
- OS compatibility: Win98SE/ME/2K/XP/Vista/7, Windows CE, Windows 10, Mac OSX and Linux
- Output channels use 74HCT type devices. These are specified as: VOL = 0.4V, VOH= 3.7V, IOUT= 4mA.
- Low cost, small profile, stackable
- 2 x 50-way header access to each of the 96 digital output channels
- Designed to be compatible with our DIO96Adapter card, and our range of stacked or shelf mounted, 24 channel relay cards, or can be used stand alone.
- Fully compatible with our 96 channel relay cards in either stack, 2U or 3U shelf formats
- Onboard LED indicator shows power on function
- Supplied with nylon feet. Clear Perspex cover & base available.
- CE & RoHS compliant

Description

Low cost, general purpose, 96 channel USB digital output card.

Pin compatible with our DIO96Adapter and our range of 96 channel relay stacks, 2U (up to 96 channels) and 3U (up to 192 channel) shelf options. It is also pin compatible with the NIDAQ DIO96, PXI-6508, PCI-6509, PCI-DIO96 & DAQPAD6508 cards. Hot pluggable and powered from the USB port.

LabView, C & Visual Basic example programs available which demonstrate the functionality of the card.

Specifications

USB Interface
USB 1 & 2 compatible (virtual COM port)

Digital output signals
Output high, 3.7V DC (Typ.), output low 0.4V (max), 4mA typ. per output channel.

Operating temp range
0-70^{0}C

Power
5V DC @ 50mA (max), powered from the USB port. **Dimensions**
Dimensions 100mm (D) 127mm (W) 25mm (H) (exc feet), Weight 50g.

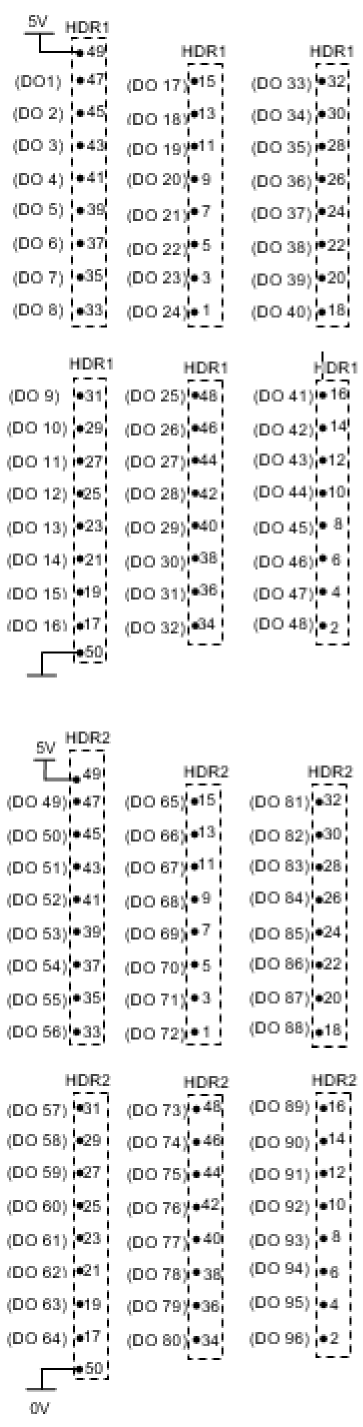
Order code

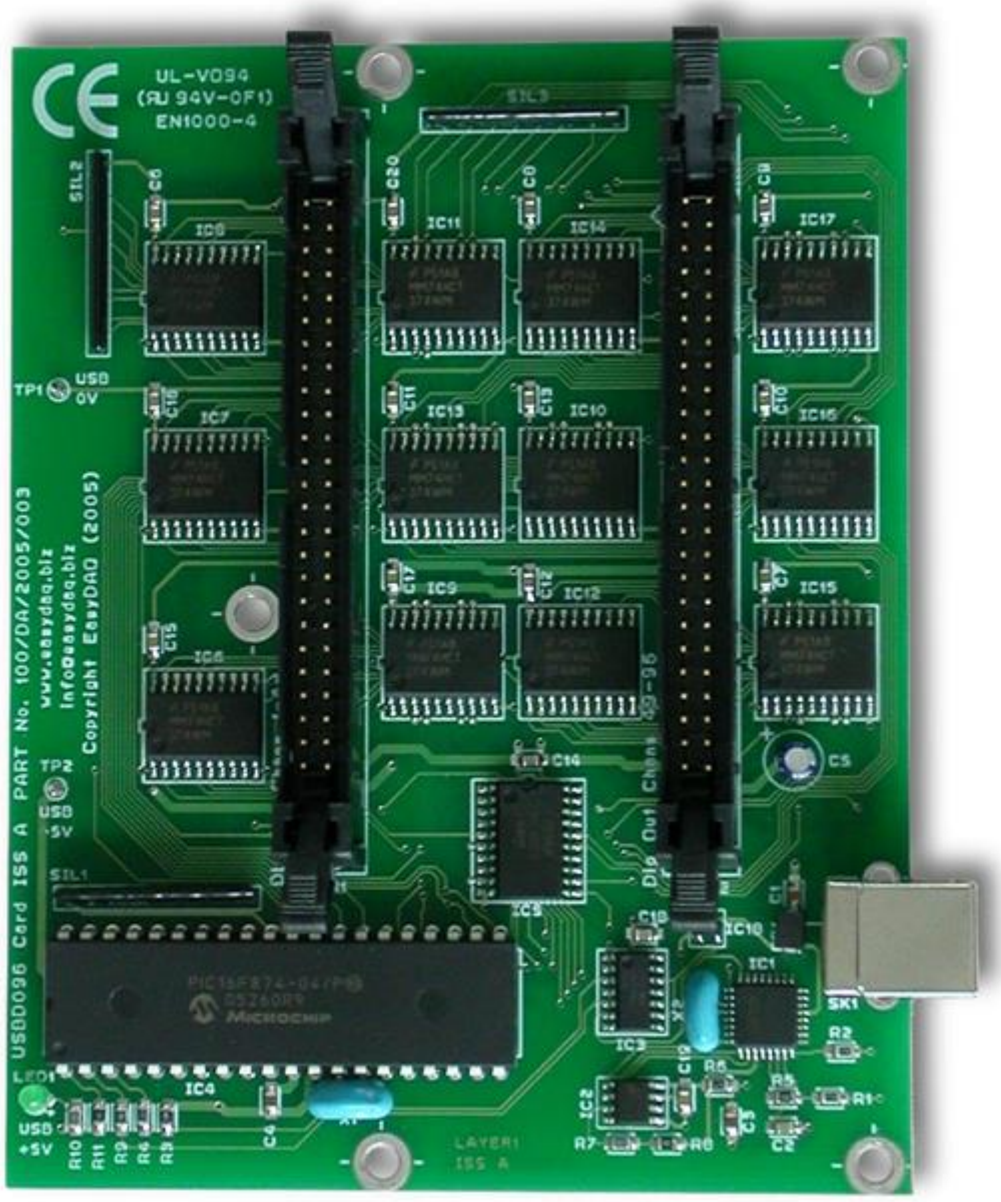
USBDO96
96 channel, USB compatible digital output card. Pin compatible with the NIDAQ DIO96 card and with our range of 24 channel relay cards & 96 channel relay stack options.
50-way (male) header connector access to all digital output signals. (See page 2 for header pin connections).

Product Datasheet 24

Card connectivity

50 Way Header connector pin connections (HDR1 & HDR2)





Configuring & programming the USBDO96 card:

General overview:

This section gives a general overview of how the card works and the steps that are required to configure and command it. (For details on the text commands that are required to program the USBDO96 card, see the 'Command format' section on pages 4 & 5).

Please refer to the following 'Quick start guide' for information on downloading & installing the USB 'Virtual COM port' drivers for the USBDO96 card. Click on the following link or paste into your browser:

Data Sheet 32 (USB DAQ & Relay Card devices - Installation & Quickstart Guide)

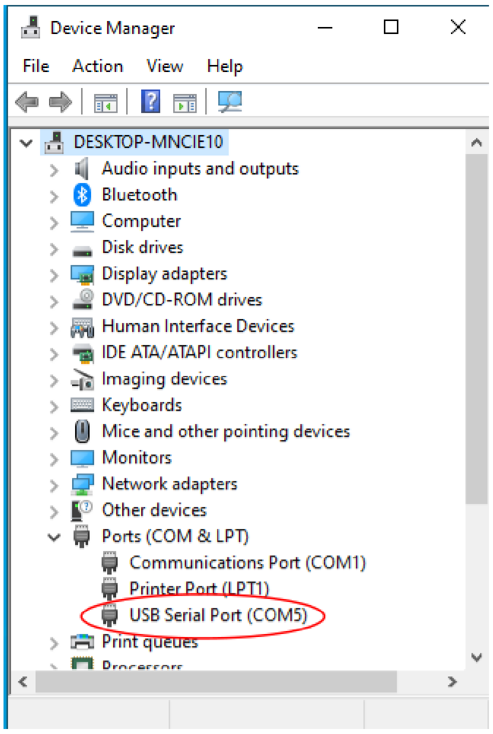
Example programs (applications and library files) are available in a range of programming languages and are designed as 'basic building blocks' to allow you to get up and running quickly with the USBDO96 card: Code Snippets

This card uses a simple ASCII/Hex text command set (compatible with our entire range of USB/serial relay cards), however the requirements for initialising and activating/de-activating individual relays are different and require further explanation.

Using a terminal emulator

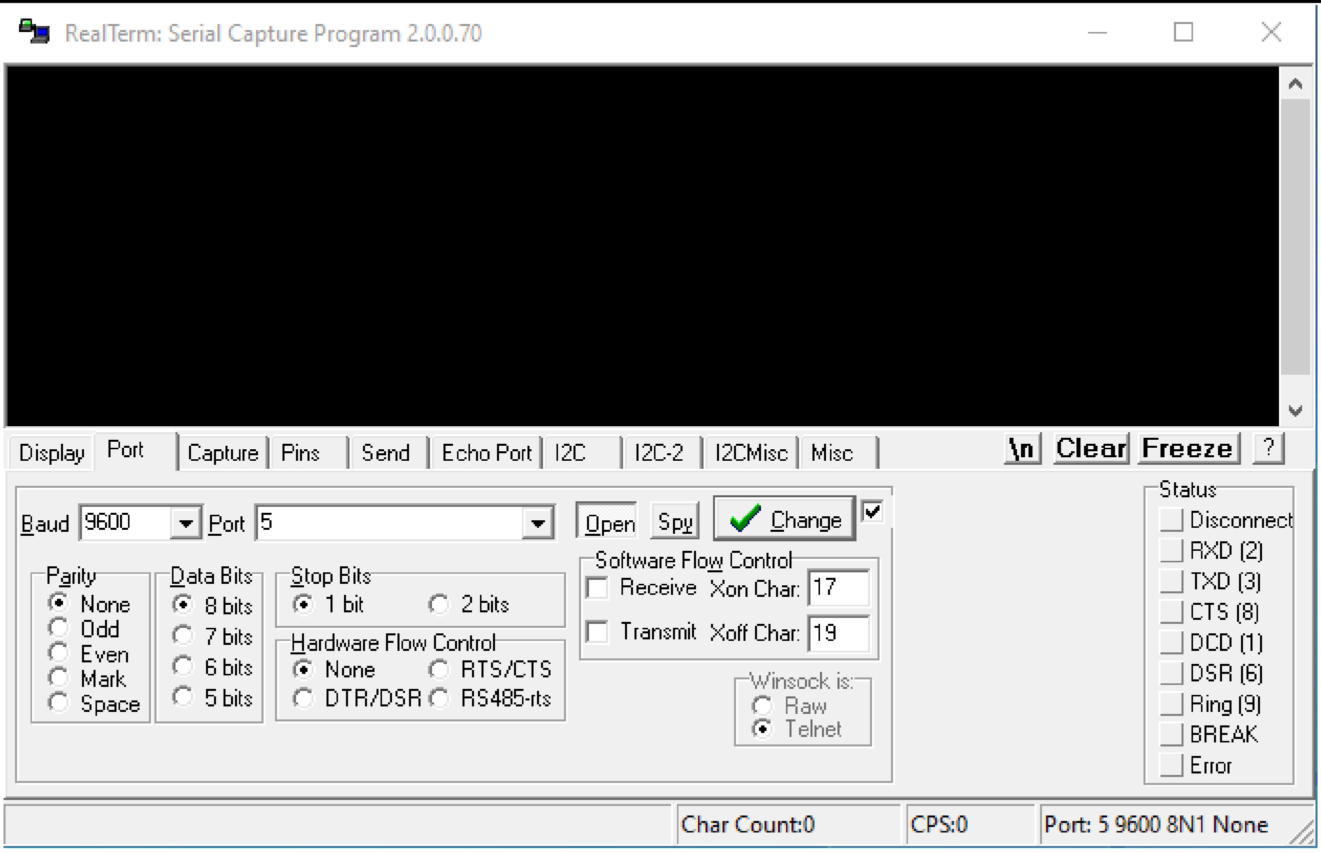
In order to test operation, the card can be connected (using the 'virtual COM'/serial port and controlled from a terminal emulator such as RealTerm. Ensure port configuration is set as shown above, type (ASCII/Hex) characters shown above to achieve port direction and read or write command/data.

In Windows use "Device manager" to find the port number that the USBDO96 has been assigned:



In this case Com Port 5 (COM5) has been assigned.

Use the assigned Com port and the serial port settings to set up RealTerm:



Baud 9600 Port 5

Parity: [✓ None] Odd Even Mark Space

Data Bits: [✓ 8 bits] 7 bits 6 bits 5 bits

Stop Bits: [✓ 1 bit] 2 bits

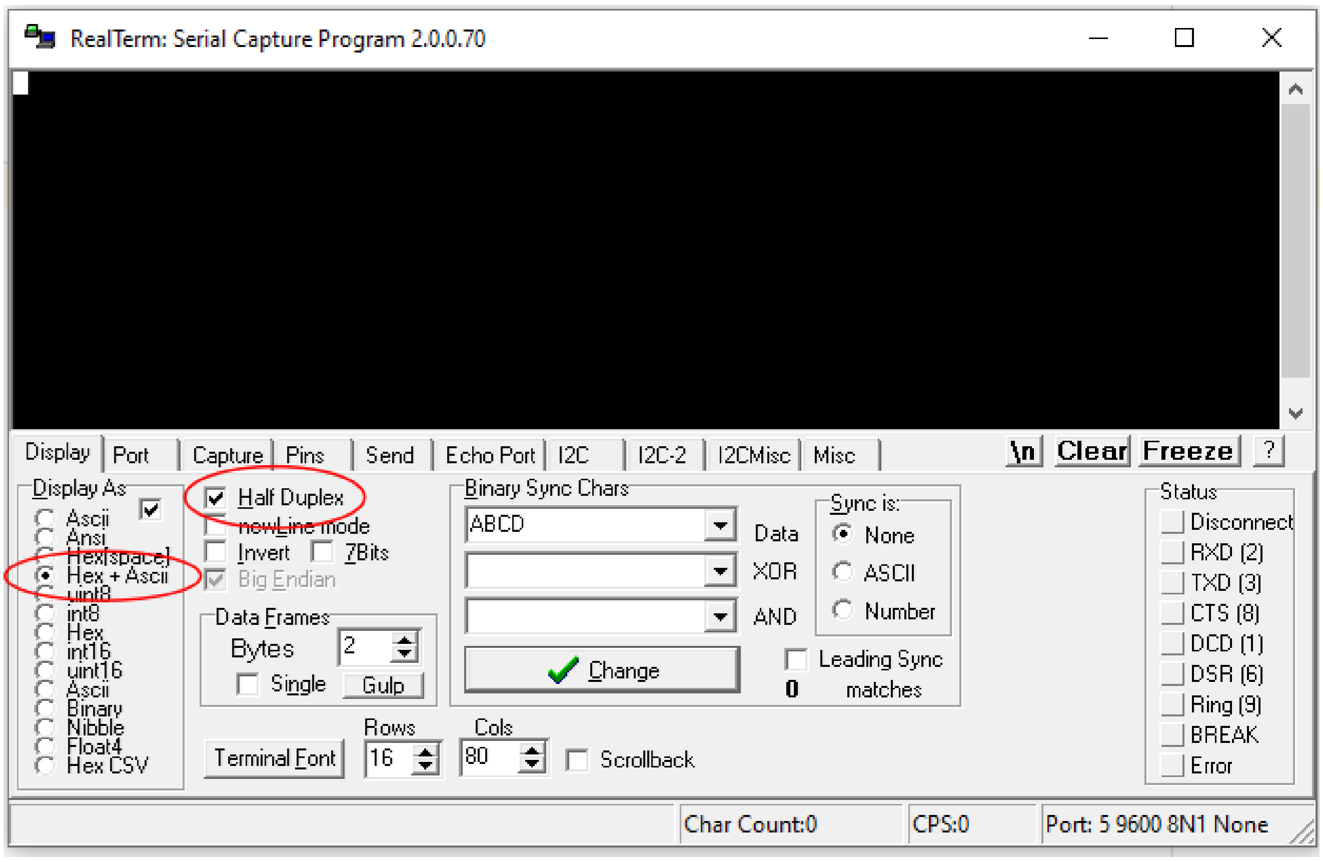
Hardware Flow Control: [✓ None] RTS/CTS DTR/DSR RS485-rts

Software Flow Control: Receive Xon Char: 17 Transmit Xoff Char: 19

Winsock is: Raw [✓ Telnet]

Char Count:0 CPS:0 Port: 5 9600 8N1 None

Configuring RealTerm to display "Hex+Ascii" and "Half Duplex" will show the sent information on the terminal display.



Display As: Ascii Ansi Hex[space] [✓ Hex + Ascii] uint8 int8 Hex int16 uint16 Ascii Binary Nibble Float4 Hex CSV

[✓ Half Duplex] newLine mode Invert 7Bits Big Endian

Data Frames: Bytes 2 Single

Binary Sync Chars: ABCD Data XOR AND

Sync is: [✓ None] ASCII Number

Leading Sync 0 matches

Rows 16 Cols 80 Scrollback

Char Count:0 CPS:0 Port: 5 9600 8N1 None

In general, this card is initialised and programmed as six individual groups, each group consisting of 16 relay relays, making 96 relays in total (6x16). Each group can be programmed in sub groups of 8.

This card uses our DO24 PIC chip – it has four 8-bit ports:

Port A (not programmable – it is used for communications functions)

Port B: Bits B0 to B7 (this is used for initialisation and data latching functions)

Port C: Bit C0 to C7 (this is used for the lower 8 bits of each 16-bit relay group)

Port D: Bit D0 to D7 (this is used for the upper 8 bits of each 16-bit relay group)

Port B functions:

Bit B0: This bit is used to control the output enable the 6 off 16-bit latches. It is gated with anti 'power on glitch' circuitry, designed to ensure that the card can always be initialised and configured prior to enabling each group of 16 (this avoids a potential random power on status – provided each group of 16 has been correctly initialised as per our example programs). Bits B1 to B7, C0 to C7 and D0 to D7 should all be set to logic 0 before this bit (B0) is set to logic 1 (i.e., enable). This will ensure that all digital output channels/relays are set to logic 0 after initialisation.

Bit B1: Data Latch signal for digital outputs DO1 to DO16 (Group 1). (Data Bits C0 to C7 and D0 to D7 will be latched by this signal changing from logic 0 to logic 1). Bit B0 must also be set to logic 1 (enable).

Bit B2: Data Latch signal for digital outputs DO17 to DO32 (Group 2). (Data Bits C0 to C7 and D0 to D7 will be latched by this signal changing from logic 0 to logic 1). Bit B0 must also be set to logic 1 (enable).

Bit B3: Data Latch signal for digital outputs DO32 to DO48 (Group 3). (Data Bits C0 to C7 and D0 to D7 will be latched by this signal changing from logic 0 to logic 1). Bit B0 must also be set to logic 1 (enable).

Bit B4: Data Latch signal for digital outputs DO49 to DO64 (Group 4). (Data Bits C0 to C7 and D0 to D7 will be latched by this signal changing from logic 0 to logic 1). Bit B0 must also be set to logic 1 (enable).

Bit B5: Data Latch signal for digital outputs DO65 to DO80 (Group 5). (Data Bits C0 to C7 and D0 to D7 will be latched by this signal changing from logic 0 to logic 1). Bit B0 must also be set to logic 1 (enable).

Bit B6: Data Latch signal for digital outputs DO81 to DO96 (Group 6). (Data Bits C0 to C7 and D0 to D7 will be latched by this signal changing from logic 0 to logic 1). Bit B0 must also be set to logic 1 (enable).

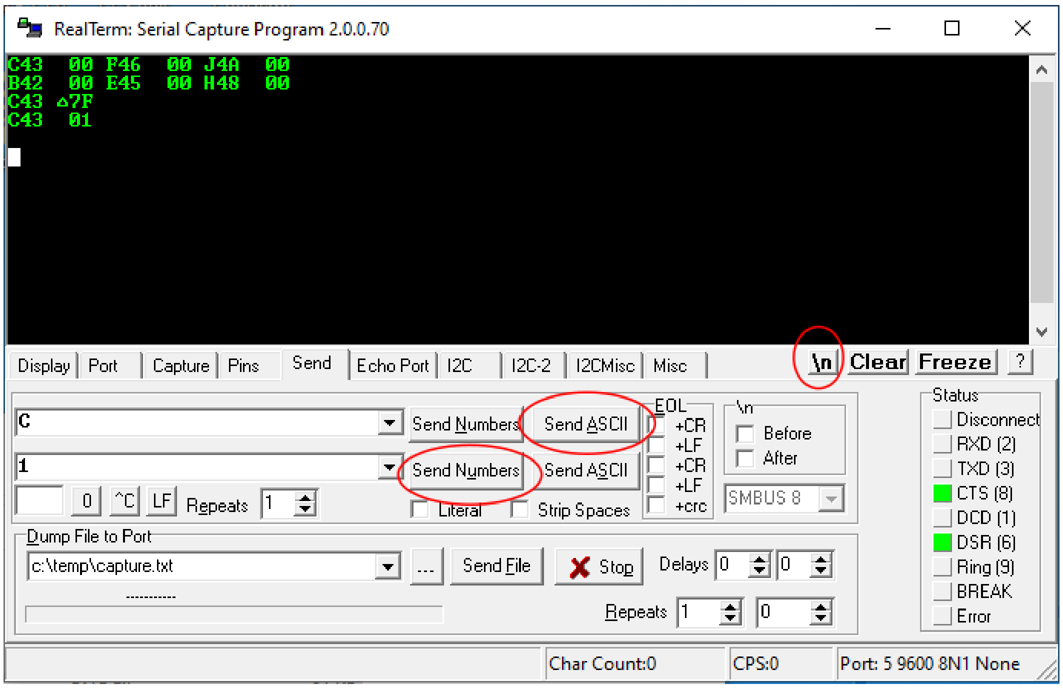
Note: Each group of 16 channels must be programmed individually (six groups of 16), they cannot be programmed simultaneously (i.e., 96 channels at the same time). However, it is possible to configure all 96 channels in a few 10's of milliseconds.

After setting each port B bit (B1 to B7) to a logic 1 (i.e., latching the status of each group of 16), each bit is then set to a logic 0. The data latching function only occurs on a positive going transition of each bit (i.e., logic 0 to logic 1).

Example 1: A typical initialisation sequence:

ASCII 'C' (43H), 00H ASCII 'F' (46H), 00H ASCII 'J' (4AH), 00H	On power up, ports B, C and D are set to input. This set of commands initialises Ports B. C & D at 00H (i.e., no channels/relays activated)
ASCII 'B' (42H), 00H ASCII 'E' (45H), 00H	Sets direction of ports B, C & D as outputs

ASCII 'H' (48H), 00H	
ASCII 'C' (43H), 7FH	This command writes 00H to all the latches simultaneously and enables all latch outputs
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition


RealTerm: Serial Capture Program 2.0.0.70

```
C43  00 F46  00 J4A  00
B42  00 E45  00 H48  00
C43 ▵7F
C43  01
```

C
1

Dump File to Port: c:\temp\capture.txt

Status: 🟩 CTS (8), 🟩 DSR (6)

Char Count:0 | CPS:0 | Port: 5 9600 8N1 None

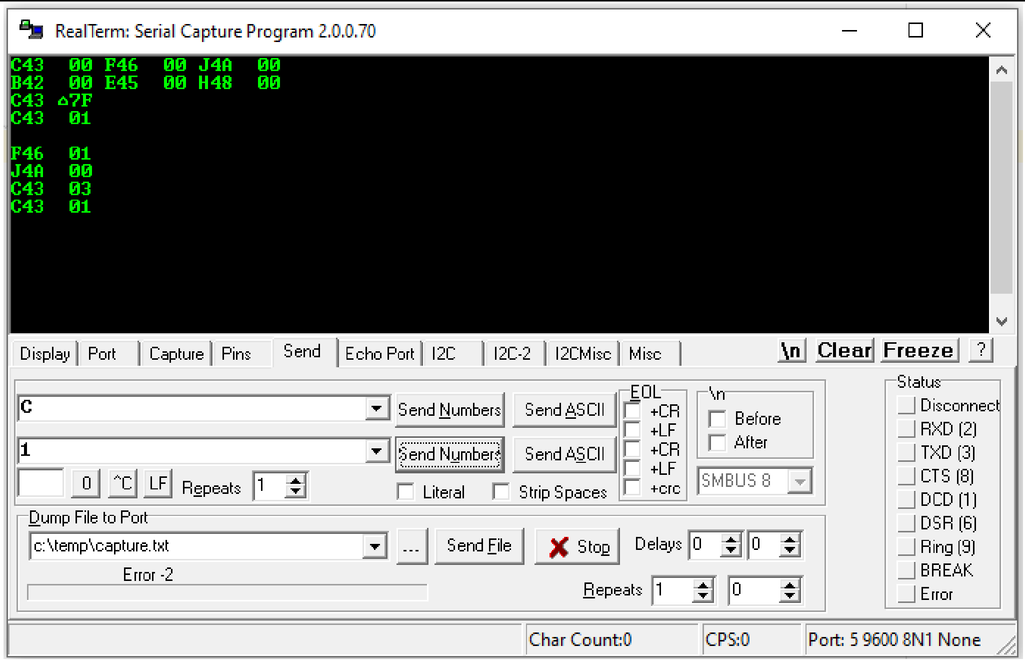
Note:

- Using the "/n" button on the send tab of RealTerm gives a new line and allows easier display of the sent commands.
- RealTerm allows sending of Ascii or numerical values.

A simple command sequence: Turn on relay 1. Relay 1 is bit 0 of latch bank 1.

ASCII 'F' (46H), 01H ASCII 'J' (4AH), 00H	These commands set bit 0 of port C and ensures that all bits of port D are clear.
ASCII 'C' (43H), 03H	This command writes 03H to Port B. This writes the value of ports C and D to latch bank 1
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

The next screenshot shows the additional sequence to turn on Relay 1:


RealTerm: Serial Capture Program 2.0.0.70

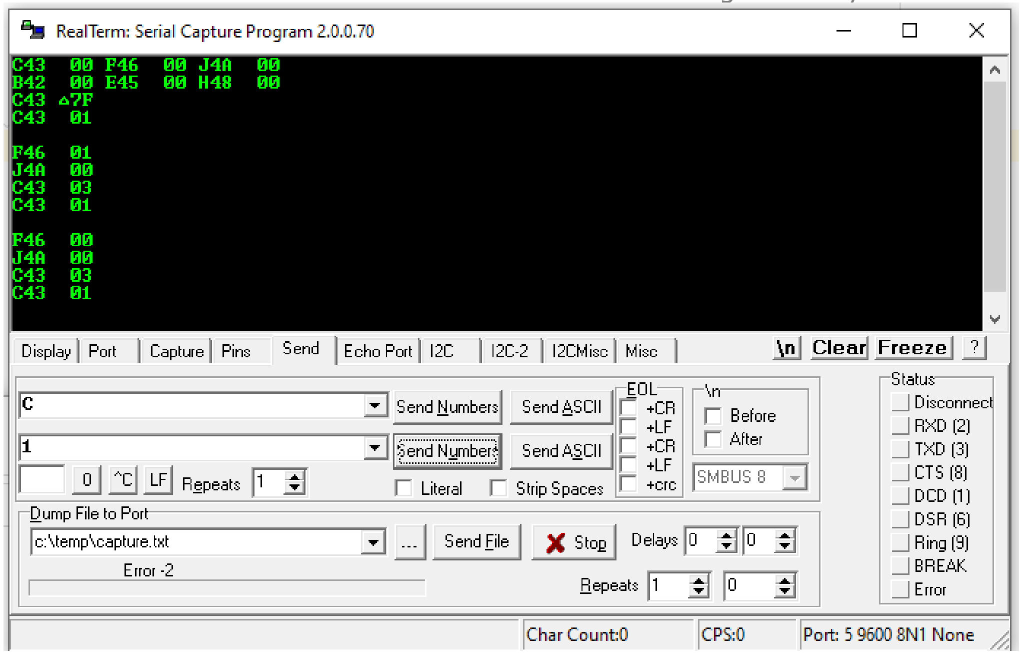
```
C43  00 F46  00 J4A  00
B42  00 E45  00 H48  00
C43 ⌂7F
C43  01

F46  01
J4A  00
C43  03
C43  01
```

C
1
Repeats 1
Dump File to Port: c:\temp\capture.txt
Error -2
Delays 0 0
Repeats 1 0
SMBUS 8
Char Count:0 CPS:0 Port: 5 9600 8N1 None

To turn off Relay 1…

ASCII 'F' (46H), 00H ASCII 'J' (4AH), 00H	This command clears bit 0 of port C
ASCII 'C' (43H), 03H	This command writes 03H to Port B. This writes the value of ports C and D to latch bank 1
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

The next screenshot shows more commands switching off Relay 1.


RealTerm: Serial Capture Program 2.0.0.70

```
C43  00 F46  00 J4A  00
B42  00 E45  00 H48  00
C43 ⌂7F
C43  01

F46  01
J4A  00
C43  03
C43  01

F46  00
J4A  00
C43  03
C43  01
```

C
1
Repeats 1
Dump File to Port: c:\temp\capture.txt
Error -2
Delays 0 0
Repeats 1 0
SMBUS 8
Char Count:0 CPS:0 Port: 5 9600 8N1 None

Example 2: Set Group1 all high, Group 2-6 all low. Assuming that the card has been initilised:

ASCII 'F' (46H), FFH ASCII 'J' (4AH), FFH	These commands set all bits of port C and D.
ASCII 'C' (43H), 03H	This command writes 03H to Port B. This writes the value of ports C and D to latch bank 1

ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

Example 3: Group 2 all high, Group 1 and 3-6 all low and following on from above:

ASCII 'F' (46H), 00H ASCII 'J' (4AH), 00H	These commands ensures that all bits of port C and D are clear.
ASCII 'C' (43H), 7FH	This command writes 03H to Port B. This writes the value of ports C and D to all latch banks
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition
ASCII 'F' (46H), FFH ASCII 'J' (4AH), FFH	These commands ensures that all bits of port C and D are set.
ASCII 'C' (43H), 05H	This command writes 05H to Port B. This writes the value of ports C and D to latch bank 2
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

Example 4: Group 1 relay no. '8' high, all other Group1 relays low, Group 2-6 all low and following on from above:

ASCII 'F' (46H), 00H ASCII 'J' (4AH), 00H	These commands ensures that all bits of port C and D are clear.
ASCII 'C' (43H), 7FH	This command writes 7FH to Port B. This writes the value of ports C and D to all latch banks
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition
ASCII 'F' (46H), 80H ASCII 'J' (4AH), 00H	These commands ensures that only the top bit of port C is set and port D is clear.
ASCII 'C' (43H), 03H	This command writes 03H to Port B. This writes the value of ports C and D to latch bank 1
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

Example 5: Group 1 relay no. '11' high, all other Group1 relays low, Group 2-6 all low and following on from above:

ASCII 'F' (46H), 00H ASCII 'J' (4AH), 04H	These commands ensures that only Relay 11 is selected...
ASCII 'C' (43H), 03H	This command writes 03H to Port B. This writes the value of ports C and D to latch bank 1
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

Example 6: Group 5 relay no. '8' high, all other Group5 relays low, Group 1-4, and 6 all low and following on from above:

ASCII 'F' (46H), 00H ASCII 'J' (4AH), 00H	These commands ensures that all bits of port C and D are clear.
ASCII 'C' (43H), 7FH	This command writes 7FH to Port B. This writes the value of ports C and D to all latch banks
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

ASCII 'F' (46H), 80H ASCII 'J' (4AH), 00H	These commands ensures that only the top bit of port C is set and port D is clear.
ASCII 'C' (43H), 21H	This command writes 21H to Port B. This writes the value of ports C and D to latch bank 5
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

Example 6: Group 5 relay no. '1', and '2', and '6' high, all other Group5 relays low, Group 1-5, and 6 all low and following on from above:

ASCII 'F' (46H), 23H ASCII 'J' (4AH), 00H	These commands select relays 1, 2 and 6 of a bank.
ASCII 'C' (43H), 21H	This command writes 21H to Port B. This writes the value of ports C and D to latch bank 5
ASCII 'C' (43H), 01H	This command writes 01H to Port B. This returns the latch signals of all the latches to logic 0 - signals are only set on the logic 0 to logic 1 transition

A Python program demonstrating the above examples is available in Code snippets at:
https://www.easydaq.co.uk/downloads/Python/USBDO96 Examples.zip

A sequential test program written in Python is also available:
https://www.easydaq.co.uk/downloads/Python/USBDO96 Test.zip

Serial Port settings

Baud rate:	9600
Parity:	0
Data:	8 bits
Stop bits:	1
Handshaking:	None

Auto detection & com port assignment

When you connect this card to a USB port of your computer for the first time, it will be auto-detected and ask you to install drivers (downloadable from the 'downloads' section of our website). After installation, the card will appear as a 'virtual' COM port and be automatically assigned the next available COM port. Following installation, the COM port number can be manually re-assigned via the control panel if required. Following reboots or disconnects of the USB card, the same COM port number will be assigned to the unique serial number of this card.

Command format

The card is initialised & commanded by sending a single ASCII character followed by a hex number (representing the required port status). The commands address each 8-bit DIO port of the PIC device. (ASCII character Hex equivalent is shown in brackets). You must first set the port direction (as either input or output). If a channel is set as an input, your software must send a read command (of that channel) followed by a read of the serial port. You should allow approximately 10mS between successive commands.

Port B (Bits 0-7), C (Bits 8-15) & D (Bits 16-23) commands:

Note: The USBDO96 card is an output only card – it cannot be used in input/read mode.

ASCII 'B' (42H), 0H	Initialises the card (sets the port & channel I/O directions). Set direction of Port B, 1=Input, 0= output. (i.e., where 0H sets bits 0-7 as outputs).
ASCII 'C' (43H), X	Write data X to Port B (i.e., X=00000011 (03H), latches data from ports C and D onto the 16-bit latch of bank 1). Valid data bytes are latched by the card until a further valid data byte is written to it.

ASCII 'E' (45H), 0H	Initialises the card (sets the port & channel I/O directions). Set direction of Port C
ASCII 'F' (46H), X	Write data X to Port C (i.e., X=00000001 (01H), sets channel 1 to active).
ASCII 'H' (48H), 0H	Initialises the card (sets the port & channel I/O directions). Set direction of Port D
ASCII 'J' (4AH), X	Write data X to Port D (i.e., X=00000001 (01H), sets channel 1 to active).

Example downloads

Example driver files and executables are available from the 'downloads' or 'Code snippets' areas of our website (www.easydaq.co.uk). Example programs for our USB relay & DIO cards are currently available for Python, LabView, Visual Basic, Visual C, C#, JAVA, Agilent VEE & Delphi.

Uses existing USB & serial port software examples

This card uses the same software drivers & command interface that is used in our existing range of USB/Serial port relay & DIO card products. Therefore, if you have already used our USB or serial port products on a previous project, you can re-use some or all of your code (or switch to using a USB relay card instead of a serial port card).

If you are a Linux user, please refer to this web link for additional information and low-level details on how to address and command our USB/serial relay & DIO cards:

Data Sheet 34 (Using Linux with EasyDAQ USB Products)

External links

https://www.easydaq.co.uk/downloads/Python/USBDO96 Examples.zip
https://www.easydaq.co.uk/downloads/Python/USBDO96 Test.zip
https://www.easydaq.co.uk/downloads/USBDO96_Generic_Test_Module_LV7_1_LLB.zip

Product Datasheet 24

NOTE.
Information in this document is believed to be accurate and reliable. However, EasyDAQ does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

Document versions

Version number	Date	Notes
1.0	26th November 2005	First release
1.1	2nd April 2018	Updated to links to EasyDAQ.co.uk
1.2	17th November 2021	This version. Corrected "J" = ASCII 0x4A (Not "K"). Various typing errors corrected.
1.3	11th July 2022	Updated format and added lots of examples including use of RealTerm. Also links to Python test programs.