889A/889B – Remote Control Interface

The 889A and 889B benchtop LCR meters both use the same set of commands for remote communication over RS2332 (889A) or USB virtual COM (889B). This article will provide details on the different modes and settings available for remote communication, as well as details on how to decode some of the commands.

RS232/Virtual COM settings

For 889B, please install the USB drivers first prior to connecting the instrument to the PC via USB. The USB is a virtual COM port, thus behaving like a serial RS232 port. Windows will automatically assign a COM port, which can be verified from "Device Manager". The driver is named as CP210x.

Here are the serial port settings to use:

Baudrate: 9600
Parity: None
Data bits: 8
Stop bit: 1

Remote Interface Modes

There are 4 different remote modes available on these LCR meters.

NORMAL – This is the default mode of the LCR meter. When in this mode, the meter operates in LOCAL mode and front panel keys are accessible to setup and control the instrument.

BINNING – This mode is reserved for future use. Currently, it is set to operate in the same fashion as **NORMAL** mode.

REMOTE BINNING – In this mode, results of the measurement on the display are continuously sent to the USB port. This includes the measured readings, as well as all the settings that are shown on display.

REMOTE – In this mode, data will not be automatically sent to via USB continuously. Instead, users can send SCPI like commands to query, write, or read data from the LCR meter. This is the only mode where the SCPI like commands are available for use.

Remote Binning

In remote binning mode, strings of **hexadecimals** are continuously sent via USB to provide fast, updated measurements to the PC for interpretation and analysis. There's a standard format in which the LCR meter will follow by for every strings of data it sends. It follows the below structure:

Measurement Data (7 bytes or 11 bytes)	Setup/Parameter Data (6 bytes)

Measurement Data

There are two data formats for the measurement data string of hex. 7 bytes, and 11 bytes. 7 bytes are used when there's only one measurement made and shown on display. For example DCR function would have this format. 11 bytes are used when there are two measurements made and shown on both primary and secondary display. For example, Cp and D would have this format since two different measurements are made simultaneously. It can also be used when only secondary display readings are made, such as DCV function.

7 bytes Format

With the 7 bytes format scheme, the string of hex can be divided like the following:

02	03	M-B0	M-B1	M-B2	M-B3	CS

02 and 03: These are the leading codes that mark the beginning of the string of the 7 bytes format. These do not change.

M-B0 to M-3: These represent the actual measured data on the primary display of the LCR meter. It's represented in a total of **four bytes**, and it requires conversion to 32-bit floating point format. The values are in little endian format.

Example from DCR measurement:

M-B0	M-B1	M-B2	M-B3
9B	37	97	4B

1. Since it is in little endian format, change and reverse the order of the bytes.
   4B 97 37 9B
2. Convert the above string to 32-bit floating point format. This conversion may be tedious, but most software development languages already contain functions, libraries, or methods that can do a direct conversion between hex to 32-bit floating point or binary to 32-bit floating point. There are also many online websites that can compute the conversion for you simply by entering hexadecimals.

   In this example, **4B 97 37 9B** converts to **19820342**.

The measured DCR value on display is 19.82 Mohm. In this case, the converted value gives you the complete reading in ohms, with an addition of 4 extra digits of resolution.

CS: This is the checksum byte that is used to indicate the end of the string of data. According to the manual, it is indicated as: -((02+03+data_code) && 0x00FF). This is calculated in the following way:

Step 1: Add the first 6 bytes together.

Step 2: Convert the results into binary, and then do a logical AND operation with 0x00FF.

Step 3: Take the result and take the two's complement.

Step 4: Convert this back to hex to obtain the CS byte.

Example:

02	03	9B	37	97	4B	47

Step 1: Add first 6 bytes:

02 + 03 + 9B + 37 + 97 + 4B = 1B9

Step 2: Convert to binary:

1BA => 1 1011 1001

Logical AND operation with 0x00FF (= 0000 0000 1111 1111 in binary)

Converted binary	0000	0001	1011	1001
0x00FF(in binary)	0000	0000	1111	1111
Logical AND	0000	0000	1011	1001

Note: Since the first byte of 0x00FF will always be zero, logical AND operation of this byte will also always be zero. Thus, it can be ignored.

Result : 1011 1001

Step 3: Take the two's complement of **Result**

1011 1001 => 0100 0110 + 1 = **0100 0111**

Step 4: Convert back to hex to obtain CS byte

0100 0111 **= 47 (in hex)**

11bytes Format

There are two types of 11 bytes format that are used with these LCR meters. One type is available when there are two measurement data (i.e. Cp and D) that can be obtained from display. Another type is when only secondary display reading is available (i.e. DCV, ACV, DCA, ACA)

Dual Measurement Data type

With the 11 bytes format scheme, the string of hex can be divided like the following:

02	09	M-B0	M-B1	M-B2	M-B3	S-B0	S-B1	S-B2	S-B3	CS

02 and 09: These are the leading codes that mark the beginning of the string of the 11 bytes format. These do not change.

M-B0 to M-3: These represent the actual measured data on the primary display of the LCR meter. It's represented in a total of **four bytes**, and it requires conversion to 32-bit floating point format. The values are in little endian format.

S-B0 to S-B3: These represent the actual measured data on the secondary display of the LCR meter. It's represented in a total of **four bytes**, and it requires conversion to 32-bit floating point format. The values are in little endian format.

Example from Cp measurement on primary display:

M-B0	M-B1	M-B2	M-B3
D1	30	91	3F

1. Since it is in little endian format, change and reverse the order of the bytes.
   3F 91 30 D1
2. Convert the above string to 32-bit floating point format. This conversion may be tedious, but most software development languages already contain functions, libraries, or methods that can do a direct conversion between hex to 32-bit floating point or binary to 32-bit floating point. There are also many online websites that can compute the conversion for you simply by entering hexadecimals.

   In this example, **3F 91 30 D1** converts to **1.1343023**.

   The measured Cp value on display is 1.134.

Example from D measurement on secondary display:

S-B0	S-B1	S-B2	S-B3
3C	A7	90	3D

1. Since it is in little endian format, change and reverse the order of the bytes.
   3D 90 A7 3C
2. Convert the above string to 32-bit floating point format. This conversion may be tedious, but most software development languages already contain functions, libraries, or methods that can do a direct conversion between hex to 32-bit floating point or binary to 32-bit floating point. There are also many online websites that can compute the conversion for you simply by entering hexadecimals.

In this example, **3D 90 A7 3C** converts to **7.0631474e-2 or 0.070631474**.

The measured D value on the secondary display is .0706.

CS: This is the checksum byte that is used to indicate the end of the string of data. According to the manual, it is indicated as: -((02+09+data_code) && 0x00FF). This is calculated in the following way:

Step 1: Add the first 10 bytes together.

Step 2: Convert the results into binary, and then do a logical AND operation with 0x00FF.

Step 3: Take the result and take the two's complement.

Step 4: Convert this back to hex to obtain the CS byte.

Example:

02	09	D1	30	91	3F	3C	A7	90	3D	74

Step 1: Add first 10 bytes:

02 + 09 + D1 + 30 + 91 + 3F + 3C + A7 + 90 + 3D = 38C

Step 2: Convert to binary:

38C => 11 1000 1100

Logical AND operation with 0x00FF (= 0000 0000 1111 1111 in binary)

Converted binary	0000	0011	1000	1100
0x00FF(in binary)	0000	0000	1111	1111
Logical AND	0000	0000	1000	1100

Note: Since the first byte of 0x00FF will always be zero, logical AND operation of this byte will also always be zero. Thus, it can be ignored.

Result : 1000 1100

Step 3: Take the two's complement of **Result**

1000 1100 => 0111 0011 + 1 = **0111 0100**

Step 4: Convert back to hex to obtain CS byte

0111 0100 **= 74 (in hex)**

Secondary Display Reading type

With the 11 bytes format scheme, the string of hex can be divided like the following:

02	09	S-B0	S-B1	S-B2	S-B3	S-B0	S-B1	S-B2	S-B3	CS

02 and 09: These are the leading codes that mark the beginning of the string of the 11 bytes format. These do not change.

S-B0 to S-B3: There are two sets of these bytes that are exactly identical. One set can be treated as redundant data. These represent the actual measured data on the secondary display of the LCR meter when this display is the only display used for measurement (i.e. DCV function). It's represented in a total of **four bytes**, and it requires conversion to 32-bit floating point format. The values are in little endian format. We will only look at one set of 4 bytes here since the other set is the same and is redundant.

Example from DCV measurement on secondary display:

S-B0	S-B1	S-B2	S-B3
52	49	1D	3B

1. Since it is in little endian format, change and reverse the order of the bytes.
   3B 1D 49 52
2. Convert the above string to 32-bit floating point format. This conversion may be tedious, but most software development languages already contain functions, libraries, or methods that can do a direct conversion between hex to 32-bit floating point or binary to 32-bit floating point. There are also many online websites that can compute the conversion for you simply by entering hexadecimals.

   In this example, **3B 1D 49 52** converts to **2.4000001e-3**.

   The measured DCV value on the secondary display is 2.400 mV.

CS: This is the checksum byte that is used to indicate the end of the string of data. According to the manual, it is indicated as: -((02+09+data_code) && 0x00FF). This is calculated in the following way:

Step 1: Add the first 10 bytes together.

Step 2: Convert the results into binary, and then do a logical AND operation with 0x00FF.

Step 3: Take the result and take the two's complement.

Step 4: Convert this back to hex to obtain the CS byte.

The calculation is the same as that calculated for Dual Measurement Data Type above.

Note: The checksum still accounts for the first 10 bytes in the string even though the measurement data is repeated twice.

Setup/Parameter Data

After the string of 7bytes or 11 bytes are sent, the next string of bytes includes the settings and status of the instrument. It is in a frame of 6 bytes.

With the 6 bytes format scheme, the string of hex is divided like the following:

02	04	B0	B1	B2	CS

02 and 04: These are the leading codes that mark the beginning of the string of the 6 bytes format. These do not change.

B0 – B2: These represent the settings and status of the instrument. They are in little endian format, and conversion to binary is necessary to determine the settings and status of the instrument.

Example:

B0	B1	B2
D2	E2	85

1. Since it is in little endian format, first reverse the order:
   85 E2 D2
2. Convert to binary:
   85 E2 D2 = 1000 0101 1110 0010 1101 0010
3. There are total of 24 bits from the binary conversion.

Bit #	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary	1	0	0	0	0	1	0	1	1	1	1	0	0	0	1	0	1	1	0	1	0	0	1	0

Each bit or set of bits indicate a setting or status of the instrument. They are defined by the chart below:

Bit Position	LCR	DC/AC V/A
Bit 2 – 0	**Test Frequency**	**Reserved**
000	100 Hz	
001	120 Hz	
010	1 kHz	
011	10 kHz	
100	100 kHz	
101	200 kHz	
110	Reserved	
111	Reserved	
Bit 4 – Bit 3	**Test Level**	**Reserved**
00	50 mVrms	
01	250 mVrms	

10	1 Vrms	
11	Reserved	
Bit 5	**Reserved**	
0	Default	Default
1	Reserved	Reserved
Bit 6		
0	Relative	Relative
1	Normal	Normal
Bit 7		
0	Calibration	Calibration
1	Normal	Normal
Bit 10 – 8	**Primary Function**	**Reserved**
000	Lp	
001	Ls	
010	Cp	
011	Cs	
100	Z	
101	DCR	
110	Reserved	
111	Reserved	
Bit 12 – 11	**Secondary Function**	**Reserved**
00	D	
01	Q	
10	DEG(phase)	
11	ESR	
Bit 16 – 13	**Range Hold & unit**	
0000	RH nH	Reserved
0001	RH uH	RH mV, mA
0010	RH mH	RH V, A
0011	RH H	Reserved
0100	RH pF	
0101	RH nF	
0110	RH uF	
0111	RH mF	
1000	RH F	
1001	RH Ohm	
1010	RH kOhm	
1011	RH MOhm	
1100	Reserved	
1101	Reserved	
1110	Reserved	
1111	Auto-ranging	Auto-ranging
Bit 17		
0	Short Cal	Short Cal
1	Open Cal	Open Cal
Bit 21 – 18	**Measurement Modes**	

0000	Reserved
0001	LCR
0010	DCV
0011	ACV
0100	Diode
0101	Continuity
0110	DCA
0111	ACA
Others	Reserved
Bit 23 – 22	**Remote mode**
00	Normal
01	Binning
10	Remote Binning
11	Reserved

4. Taking our example, it translates to the following:

Bit #	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binary	1	0	0	0	0	1	0	1	1	1	1	0	0	0	1	0	1	1	0	1	0	0	1	0

Instrument settings and status:

Bits	Binary code	Representation
Bit 2 – 0	010	1 kHz
Bit 4 – 3	10	1 Vrms
Bit 5	0	Default
Bit 6	1	Normal
Bit 7	1	Normal
Bit 10 – 8	010	Cp
Bit 12 – 11	00	D
Bit 16 – 13	1111	Auto ranging
Bit 17	0	Short Cal
Bit 21 – 18	0001	LCR
Bit 23 – 22	10	Remote Binning

CS: This is the checksum byte that is used to indicate the end of the string of data. It is indicated as: -((02+04+data_code) && 0x00FF). This is calculated in the following way:

Step 1: Add the first 5 bytes together.

Step 2: Convert the results into binary, and then do a logical AND operation with 0x00FF.

Step 3: Take the result and take the two's complement.

Step 4: Convert this back to hex to obtain the CS byte.

The calculation is similar to the checksum calculations for 7 bytes and 11 bytes format. In this example, we take into consideration:

02	04	D2	E2	85	C1

Following the same steps, the checksum can be calculated as: **C1**

Example

Here's an example of what you might see through the USB sent from the instrument when in remote binning mode:

02 09 FA 10 91 3F CA 90 92 3D F2 02 04 D2 C2 04 62 02 09 09 11 91 3F 06 8E 92 3D A8 02 04 D2 C2 04 62 02 09 08 11 91 3F 4B 8F 92 3D 63 02 04 D2 C2 04 62……..

Green – These are the measurement readings. In this case, it is in the 11 bytes format.

Yellow – These are the instrument settings and status. It has the 6 bytes format.

Note: All the yellow highlighted bytes are the same because settings are always the same unless you change them. In this case, nothing has changed and therefore the bytes do not change.

Following the instructions from the above sections, we can obtain the following from this example:

02 04 D2 C2 04 62 – Meter is in: Cp primary mode, D secondary mode, 1 kHz frequency, 1 Vrms test level, RH on, short CAL, and readings for primary display is in uF (microfarads).

02 09 FA 10 91 3F CA 90 92 3D F2 – Measurement: Cp = 1.1333306 uF, D = 0.071565226

02 09 09 11 91 3F 06 8E 92 3D A8 – Measurement: Cp = 1.1333324 uF, D = 0.071559951

02 09 08 11 91 3F 4B 8F 92 3D 63 – Measurement: Cp = 1.1333323 uF, D = 0.071562372

Remote

In remote mode, the instrument will not continuously send measurement data and instrument status/settings. Instead, users can use the remote protocols that can be found in the user manual to send SCPI like commands for querying, writing, and reading between the instrument and PC. Refer to the user manual for complete set of supported protocols.