Consort
Hertenstraat 56/9
B-2300 Turnhout
Belgium
+32 (0)14 411279
+32 (0)14 429179
support@consort.be

EV2000 Digital Communication

Contents

Hertenstraat 56/9
B-2300 Turnhout
Belgium
+32 (0)14 411279
+32 (0)14 429179
support@consort.be

1 Introduction

The EV2000 power supply range is equipped with a USB serial port. This allows the device to communicate with a PC. The communication software has to use specific commands according to a tight protocol to ensure correct data transmission.

The goal of this note is to describe how to communicate with the power supplies.

2 Connection

The connection is physically made with a USB port for which a driver should be installed. This driver can be downloaded from this web page of FTDI: http://www.ftdichip.com/Drivers/VCP.htm. The easiest method to install the driver for Windows system is choosing the *setup executable*.

After the installation of the driver, connect the power supply to the USB port. The computer will connect to it and install a Virtual Comm Port (VCP) which allows to communicate digitally over a Serial Comm Port. This serial port will have a name with number such as ***COM3*** or higher. This port number will be assigned to the specific USB port entrance to which the power supply is connected. The communication settings of the serial port are: ***57600 baud, no parity, 8 data bits, 1 stop bit, no handshaking***.

3 Protocol

3.1 Conventions

The following information and codes are given with examples. To understand exactly which are codes are passed, the following conventions are being used:

- The examples are written in a coloured `fixed pitch font`.
- The `blue` coloured text in the example highlights the text sent by the computer (see Send a command).
- The `magenta` coloured text highlights the answer received from the device (see Receive an answer).
- `Green` text highlights additional comments or clarifications which are not sent or received.
- The examples are written with a hexadecimal notation which show best the binary differences in a short manner. More information on hexadecimal numbers can be found on Wikipedia.
- Groups of bytes can be underscored to show they represent one value.

3.2 Send a command

The communication protocol of the device is designed with the possibility to check the transferred information by use of a simple checksum and with the number of incoming bytes. Each set of characters is terminated with the combination CR (Carriage return, 0x0D) and LF (Linefeed, 0x0A).

Generally a command is sent to the device this way:

> '**V**' + number of bytes to follow excluding (CR+LF) + Command byte (+ data bytes) + Low byte checksum of previous bytes (starting from 'V') + CR + LF

e.g.

56 02 CD 25 0D 0A

0x56 = character V
0x02 = 2 bytes are following (without CR+LF)
0xCD = Command 205
0x25 = lowest byte of checksum of preceding bytes
0x0D = Carriage return
0x0A = Linefeed

3.3 Receive an answer

Similarly the device responds to the command requests as follows:

'**P**' + number of bytes to follow excluding (CR+LF) + Command byte to respond (+ information bytes) + Low byte checksum of previous bytes (starting from 'P') + CR + LF

e.g. Answer received upon command sent in previous example (Send a command)

50 02 CD 1F 0D 0A

0x50 = character P
0x02 = 2 bytes are following (without CR+LF)
0xCD = Command 205
0x25 = lowest byte of checksum of preceding bytes
0x0D = Carriage return
0x0A = Linefeed

4 General information about the programming

4.1 Operation

The power supply should be handled as if it were manually programmed, i.e. it listens to commands as if they are entered by hand through the keyboard. e.g. When the device starts up in stand-by mode, send the RUN key to start the immediately the running of the last used program for running a gel electrophoresis. Therefore it is important that the programmer should get familiar with the manual action of programming the device.

4.2 Types of commands

The commands the device is listening to can be divided into 3 groups.
Command that can be used:

1. at any time
2. only when in stand-by
3. only during a run (when controlling the output parameters)

4.3 Conversion of multi-byte values

Many values are entered or returned as multi-bytes. These new power supplies are making use of a microcontroller which uses the Little-Endian system. This system sends multi-byte values starting with the lowest byte first. More information can be found here: http://en.wikipedia.org/wiki/Endianness.

4.3.1 Two bytes

The parameters of the device are often exchanged in a 2 byte format, namely as a binary 2's complemented number. To convert a pair of bytes (low and high byte) to the value, simply take the ASCII value of the second byte (high byte), multiply it with 2^8 (=256) and add the ASCII value of the first byte (low byte).

e.g. ASCII value first (low) byte = 195
ASCII value second (high) byte = 2
Number = 195 + 2 * 256 = 707
The value of the 2's byte complemented binary number is 707.

To send a 2 byte value, the first byte should be *[number - 256 * (number/256)]*, the second byte should be *(number/256)*.

4.3.2 Four bytes

The parameters of the device are often exchanged in a 4 byte format, namely as a binary 2's complemented number. To convert these 4 bytes to the value, take the ASCII value of the first byte (low byte), multiply the second byte with 2^8 (=256), the third byte with 2^{16} (=65.536), the fourth byte with 2^{24} (16.777.216) and make the sum of the four results to obtain the final value.

e.g. ASCII value first (high) byte = 100
ASCII value second (low) byte = 19
ASCII value second (low) byte = 1
ASCII value second (low) byte = 0
Number = 100 + 19*256 + 1*65.536 + 0 * 16.777.216 = 70.500

The value of the 4's byte binary number is 70500.

Sending a 4 byte value should be done by splitting up the value into 4 bytes using the same byte weights.

4.4 Programming tips

- It is advised to send all codes as bytes (8 bit), not as characters or a string which might consist of multiple or different bytes in some locales.
- It is also advisable to retry to send the command when no answer is returned within max. 0,5 second.
- Commands that request information are automatically confirmed by returning the requested information. Other commands are usually confirmed by returning the command code.

4.5 Error codes

The device can also return following error codes to indicate the command is not executed:

- 241 or 0xF1 : Code is recognised and not executed for some reason.
- 242 or 0xF2 : Code is recognised but not executable at the moment (e.g. not in stand-by).
- 243 or 0xF3 : Code is not recognised.
- 245 or 0xF5 : An error occurred in the data.
- 255 or 0xFF : There are more bytes expected. Send a complete protocol string or check the given value of size.

Hertenstraat 56/9
B-2300 Turnhout
Belgium
+32 (0)14 411279
+32 (0)14 429179
support@consort.be

5 Commands to be used at any time

The following commands can be used at any time, i.e. during stand-by and during the run.

	Code	*Description*
1.	105	Specific purpose codes
2.	10	Send a key
3.	205	Block manual key input (except STOP during a run)
4.	210	Enable manual key input
5.	25	Read method/phase
6.	120	Read data log parameters

5.1 Specific purpose code (105)

The code 105 can be used for 2 purposes:

- Read specific device information
- Unlock the write protection

5.1.1 Device information

This code requests specific information of the device (model, version and serial number). The required information needs to be specified with an extra byte:

0 : request the Model type
1 : request the Version
2 : request the Serial number

The returned information is given as ASCII characters which are encapsulated according to the protocol.

e.g. Request model:

```
56 03 69 00 C2 0D 0A
50 08 69 45 56 32 36 35 30 29 0D 0A  /* this is model EV2650 */
```

Request version:

```
56 03 69 01 C3 0D 0A
50 05 69 33 2E 30 4F 0D 0A           /* this is version 3.0 */
```

5.1.2 Unlock write protection

The code 105 can also be used to send the unlock codes before sending a command that requires such a sequence to be able to be executed (e.g. see code 197, store parameters). The 2 unlock codes are required to be sent consecutively. The command to be executed must follow immediately after sending the second unlock code. There should be no other communication done between the 2 unlock codes or another command sent for which it is intended. The device will relock the write protection in any other case.

199 : first unlock code
99 : second unlock code

e.g.

```
56 03 69 C7 89 0D 0A          /* 0xC7= 199, first unlock code */
50 02 69 BB 0D 0A
56 03 69 63 25 0D 0A          /* 0x63= 99, second unlock code */
50 02 69 BB 0D 0A
```

5.2 Send a key (10)

To send a key as if it were pressed through the keyboard. The device recognises the keys as follows:

1. Value = *1*, MINUS or DOWN key
2. Value = *2*, RUN_STOP key
3. Value = *4*, SET_ENTER key
4. Value = *8*, PLUS or UP key
5. Value = *16*, MENU key, special key only available through RS232 for directly returning to the previous menu. Manually this effect can be entered by pressing the SET key for at least 2 seconds.

e.g. Send the following characters to send the SET key:
'V' + asc(3) + asc(10) + asc(4) + asc(102) + asc(13) + asc(10)
The device will respond:
'P' + asc(3) + asc(10) + asc(240) + asc(77) + asc(13) + asc(10)

The information byte ascii(240) should be returned as a confirmation code to inform that the command was correct and accepted.

5.3 Block manual key input (205)

This code blocks manual key input except the STOP during a run for security reasons.

e.g.

```
56 02 CD 25 0D 0A
50 02 CD 1F 0D 0A
```

5.4 Enable manual key input (210)

This code enables manual key input.

e.g.

```
56 02 D2 2A 0D 0A
50 02 D2 24 0D 0A
```

5.5 Read Method/Phase (25)

This code requests for the actual settings of the method and its phase. Three information bytes are returned, the second byte equals the method (Method number – 1) and the third byte equals the phase (Phase number – 1). For the manual program, the method equals 10 and the phase equals 1.

The first byte contains general settings of the power supply:

- bit0 is 1 →Power fail detection is enabled
- bit1 is 1 →Alarm enabled when too low current
- bit2 is 1 →Manual method is being used
- the other bits are not used

e.g. (1)

```
56 02 19 71 0D 0A
50 05 19 7F 09 00 F6 0D 0A
```

The returned method equals 10 (9 + 1), the step is 1 (0 + 1). These are the settings for the manual method. bit2 of the first byte is also set = 1.

(2)

```
56 02 19 71 0D 0A
50 05 19 7B 02 01 EC 0D 0A
```

The returned method equals 3, the step is 2.

5.6 Read data log parameters (120)

This command returns information about the data logging. This data includes flags about the settings, the interval of time that is used and the number of logged values.

e.g. Characters to send:

'V' + asc(2) + asc(120) + asc(208) + asc(13) + asc(10)

The device responds for this command with 4 bytes of data as the information bytes:

'P' + asc(18) + asc(120) + asc(flags) + asc(interval) + asc(high byte number of values) + asc(low byte number of values) + asc(low byte Checksum) + asc(13) + asc(10)

flags = When the highest bit is zero, the data logging is activated.
Interval = The interval of time between the logged values in seconds.
Number of values = 2 byte number indicating the number of points that the device has logged.

5.7 Read parameters (30)

To read the parameters in memory for the current method (see also code 25) when in stand-by. The device responds with 17 bytes containing the following information:

- 4 value bytes of the preset Voltage in units of 0.1 V
- 4 value bytes of the preset Current in units of 0.01 mA
- 4 value bytes of the preset Power in units of 0.01 W
- 4 value bytes of the preset timer in units of seconds or 0.1 Volthours
- 1 byte with 3 bits of information for the settings of:
  - bit 0 → Timer: hours (0) or kiloVolthours (1)
  - bit 1 → End of method (0) or continue with next step (1)
  - bit 2 → regular Voltage control (0) or Voltage gradient control (1)

e.g. Request and answer of the parameters:

```
56 02 1E 76 0D 0A
50 13 1E D0 07 00 00 50 C3 00 00 98 3A 00 00 78 00 00 00 06 BB 0D 0A
        /*   V          mA          W         Timer      */
```

The current settings are:

D0 07 00 00	Voltage = 208 + 7 * 256 = 2000 units, corresponds to 200.0 V
50 C3 00 00	Current = 80 + 195 * 256 = 50000 units, corresponds to 500.00 mA (1)
98 3A 00 00	Power = 152 + 58 * 256 = 15000 units, corresponds to 150.00 W
78 00 00 00	Timer = 120 → 120 seconds or 12.0 Volthours
06	bit 0 = 0 → timer in seconds bit 1 = 1 → continue with next step, in case of method program! bit 2 = 1 → Voltage gradient control, in case of method program!

(1) For the models EV3330 and EV3620 will this correspond with 50.000 mA.

5.8 Set parameters (40)

It is also possible to set new parameters for the current active method/step by using the code 40. The parameters need to be given similarly as they can be read (see code 30).

e.g. Set V = 300 (3000 units, Hex. 0x000008BB),
mA = 400 (40000 units(2) Hex. 0X00009C40),
W = 100 (10000 units, Hex.0x00000E10),
timer = 3600 (seconds, Hex.0x00000E10),
last byte = 0 (no gradient, end of method, timer in hours).

```
56 13 28 BB 08 00 00 40 9C 00 00 10 27 00 00 10 0E 00 00 00 85 0D 0A
50 02 28 7A 0D 0A
```

(2) For EV3330 and EV3620: 40000 units correspond to 40.000 mA

Notes

When in run mode, do not send the timer value nor the last byte. In run mode, the timer can be adjusted using code 45 (7.3).

After setting the parameters in stand-by mode, will it be required to save them into the nonvolatile memory before continuing. Therefore are the following steps required to do a complete programming of 1 step:

1. Set the correct method and step (point 6.1, code 50)
2. Set the parameters (code 40)
3. Follow the unlock sequence (point 5.1, code 105 with 199 and 99)
4. Send the Store parameter code (point 6.2, code 197)

If the unlock sequence is not followed immediately with the code to store the parameters, storing of the parameters will fail.

It is best that the device is in the main menu in stand-by mode, not in enter of parameters mode.

6 Commands available during stand-by

Additionally to the commands of point 5, the instrument accepts several commands during the stand-by. These commands are not accepted during a run and will cause an error code to be returned.

Code	*Description*
50	Set Method and Step
197	Store parameters
85	Read the logged data

6.1 Set Method and Step (50)

To set the current method and phase for reading or entering the parameters. Only valid method and step numbers are accepted. These values should be less 1 than the number that is displayed. The manual method should be numbered as 9 (10-1) with step 0 (1-1).

e.g. Set the manual mode as method.

```
56 04 32 09 00 95 0D 0A
50 02 32 84 0D 0A
```

6.2 Store parameters (197)

When new parameters (code 40) and the correct combination method/step (code 50) has been set into the memory, the parameters can be stored in the non-volatile memory using the code 197.

To prevent erratic use and/or disturbances, a write protection has been built in. To be able to use the code, the device needs to be unlocked (see unlock codes) just before sending the command.

6.3 Reading the logged data (85)

Once known how many points have been recorded, they can be read from the memory by reading blocks of 8 bytes. The log information starts at the memory address 3956, the next block is placed at 3964 and so on.

Each block contains the following information:

- The first 2 bytes contains information about the settings, the control, the method and the step that was recorded. These bytes can also indicate a special code when abnormalities were recorded such as Alarm conditions, the program was stopped, ended by timer …
- The next 2 bytes contain the measured voltage.
- The next 2 bytes contain the measured current.
- The next 2 bytes contain the remaining time in minutes or 0.10 kVh (depending on the setting in the instrument) when a timer was set (down counting), otherwise it is the time that has passed (up counting).

The ASCII code to read a block from the memory is 85. The command to read a block of 8 bytes from a given address can be sent as follows:

Characters to send:
'V' + asc(7) + asc(85) + asc(1st byte address) + asc(2nd byte address) + asc(3rd byte address) + asc(4th byte address) + asc(8) + asc(low byte Checksum) + asc(13) + asc(10)

The device responds for this command with 8 bytes of data as the information bytes:

'P' + asc(10) + asc(85) + asc(byte 1) + asc(byte 2) + asc(byte 3) + asc(byte 4) + asc(byte 5) + asc(byte 6) + asc(byte 7) + asc(byte 8) + asc(low byte Checksum) + asc(13) + asc(10)

The first 2 bytes (in normal operation) contains important information about the settings and the regulation at that point.

- Byte 1, bit 7 = no importance, to be ignored
- Byte 1, bit 6 = hours (1) or kiloVolthours (0)
- Byte 1, bit 5 = End method (1) at the end of this step or proceed with next step (0)
- Byte 1, bit 4 = Fixed voltage (1) or gradient (0)
- Byte 1, bit 3 = Regulated (1) or not (0)
- Byte 1, bit 2 = Power is kept constant
- Byte 1, bit 1 = Current is kept constant
- Byte 1, bit 0 = Voltage is kept constant

- Byte 2, bit 4-7 = number Method – 1, Manual program = 9
- Byte 2, bit 0-3 = number Step – 1, Manual program = 0

The first 2 bytes can also contain the following special codes (values of byte 1 and 2):

- Alarm FAIL asc(255) + asc(247) or Hex: FF F7
- Alarm Low Current asc(255) + asc(248) or Hex: FF F8
- Alarm Ground Leak asc(255) + asc(249) or Hex: FF F9

- Alarm SHORT asc(255) + asc(250) or Hex: FF FA
- Alarm Low Resistance asc(255) + asc(251) or Hex: FF FB
- Power Fail asc(255) + asc(252) or Hex: FF FC
- Stopped asc(255) + asc(253) or Hex: FF FD
- End Run asc(255) + asc(254) or Hex: FF FE
- End Log asc(255) + asc(255) or Hex: FF FF

The third and fourth byte contain the voltage (in Volts). See **Conversion numbers** (4.3) to convert these 2 bytes to a useable value. This is similar for the current (bytes 5 and 6, in mA) and the timer (bytes 7 and 8, minutes or 0.1 kVh).

The highest address to use is the base address 3956 + (number of logged points * 8 bytes). This last block should normally contain the 'End Log' code. There are maximal 3600 points logged.

Hertenstraat 56/9
B-2300 Turnhout
Belgium
+32 (0)14 411279
+32 (0)14 429179
support@consort.be

7 Commands available during the run

While a run is executed (program is running), the following commands can also be used additionally to those from point 5.

	Code	*Description*
1.	15	Read measurements
2.	20	Read timer
3.	35	Read flags (several settings)
4.	30	Read parameters
5.	40	Set parameters
6.	45	Set timer

7.1 Read measurements (15)

This command allows to read the voltage, current, power and resistance during a run. The device returns 16 information bytes, 4 bytes for each parameter: Voltage,Current, Power and Resistance of the charge.

e.g.

```
56 02 0F 67 0D 0A
50 12 0F CF 06 00 00 63 B2 00 00 17 1F 00 00 E8 0E 00 00 87 0D 0A
         /*   V   */ /*   mA  */ /*   W   */ /*  Ohm  */
```

The voltage is $207*2^0 + 6*2^8 + 0*2^{16} + 0*2^{24} = 1743$ measurement units = 174.3 V
The current is $99*2^0 + 178*2^8 + 0*2^{16} + 0*2^{24} = 45667$ measurement units = 456.67 mA
The power is $23*2^0 + 31*2^8 + 0*2^{16} + 0*2^{24} = 7959$ measurement units = 79.59 W
The resistance is $232*2^0 + 14*2^8 + 0*2^{16} + 0*2^{24} = 3816$ measurement units = 381.6 Ohm

7.2 Read timer (20)

This command allows to read the timer values during a run. The device returns 20 information bytes containing several timers:

- Total program run time in seconds
- Total program run time in 0.1 Volthours
- Downcounting timer in seconds of the current step
- Upcounting timer in seconds of the current step
- Voltage-time integrating timer (10 adjustments/second) which triggers 0.1 Volthour every 36000 units.

e.g.

```
56 02 14 6C 0D 0A
50 16 14 8C 02 00 00 38 01 00 00 84 0B 00 00 8C 02 00 00 D3 56 00 00 87 0D 0A
         /* Tot.h */ /*Tot.kVh*/ /* Down  */ /*  Up   */ /*  Vs   */
```

The Total timer is $140*2^0 + 2*2^8 + 0*2^{16} + 0*2^{24} = 652$ seconds
The Total kVh timer is $56*2^0 + 1*2^8 + 0*2^{16} + 0*2^{24} = 312$ units = 0.0312 kVh
The current Down timer is $132*2^0 + 11*2^8 + 0*2^{16} + 0*2^{24} = 2948$ seconds remaining
The current Up timer is $140*2^0 + 2*2^8 + 0*2^{16} + 0*2^{24} = 652$ seconds
The current integrating timer is $211*2^0 + 86*2^8 + 0*2^{16} + 0*2^{24} = 22227$ units

The time unit of the Up and Down timers depends on the settings of the active method. When hours have been chosen, the time units are seconds. When kVh was chosen, the time units are 0.1 Volthours or 0.0001 kVh.

7.3 Read Status (35)

This command allows to read the control status flags during a run. The device returns 2 information bytes.

- First byte
  - bit0: Control is active when 1, disabled when 0
  - bit1: User is active with modifying the settings during the run when 1
  - bit2: A stable control point has been reached
  - bit3: The control (run) has been paused

- Second byte
  - bit0 + bit1: Indicates which parameter is being held constant, when
    00 or decimal 0 →none
    01 or decimal 1 →Voltage
    10 or decimal 2 →Current
    11 or decimal 3 →Power

Other bits may change due to internal tests or controls, these are irrelevant as information.

e.g.

```
56 02 23 7B 0D 0A
50 04 23 05 11 8D 0D 0A
```

First byte = 0x05
- bit0 = 1 →Control/run is active
- bit2 = 1 →Stable control point has been reached

Second byte = 0x11 = 0b00010001
- bit0+bit1 = 01 →constant Voltage

7.4 Read parameters (30)

To read the parameters in memory for the current program (method/phase) during the run.
Same as 5.7.

e.g.
```
56 02 1E 76 0D 0A
50 12 1E 10 27 00 00 F0 49 02 00 30 75 00 00 00 00 00 00 00 97 0D 0A
```

7.5 Set parameters (40)

Change the parameters (Voltage, Current, Power) for the running program. Six bytes of data need to be supplied to input the new parameters. The device confirms simply with the command code 40.

e.g.
Enter 2000 as Voltage = 200 V, 10000 as current = 100 mA [3] and 10000 as power = 100 W as the parameters for the active run.

```
56 0E 28 D0 07 00 00 50 C3 00 00 98 3A 00 00 48 0D 0A
50 01 28 79 0D 0A
```

(3) For the models EV3330 and EV3620, enter 100000 as 100 mA.