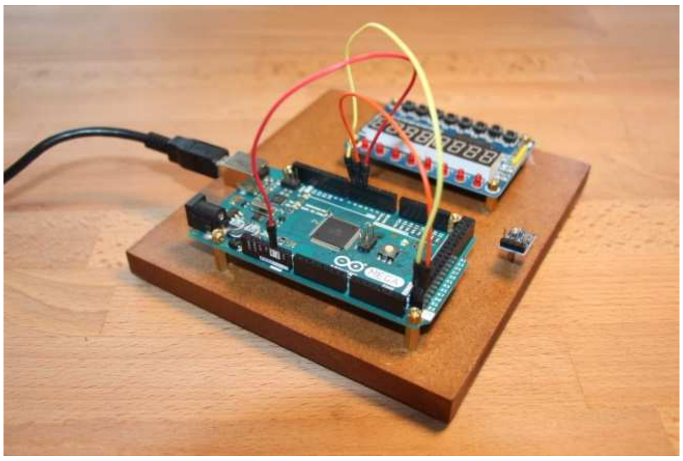
Monitor and modify variables with WawiLib

CONTENTS



1 Introduction

1.1 Objective of this document

The objective of this document is to describe how to use WawiLib to monitor and modify variables of different types, formats and sizes.

WawiWatchUsb.ino, a demo sketch supplied with the WawiSerialUsb library, will be used to explain the concepts. The example sketch declares a number of variables and arrays of variables. WawiLib will be used to observe and modify these variables on your PC.

You may ask yourself: Why this demo? The user's interface of WawiLib is quite simple. You will discover possibilities in this document that are not documented in the getting started tutorials for USB, Ethernet and WiFi.

1.2 Software and hardware requirements

The Arduino IDE (in this example 1.8.15) and WawiLib V2.0.x both need to be installed on your PC. The demo runs with licensed and unlicensed versions of WawiLib. During the grace period of 2 months, you can test and use all functions without registration. After this period registration is required in order to access all functions. At this time registration is free. In the future a small contribution might be required to register in order to support the website.

WawiLib supports multiple interface types: serial, software Serial, USB, USB-native, TCP/IP, UDP/IP via Ethernet or WiFi. In this demo, the USB programming port of the Arduino is used as the communication interface between WawiLib and the Arduino shield.

The hardware you need is an Arduino board, a USB programming cable and a Windows PC (32 or 64 bit). In this demo, we will use the Arduino MEGA board but other boards can be used in a similar or even identical way. For compatibility of boards, go to www.sylvestersolutions.com.

1.3 Required user experience

The concepts of this document build further on the tutorial "*Getting started WawiLib programming port*". Some knowledge of the C programming language, especially how variables are represented in memory, is an advantage.

2 THE “WawiWatchUSB” Demo sketch example

Many of the Arduino libraries come with examples. WawiLib is not an exception. In this demo, we will use the sketch called WawiWatchUsb.ino

- ✓ Open the demo using the menu File\Examples\WawiSerialUsb\WawiWatchUsb in the Arduino IDE.
- ✓ Compile and download the sketch to your board.

```
#include <WawiSerialUsb.h>

WawiSerialUsb WawiSrv;

// test variables for demo:
bool demoBool = true;
char demoChar = 'C';
unsigned char demoUChar = 46;
int demoInt = -10;
unsigned int demoUInt = 12;
long demoLong = 0x1212;
unsigned long demoULong = 0x1313;
float demoFloat = 3.14152;
double demoDouble = 3.14152;

bool demoBoolAr[10] = { 0,0,1,1,0,1,0,1,1,1 };
char demoCharAr[25] = "Hello world.";
unsigned char demoUCharAr[10];
int demoIntAr[10];
unsigned int demoUIntAr[10];
long demoLongAr[10];
unsigned long demoULongAr[10];
float demoFloatAr[10];
double demoDoubleAr[10];

// make variables of interest known to WawiLib:
// this function is used in WawiSrv.begin(....)
void wawiVarDef()
{
    WawiSrv.wawiVar(demoBool);
    WawiSrv.wawiVar(demoChar);
    WawiSrv.wawiVar(demoUChar);
    WawiSrv.wawiVar(demoInt);
    WawiSrv.wawiVar(demoUInt);
    WawiSrv.wawiVar(demoLong);
    WawiSrv.wawiVar(demoULong);
    WawiSrv.wawiVar(demoFloat);
    WawiSrv.wawiVar(demoDouble);

    WawiSrv.wawiVarArray(demoBoolAr);
    WawiSrv.wawiVarArray(demoCharAr);
    WawiSrv.wawiVarArray(demoUCharAr);
    WawiSrv.wawiVarArray(demoIntAr);
    WawiSrv.wawiVarArray(demoUIntAr);
    WawiSrv.wawiVarArray(demoLongAr);
    WawiSrv.wawiVarArray(demoULongAr);
    WawiSrv.wawiVarArray(demoFloatAr);
    WawiSrv.wawiVarArray(demoDoubleAr);
}
```

```
void setup()
{
    Serial.begin(115200);
    WawiSrv.begin(wawiVarDef, Serial, "My Arduino");
}

void loop()
{
    WawiSrv.loop();
}
```

The demo sketch contains a series of variables declared as (static) variables in the sketch. For each variable type an array example is included in order to demonstrate the use of arrays with WawiLib.

You will see that all these variables are declared as static variables (e.g. outside of {}). Why? WawiLib determines the address and size of a variable once by calling the function wawiVarDef() when a new variable is added in the variable table of WawiLib-PC. From then on, it remembers the address and size of a variable. If a variable would be declared locally in a function, it would be created and destroyed on the stack every loop. So, visualizing such a variable does not make a lot of sense because a real time application cycles through its main loop very fast and continuously in order to keep all services running (other types of approaches for automation application are possible but rarely used).

All data traffic is done in the WawiSrv.loop() and WawiSrv.delay() functions. Therefore, make sure that these functions are called without large interruptions. If you have a local loop, you might consider calling WawiSrv.loop() additionally in that loop. If you want to add a delay to your sketch do not call delay() but call WawiSrv.delay() instead. WawiSrv.delay() splits the entire waiting interval in small fragments and calls WawiSrv.loop() and delay() repeatedly. In this way, communication with WawiLib on the PC remains active and you get a swift response form WawiLib on the PC.

How does the communication between WawiLib and the Arduino board work?

The PC is the master of the communication. The Arduino is the slave. The PC sends requests to investigate variable memory addresses, to read bytes at memory addresses or to write bytes to other memory addresses. The Arduino replies with the values, addresses or acknowledgments depending on the request issued.

Starting from WawiLib 1.6.0, the Arduino can also send messages to the PC on its own initiative, this mechanism is used to support the WawSerialXXX.print() function.

During serial communication, the maximum message size is determined by the Arduino core Libraries and the buffer sizes in the WawiLib library objects. If you run into limits related to your maximum serial message size, increment the maximum length of the Arduino Serial library and WawiLib will use these values as default settings. The default length of 64 bytes should be sufficient in most cases.

In order to increase performance, you can also increase the buffers inside the `WawiSerialUsb` object in the Arduino, this can be done by overriding the default values during the construction of the `WawiSerialUsb` object:

```
WawiSerialUsb WawiSrv(/*RX_BUF*/128,/*TX_BUF*/128,/*PR_BUF*/ 128);
or
WawiSerialUsbLight WawiSrv(/*RX_BUF*/128,/*TX_BUF*/128);
```

WawiSerialUsbLight is the light weight version of WawiLibUsb, it is smaller as it does not support .print() statements to send info from your sketch to the WawiLib-PC output window. WawiSerialUsbLight does not support breakpoings.

The message buffers within WawiSerialUsb (and its counterparts for WiFi and Ethernet) can be larger than the internal Arduino buffers for a protocol. For example an Arduino Serial buffer on a Mega 2560 is typically 64 bytes. Data telegrams are split into packages by the software if the internal Arduino buffers are too small compared to the WawiLib message buffers. During initialization, WawiLib-PC reads the sizes of the buffers on the Arduino side to make sure that Arduino and PC are on the same page. Both Arduino protocol and WawiLib object buffers are investigated during startup.

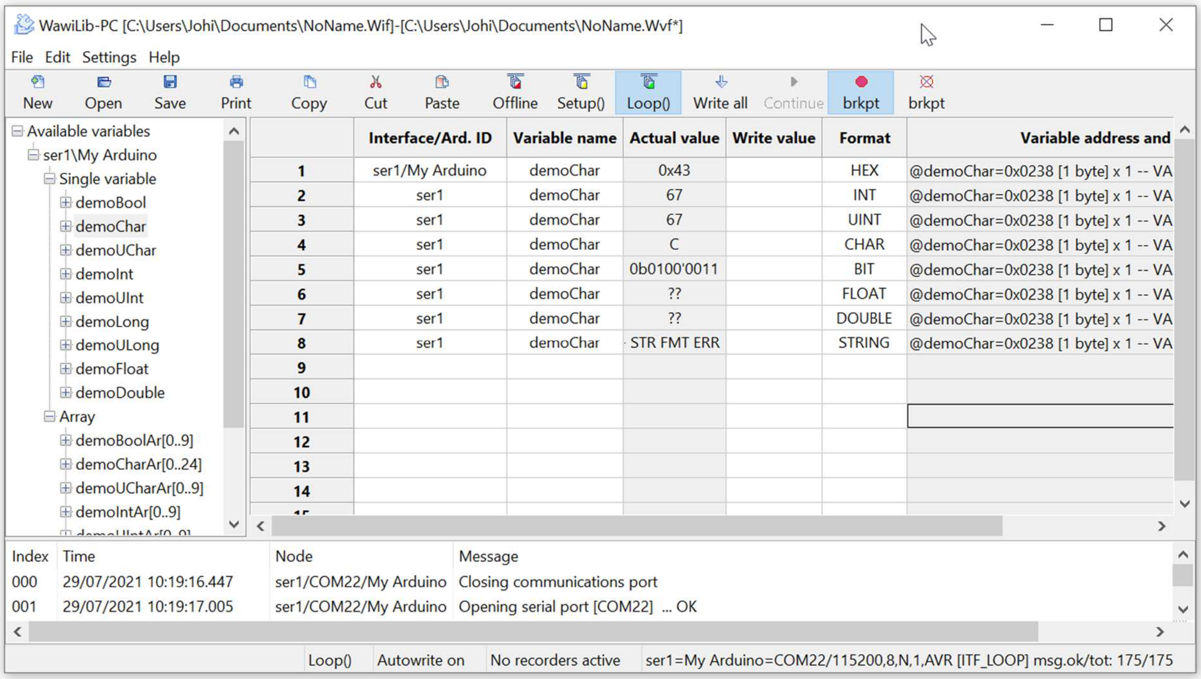
3 The WawiLib variable grid

3.1 Declaration type of a variable can differ from its displayed format.

In the Arduino processor, all variables are stored as series of bytes in memory. Whether you declare a variable as *char* or as *float*, both of them are stored as a number of bytes in the Arduino memory. The difference between *float* and *char* is very relevant once you compile the Sketch because the compiler needs to know the type of variable in order to translate your source code to the right CPU instructions.

WawiLib reads the byte(s) directly in the memory of the Arduino and puts them in the grid table. The type of formatting is determined by what you select in the WawiLib "Format" column. So, a char variable can be displayed as an *integer* and an *integer* variable can be displayed as a series of characters.

- ✓ Start WawiLib on your PC and fill in the table as below.
- ✓ Press "Setup()"
- ✓ You can fill in the variable names manually or use drag & drop from the tree.



WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]

Available variables
- ser1\My Arduino
  - Single variable
    - demoBool
    - demoChar
    - demoUChar
    - demoInt
    - demoUInt
    - demoLong
    - demoULong
    - demoFloat
    - demoDouble
  - Array
    - demoBoolAr[0..9]
    - demoCharAr[0..24]
    - demoUCharAr[0..9]
    - demoIntAr[0..9]

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable address and
1	ser1/My Arduino	demoChar	0x43		HEX	@demoChar=0x0238 [1 byte] x 1 -- VA
2	ser1	demoChar	67		INT	@demoChar=0x0238 [1 byte] x 1 -- VA
3	ser1	demoChar	67		UINT	@demoChar=0x0238 [1 byte] x 1 -- VA
4	ser1	demoChar	C		CHAR	@demoChar=0x0238 [1 byte] x 1 -- VA
5	ser1	demoChar	0b0100'0011		BIT	@demoChar=0x0238 [1 byte] x 1 -- VA
6	ser1	demoChar	??		FLOAT	@demoChar=0x0238 [1 byte] x 1 -- VA
7	ser1	demoChar	??		DOUBLE	@demoChar=0x0238 [1 byte] x 1 -- VA
8	ser1	demoChar	STR FMT ERR		STRING	@demoChar=0x0238 [1 byte] x 1 -- VA
9						
10						
11						
12						
13						
14						

Index	Time	Node	Message
000	29/07/2021 10:19:16.447	ser1/COM22/My Arduino	Closing communications port
001	29/07/2021 10:19:17.005	ser1/COM22/My Arduino	Opening serial port [COM22] ... OK

Loop() Autowrite on No recorders active ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 175/175

Fig 3.1. WawiLib with display of *demoChar* in various formats.

- ✓ Reduce the width of the “Recorder” column to 0 by dragging the right-hand column to the left.
- ✓ => Note: there is a “Reset Vieuw” option available in case you are lost in the UI.

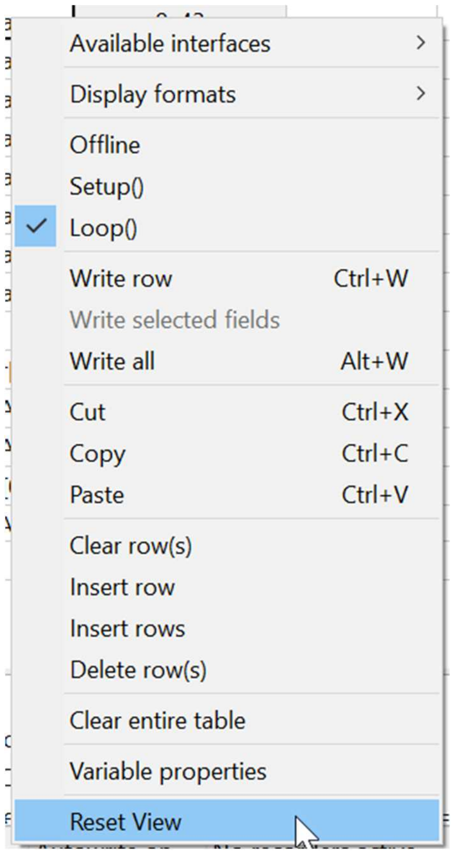


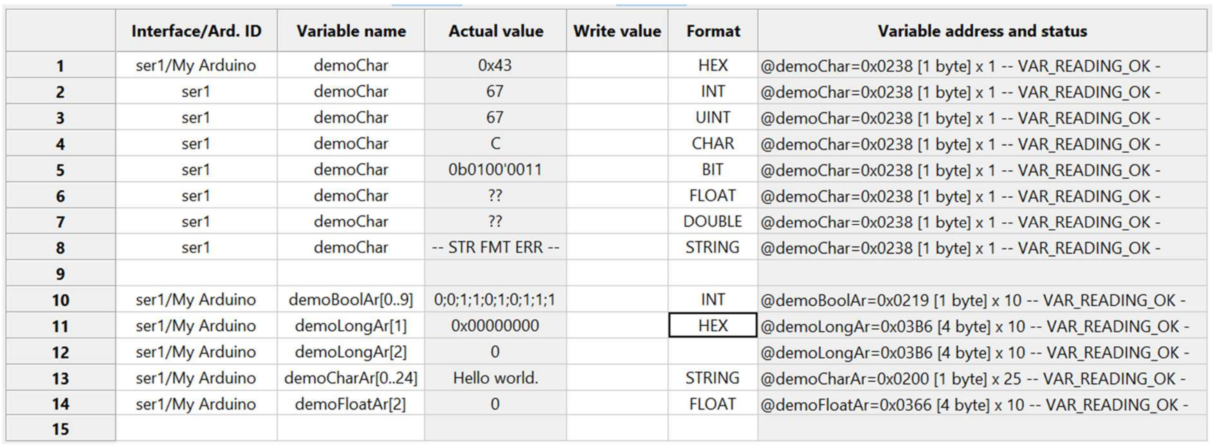
Fig 3.2. Reset view option of WawiLib user’s interface

In this example, the same USB interface is used for programming the Arduino and to communicate with WawiLib. Therefore, you cannot download your sketch to your board if WawiLib is online. In the same way, if you open a serial output window in the Arduino IDE, the Arduino IDE will claim the USB interface of your board and WawiLib will no longer be able to go online. So: during download, WawiLib needs to be offline in order to release the USB programming interface and vice versa.

If you want Arduino-IDE serial output and WawiLib communication at the same time, you can use another (serial or other) interface on your board for WawiLib. There are a number of USB to serial converters on the market. These interface converters can convert Arduino TTL logic to serial USB signals. The USB interface converter plugs into one of your PC’s USB ports.

The Mega 2560, for example, has 3 additional serial interfaces. SoftSerial and USB-native (Arduino DUE) are also supported by WawiLib. There is a tutorial explaining the details to be taken into account using USB to serial converters on www.sylvestersolutions.com.

In fig. 3.3, you see the WawiLib screen after filling in the grid wit a variety of variables. You see that the default interface was added and the actual values are displayed in the selected format. On the right-hand side, you see the address of the variable, its size and the array count. In the last column, you see the status of the last variable read and recording operation.



	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable address and status
1	ser1/My Arduino	demoChar	0x43		HEX	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
2	ser1	demoChar	67		INT	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
3	ser1	demoChar	67		UINT	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
4	ser1	demoChar	C		CHAR	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
5	ser1	demoChar	0b0100'0011		BIT	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
6	ser1	demoChar	??		FLOAT	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
7	ser1	demoChar	??		DOUBLE	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
8	ser1	demoChar	-- STR FMT ERR --		STRING	@demoChar=0x0238 [1 byte] x 1 -- VAR_READING_OK -
9						
10	ser1/My Arduino	demoBoolAr[0..9]	0;0;1;1;0;1;0;1;1;1		INT	@demoBoolAr=0x0219 [1 byte] x 10 -- VAR_READING_OK -
11	ser1/My Arduino	demoLongAr[1]	0x00000000		HEX	@demoLongAr=0x03B6 [4 byte] x 10 -- VAR_READING_OK -
12	ser1/My Arduino	demoLongAr[2]	0			@demoLongAr=0x03B6 [4 byte] x 10 -- VAR_READING_OK -
13	ser1/My Arduino	demoCharAr[0..24]	Hello world.		STRING	@demoCharAr=0x0200 [1 byte] x 25 -- VAR_READING_OK -
14	ser1/My Arduino	demoFloatAr[2]	0		FLOAT	@demoFloatAr=0x0366 [4 byte] x 10 -- VAR_READING_OK -
15						

Fig 3.3. WawiLib grid with a variety of variables and arrays of variables.

3.2 The column “Interface/Arduino ID”

In the column “Interface/Arduino ID”, you can see the name of the interface and the name of the board. If you have only 1 board connected to WawiLib, this column will be automatically filled in when you go online. If not, you can fill in the name of the interface manually (ser1 in this case) or click right and select the right interface from the menu.

If your boards are online, the menu will show the name you gave to the board in the line WawiSrv.begin(“name”) of your sketch. In the figure below, I connected an additional Arduino Uno to my USB which was given the name “My Arduino Uno” whereas my Mega2560 has the name “My Arduino”.

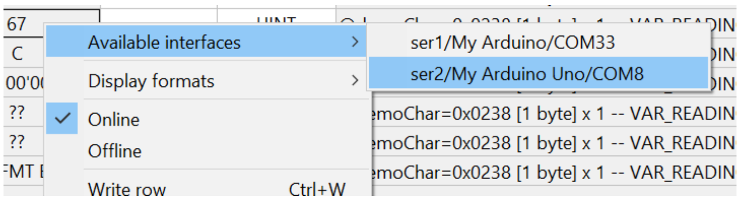


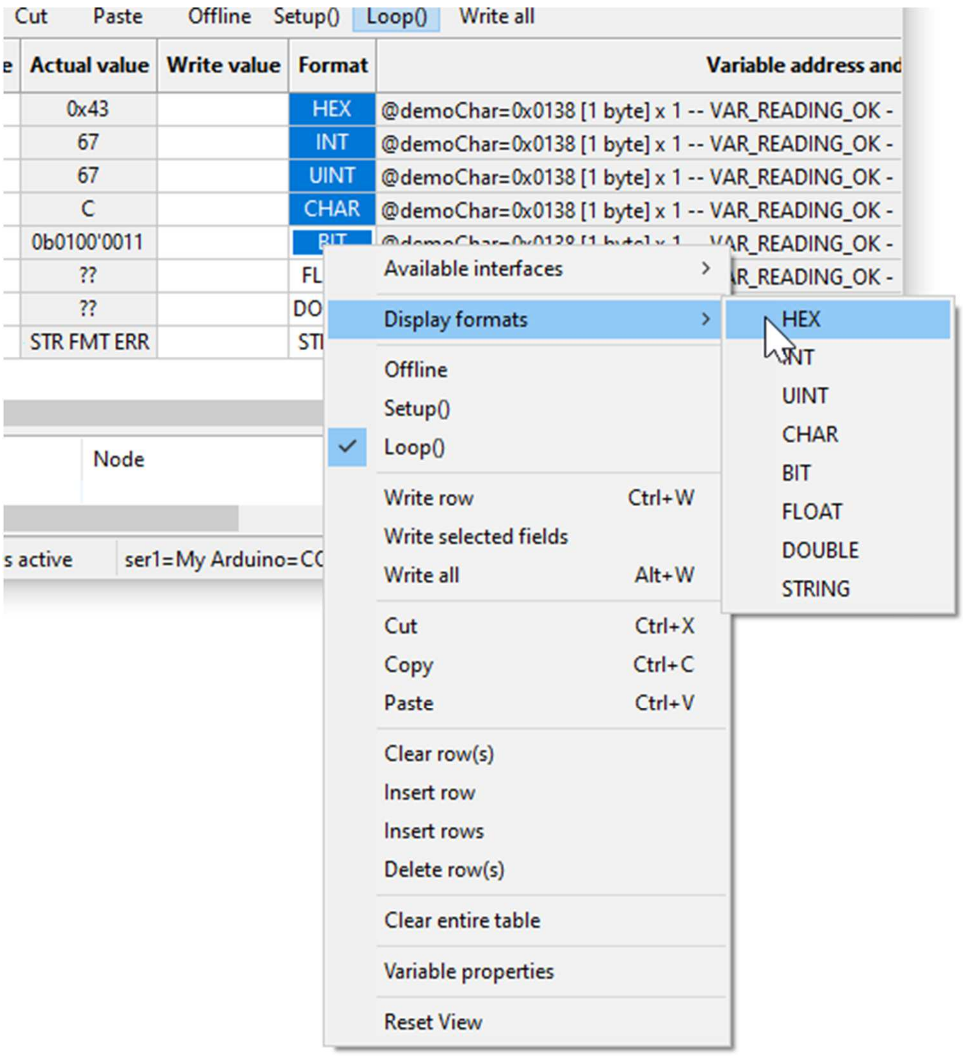
Fig 3.4. WawiLib Interface selection if linked to multiple Arduinos at the same time.

3.3 The column “Arduino variable name”

In the column “Arduino variable name”, you enter the name of the variable you are interested in. The names are case sensitive. WawiLib supports arrays and struct fields. You can use [] to address an individual element in an array and [X..Y] to address a range of elements. If the requested range is too large for the communication protocol maximum message size, use a smaller range. The limitations are on the range requested and not on the size of the array as it is defined in your sketch. So *demoVar* [1000..1010] could be OK whereas *demoVar*[0..1000] is not.

3.4 The columns “Actual value” and “Format”

In the column ‘Actual value’, the actual value of the variable is displayed in accordance with the requested format in the ‘Format’ column. Multiple elements of an array will be separated by a ‘;’.



Actual value	Write value	Format	Variable address and
0x43		HEX	@demoChar=0x0138 [1 byte] x 1 -- VAR_READING_OK -
67		INT	@demoChar=0x0138 [1 byte] x 1 -- VAR_READING_OK -
67		UINT	@demoChar=0x0138 [1 byte] x 1 -- VAR_READING_OK -
C		CHAR	@demoChar=0x0138 [1 byte] x 1 -- VAR_READING_OK -
0b0100'0011		BIT	VAR_READING_OK -
??		FL	R_READING_OK -
??		DO	
STR FMT ERR		STI	

Fig 3.5. WawiLib variable display format selection.

Values can be displayed in various formats:

- bit (BIT), e.g. 0b010’000
- character format (CHAR), e.g. H, \STX, \045
- hexadecimal format (HEX), e.g. 0x48
- signed or unsigned int format(INT/UINT), e.g. 101
- string format (STRING), e.g. Hello world.
- floating point (FLOAT/ DOUBLE) e.g. 3.5e-5

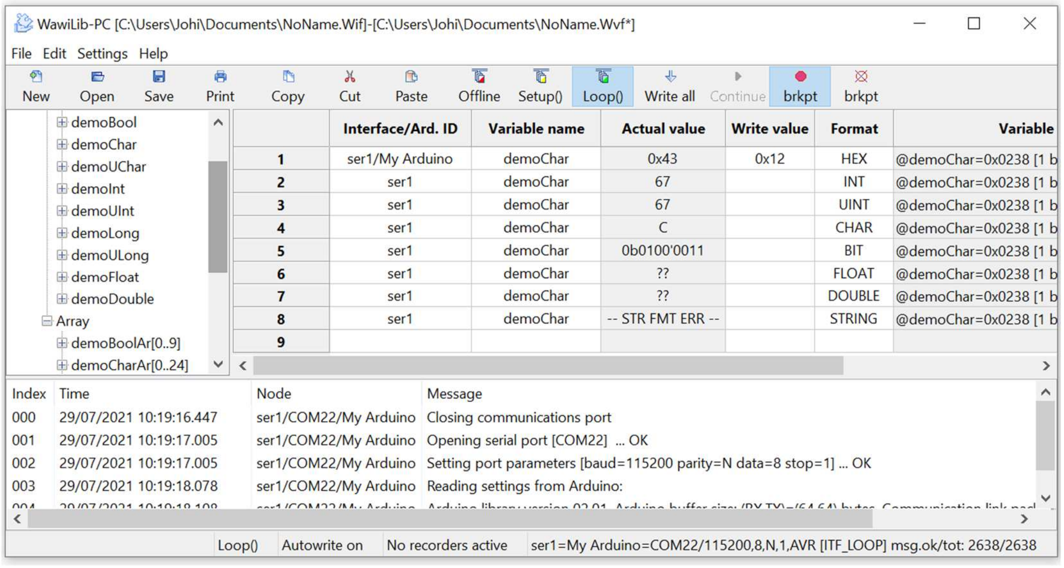
Arrays need to be 0-terminated in order to display them as a string. The index range selected [x..y] needs to include the location of the 0 end-of-string terminator. If not, WawiLib will indicate “STR FMT ERR” (string format error) for the actual value of the variable when displayed as string.

You do not need to select a format for each variable individually, you can select multiple rows and then click right to define the format for all variables. The same goes for other settings (interface types, recorders etc.).

3.5 The column “Write value”

In the write value column, you enter a new target value for a variable.

- Enter the value 0x12 as new value for *demoCharAr[0]* as indicated in the next figure.


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong
demoULong
demoFloat
demoDouble
Array
demoBoolAr[0..9]
demoCharAr[0..24]

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable
1	ser1/My Arduino	demoChar	0x43	0x12	HEX	@demoChar=0x0238 [1 b
2	ser1	demoChar	67		INT	@demoChar=0x0238 [1 b
3	ser1	demoChar	67		UINT	@demoChar=0x0238 [1 b
4	ser1	demoChar	C		CHAR	@demoChar=0x0238 [1 b
5	ser1	demoChar	0b0100'0011		BIT	@demoChar=0x0238 [1 b
6	ser1	demoChar	??		FLOAT	@demoChar=0x0238 [1 b
7	ser1	demoChar	??		DOUBLE	@demoChar=0x0238 [1 b
8	ser1	demoChar	-- STR FMT ERR --		STRING	@demoChar=0x0238 [1 b
9						

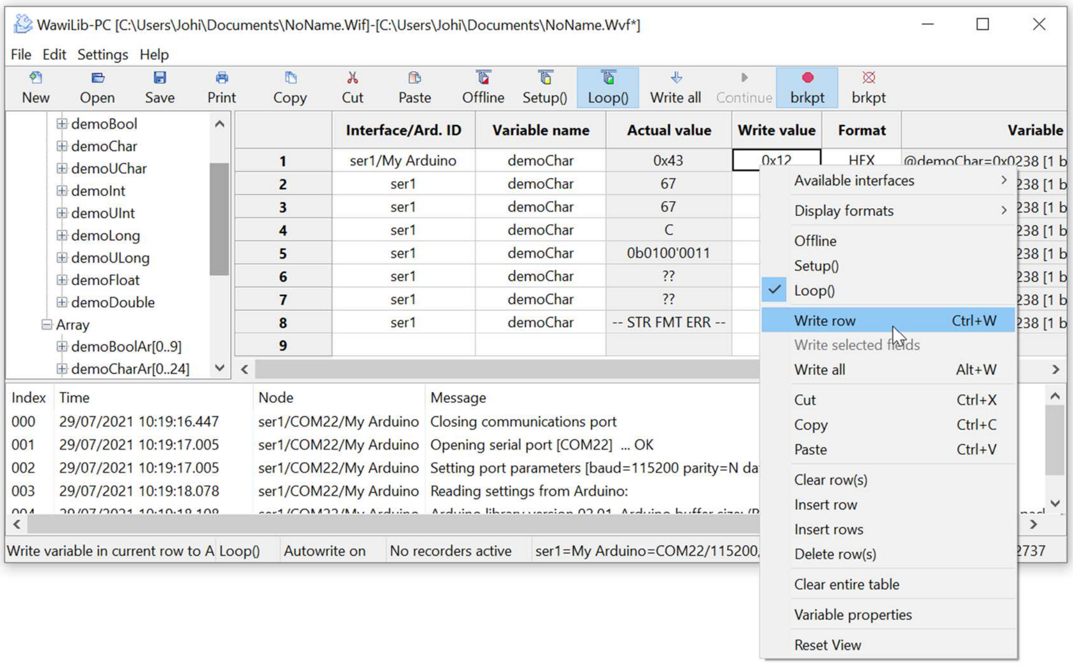
Index	Time	Node	Message
000	29/07/2021 10:19:16.447	ser1/COM22/My Arduino	Closing communications port
001	29/07/2021 10:19:17.005	ser1/COM22/My Arduino	Opening serial port [COM22] ... OK
002	29/07/2021 10:19:17.005	ser1/COM22/My Arduino	Setting port parameters [baud=115200 parity=N data=8 stop=1] ... OK
003	29/07/2021 10:19:18.078	ser1/COM22/My Arduino	Reading settings from Arduino:
004	29/07/2021 10:19:18.108	ser1/COM22/My Arduino	Arduino library version 02.01 Arduino buffer size: (RX,TX)=(64,64) bytes Communication link ...

Loop() Autowrite on No recorders active ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 2638/2638

Fig 3.6. Enter 0x12 as new value for demoChar on line 1.

Write jobs can be triggered in multiple ways:

- Using the toolbar above the grid, press the icon “Write All”.
- Using right click on the grid and selecting “Write row” or “Write All”.
- Using the shortcuts CTRL+W (Write row) or ALT+W (Write all).
- Pressing the “Enter” key while selecting the “Write value” field when the mode “Autowrite” is activated.



	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable
1	ser1/My Arduino	demoChar	0x43	0x12	HEX	@demoChar=0x0238 [1 b
2	ser1	demoChar	67			238 [1 b
3	ser1	demoChar	67			238 [1 b
4	ser1	demoChar	C			238 [1 b
5	ser1	demoChar	0b0100'0011			238 [1 b
6	ser1	demoChar	??			238 [1 b
7	ser1	demoChar	??			238 [1 b
8	ser1	demoChar	-- STR FMT ERR --			238 [1 b
9						

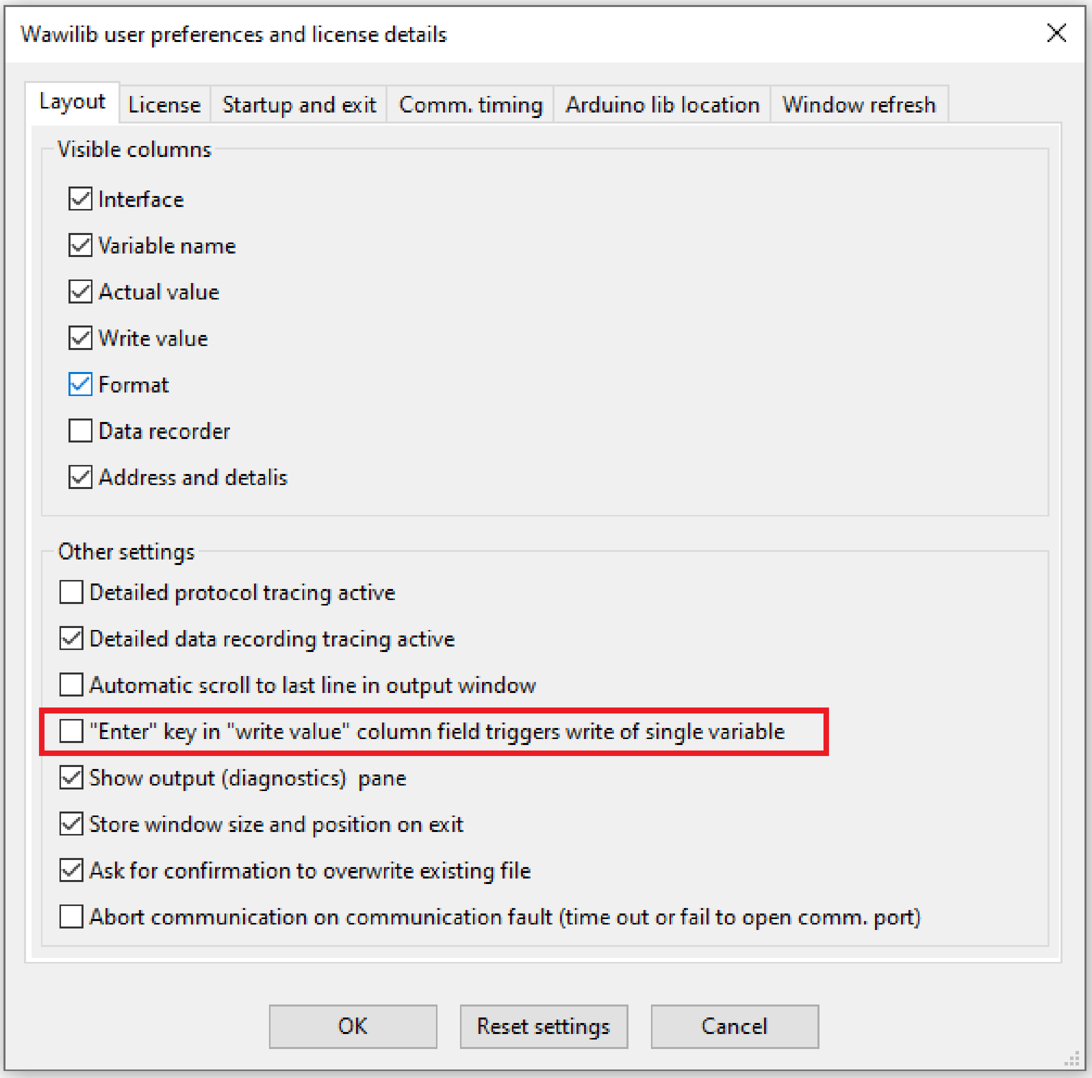
Index	Time	Node	Message
000	29/07/2021 10:19:16.447	ser1/COM22/My Arduino	Closing communications port
001	29/07/2021 10:19:17.005	ser1/COM22/My Arduino	Opening serial port [COM22] ... OK
002	29/07/2021 10:19:17.005	ser1/COM22/My Arduino	Setting port parameters [baud=115200 parity=N da
003	29/07/2021 10:19:18.078	ser1/COM22/My Arduino	Reading settings from Arduino:
004	29/07/2021 10:19:18.108	ser1/COM22/My Arduino	Arduino library version 02.01 Arduino buffer size: (R

Write variable in current row to A Loop() Autowrite on No recorders active ser1=My Arduino=COM22/115200, 2737

Fig 3.7. Trigger write row to write to set 0x12 as new value for *demoChar*.

You can see if WawiLib was able to write by checking the test in the output window. All write jobs will result in a status message in the output window if the option “Display diagnostics messages” is enabled. (Click right in the output window and enable this option in the pop-up menu if required.)

“Autowrite” can be enabled by using the “WawiLib user Preferences and license” dialog box. This dialog box is available under the menu “Settings/Preferences and license”.



Visible columns

- [✓] Interface
- [✓] Variable name
- [✓] Actual value
- [✓] Write value
- [✓] Format
- [ ] Data recorder
- [✓] Address and detalis

Other settings

- [ ] Detailed protocol tracing active
- [✓] Detailed data recording tracing active
- [ ] Automatic scroll to last line in output window
- [ ] "Enter" key in "write value" column field triggers write of single variable
- [✓] Show output (diagnostics) pane
- [✓] Store window size and position on exit
- [✓] Ask for confirmation to overwrite existing file
- [ ] Abort communication on communication fault (time out or fail to open comm. port)

Fig. 3.8. How to enable “Autowrite”.

If you check *“Enter” key in “write value” column field triggers write of single variable*, “Autowrite” will be enabled.

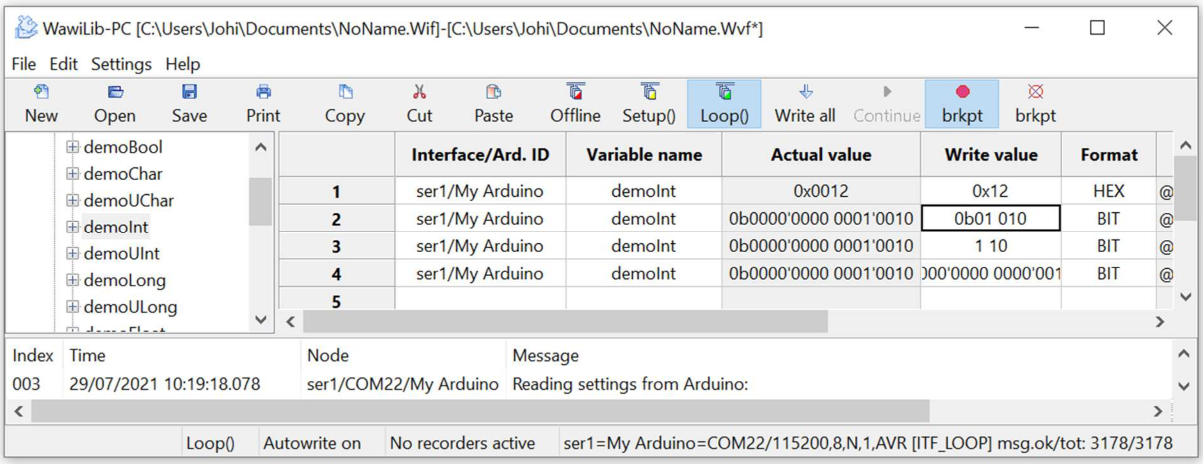
Write values have the same value format as their read counterparts. Format checks are done before writing. If the format of the entered value is not ok, a status message in the output window will indicate the nature of the problem.

3.5.1 Write bit format (BIT)

The figure below illustrates the use of BIT format. Some variables are more than 1 byte long. Writing multiple bytes is done leaving a space between the 8-bit values. The 8-bit values can be preceded by ‘0b’ but this is not mandatory. A single quote (‘) can be used to separate between the nibbles (nibble = series of 4 bits) to make values more readable.

If the write value is incomplete, WawiLib will try to complete the value; the input value "1 1" will be translated to 0b0000’0001 0b0000’0001 as will "0’1 0’1" but the input value "01" will be flagged illegal writing a 2-byte variable in the Arduino.

In order to get confirmation of your write operations in the output window, you need to enable the option “Display diagnostics messages” by clicking right in the output window (bottom window).


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong
demoULong

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	
1	ser1/My Arduino	demoInt	0x0012	0x12	HEX	@
2	ser1/My Arduino	demoInt	0b0000'0000 0001'0010	0b01 010	BIT	@
3	ser1/My Arduino	demoInt	0b0000'0000 0001'0010	1 10	BIT	@
4	ser1/My Arduino	demoInt	0b0000'0000 0001'0010	000'0000 0000'001	BIT	@
5						

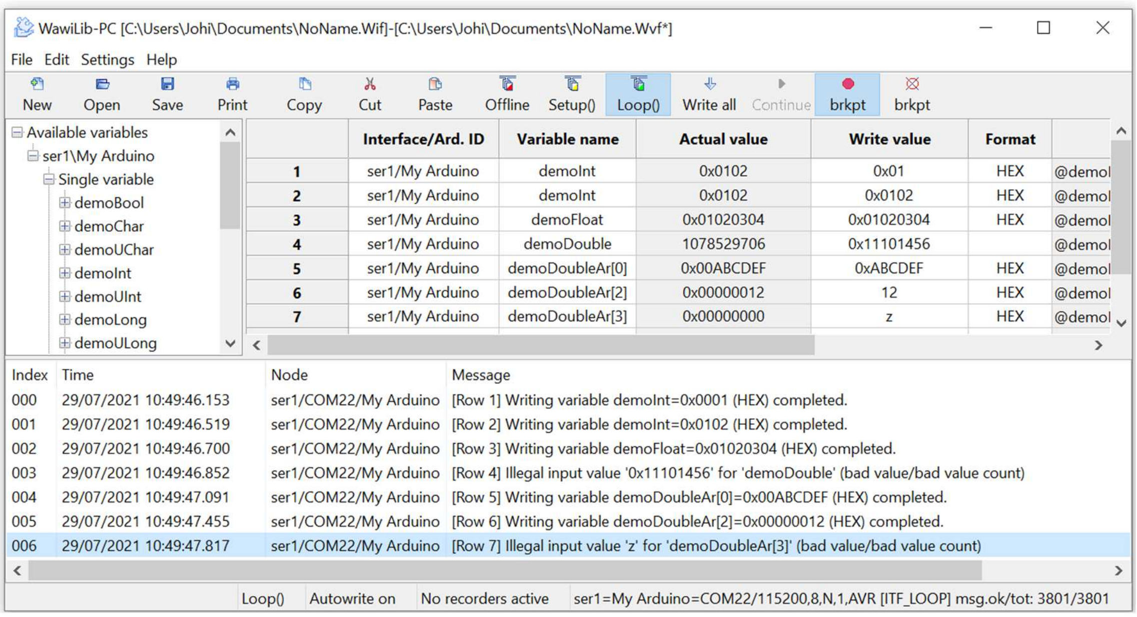
Index	Time	Node	Message
003	29/07/2021 10:19:18.078	ser1/COM22/My Arduino	Reading settings from Arduino:

Loop() Autowrite on No recorders active ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 3178/3178

Fig. 3.9. Bit write formatting.

3.5.2 Write hexadecimal format (HEX)

In the figure below (see tracing output window), the fact that 0x was forgotten when writing the value hex 12 is not a problem as WawiLib knows that the format of the output field is HEX. However, writing the value of Z as a hexadecimal value is illegal.


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
ser1\My Arduino
Single variable
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong
demoULong

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	
1	ser1/My Arduino	demoInt	0x0102	0x01	HEX	@demoI
2	ser1/My Arduino	demoInt	0x0102	0x0102	HEX	@demoI
3	ser1/My Arduino	demoFloat	0x01020304	0x01020304	HEX	@demoI
4	ser1/My Arduino	demoDouble	1078529706	0x11101456		@demoI
5	ser1/My Arduino	demoDoubleAr[0]	0x00ABCDEF	0xABCDEF	HEX	@demoI
6	ser1/My Arduino	demoDoubleAr[2]	0x00000012	12	HEX	@demoI
7	ser1/My Arduino	demoDoubleAr[3]	0x00000000	z	HEX	@demoI

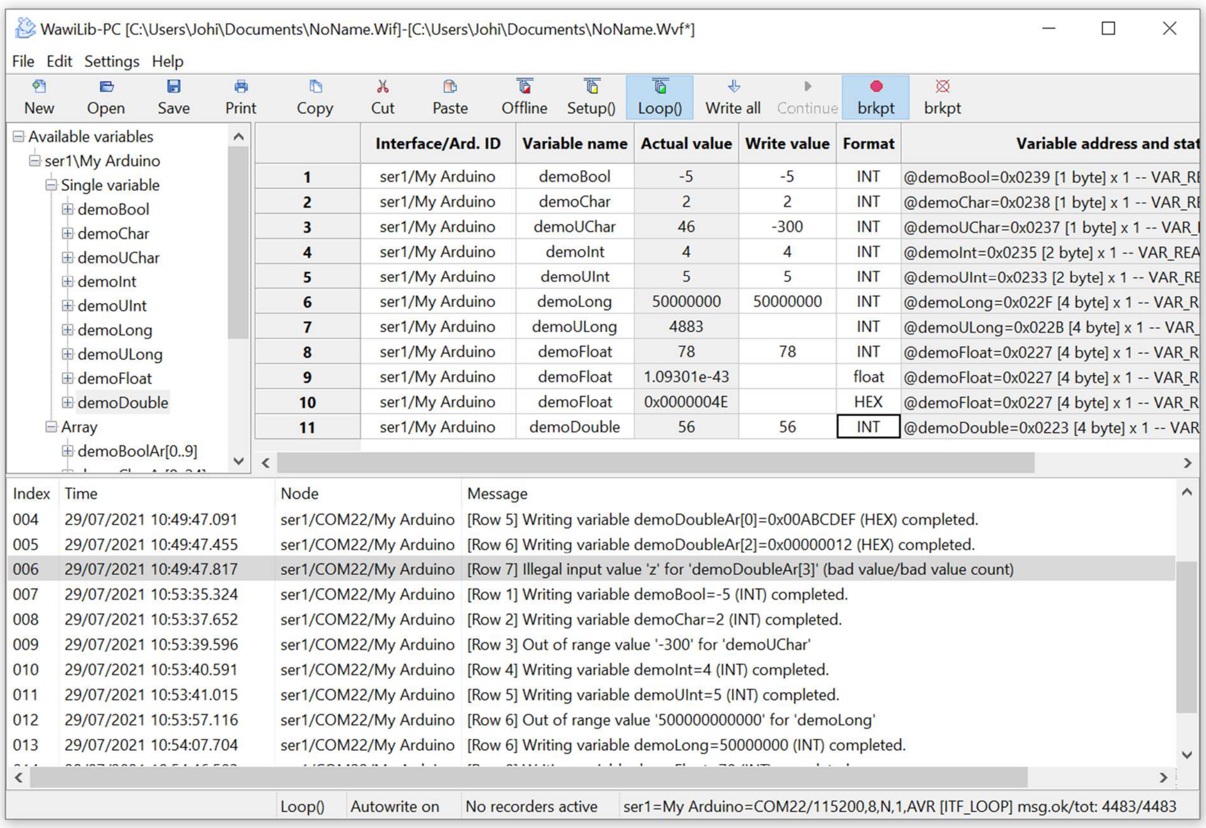
Index	Time	Node	Message
000	29/07/2021 10:49:46.153	ser1/COM22/My Arduino	[Row 1] Writing variable demoInt=0x0001 (HEX) completed.
001	29/07/2021 10:49:46.519	ser1/COM22/My Arduino	[Row 2] Writing variable demoInt=0x0102 (HEX) completed.
002	29/07/2021 10:49:46.700	ser1/COM22/My Arduino	[Row 3] Writing variable demoFloat=0x01020304 (HEX) completed.
003	29/07/2021 10:49:46.852	ser1/COM22/My Arduino	[Row 4] Illegal input value '0x11101456' for 'demoDouble' (bad value/bad value count)
004	29/07/2021 10:49:47.091	ser1/COM22/My Arduino	[Row 5] Writing variable demoDoubleAr[0]=0x00ABCDEF (HEX) completed.
005	29/07/2021 10:49:47.455	ser1/COM22/My Arduino	[Row 6] Writing variable demoDoubleAr[2]=0x00000012 (HEX) completed.
006	29/07/2021 10:49:47.817	ser1/COM22/My Arduino	[Row 7] Illegal input value 'z' for 'demoDoubleAr[3]' (bad value/bad value count)

Loop() Autowrite on No recorders active ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 3801/3801

Fig. 3.10. Bit Hex formatting.

3.5.3 Write integer format (INT/UINT)

Writing of (signed) integers (INT) and unsigned integers (UINT) is straightforward and does not need further detailing.


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
ser1\My Arduino
Single variable
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong
demoULong
demoFloat
demoDouble
Array
demoBoolAr[0..9]

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable address and stat
1	ser1/My Arduino	demoBool	-5	-5	INT	@demoBool=0x0239 [1 byte] x 1 -- VAR_RI
2	ser1/My Arduino	demoChar	2	2	INT	@demoChar=0x0238 [1 byte] x 1 -- VAR_RI
3	ser1/My Arduino	demoUChar	46	-300	INT	@demoUChar=0x0237 [1 byte] x 1 -- VAR_I
4	ser1/My Arduino	demoInt	4	4	INT	@demoInt=0x0235 [2 byte] x 1 -- VAR_REA
5	ser1/My Arduino	demoUInt	5	5	INT	@demoUInt=0x0233 [2 byte] x 1 -- VAR_RE
6	ser1/My Arduino	demoLong	50000000	50000000	INT	@demoLong=0x022F [4 byte] x 1 -- VAR_R
7	ser1/My Arduino	demoULong	4883		INT	@demoULong=0x022B [4 byte] x 1 -- VAR_
8	ser1/My Arduino	demoFloat	78	78	INT	@demoFloat=0x0227 [4 byte] x 1 -- VAR_R
9	ser1/My Arduino	demoFloat	1.09301e-43		float	@demoFloat=0x0227 [4 byte] x 1 -- VAR_R
10	ser1/My Arduino	demoFloat	0x0000004E		HEX	@demoFloat=0x0227 [4 byte] x 1 -- VAR_R
11	ser1/My Arduino	demoDouble	56	56	INT	@demoDouble=0x0223 [4 byte] x 1 -- VAR

Index	Time	Node	Message
004	29/07/2021 10:49:47.091	ser1/COM22/My Arduino	[Row 5] Writing variable demoDoubleAr[0]=0x00ABCDEF (HEX) completed.
005	29/07/2021 10:49:47.455	ser1/COM22/My Arduino	[Row 6] Writing variable demoDoubleAr[2]=0x00000012 (HEX) completed.
006	29/07/2021 10:49:47.817	ser1/COM22/My Arduino	[Row 7] Illegal input value 'z' for 'demoDoubleAr[3]' (bad value/bad value count)
007	29/07/2021 10:53:35.324	ser1/COM22/My Arduino	[Row 1] Writing variable demoBool=-5 (INT) completed.
008	29/07/2021 10:53:37.652	ser1/COM22/My Arduino	[Row 2] Writing variable demoChar=2 (INT) completed.
009	29/07/2021 10:53:39.596	ser1/COM22/My Arduino	[Row 3] Out of range value '-300' for 'demoUChar'
010	29/07/2021 10:53:40.591	ser1/COM22/My Arduino	[Row 4] Writing variable demoInt=4 (INT) completed.
011	29/07/2021 10:53:41.015	ser1/COM22/My Arduino	[Row 5] Writing variable demoUInt=5 (INT) completed.
012	29/07/2021 10:53:57.116	ser1/COM22/My Arduino	[Row 6] Out of range value '500000000000' for 'demoLong'
013	29/07/2021 10:54:07.704	ser1/COM22/My Arduino	[Row 6] Writing variable demoLong=50000000 (INT) completed.

```
Loop()   Autowrite on   No recorders active   ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 4483/4483
```

Fig. 3.11. Integer formatting.

Note:

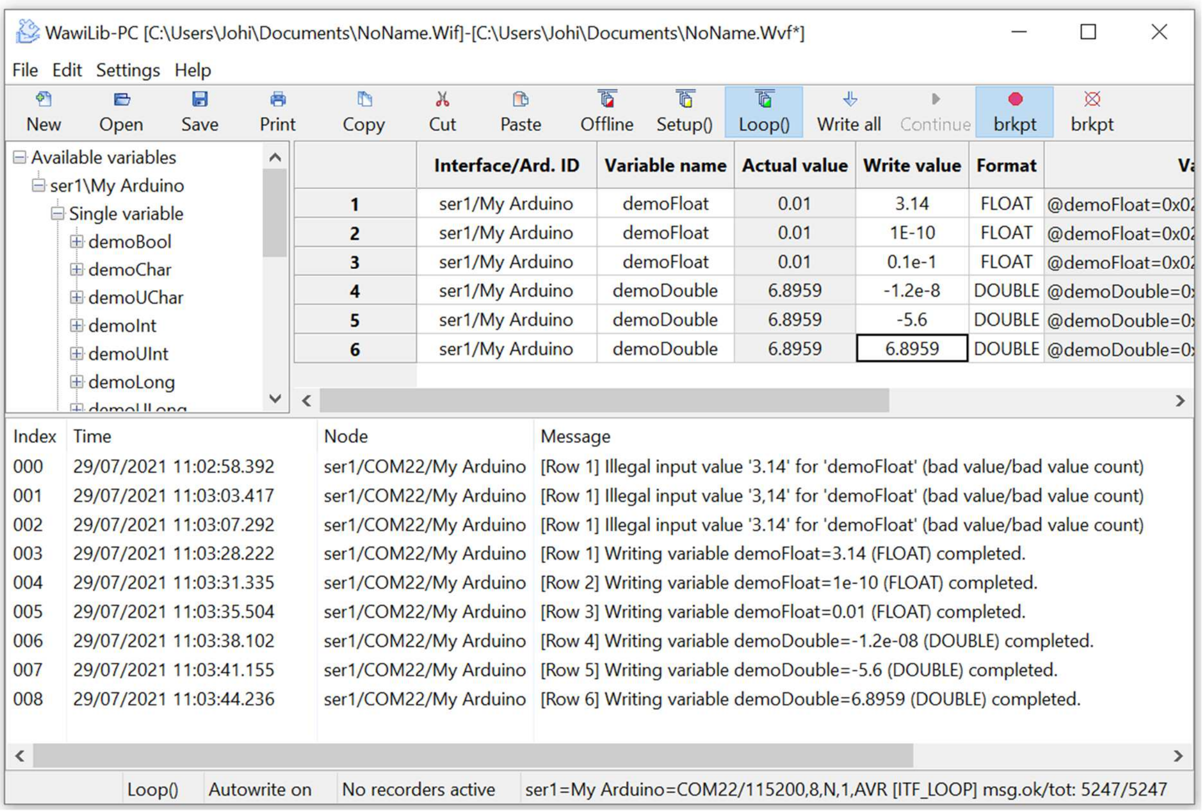
In the table above you see *demoFloat* represented as an INT and written with the value 78. On line 9 you can see the result if the 4-byte value is interpreted as a mantisse + exp value, that is the internal representation of a float. You get a floating point value of 1.09e-43!

If you tell WawiLib to visualise the variable *demoFloat*, it means that WawiLib has to take this 4 byte memory location an see it as a type indicated in the “format column” regardless of what you have defined in your Sketch.

3.5.4 Write floating point format (FLOAT / DOUBLE)

WawiLib supports different formats of floating-point variables. You can use scientific notation (1.25e-1) or standard notation (0.125).

Depending on the platform, the size of a double variable in the Arduino can be 4-byte (MEGA) or 8-byte (DUE). WawiLib is able to differentiate between the 2 floating point variable types. Therefore, if you connect a MEGA2560 to WawiLib, double and float formatting are identical (4 byte), but if you use a DUE, float will differ from double representation.


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
ser1\My Arduino
Single variable
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong

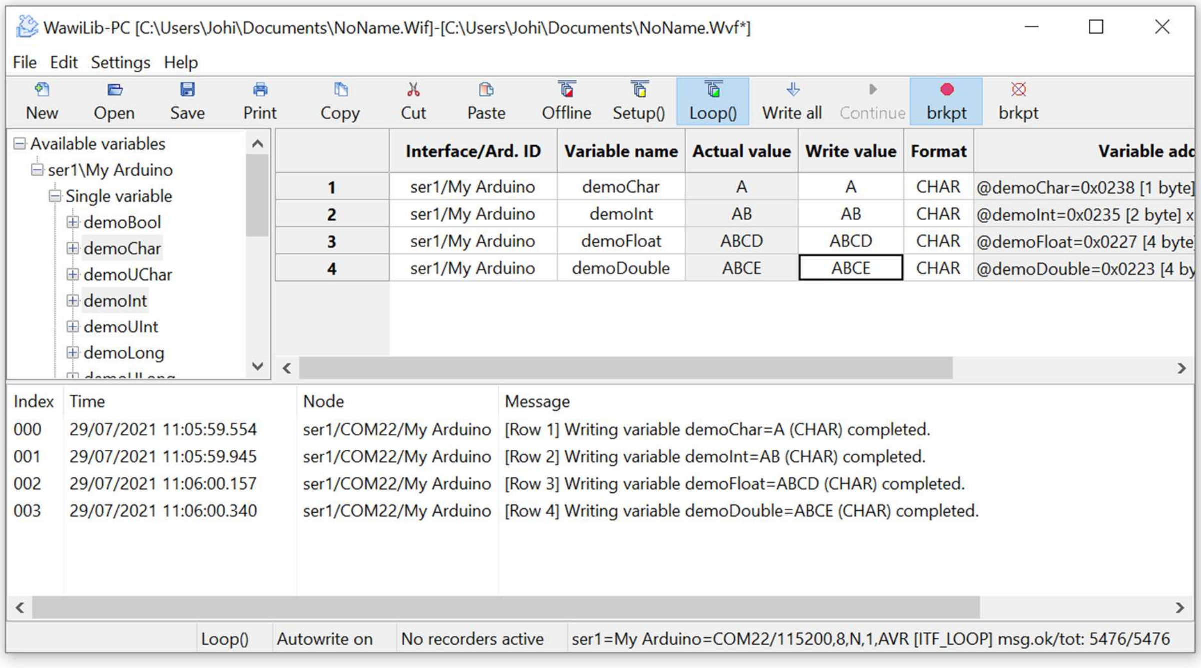
	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Va
1	ser1/My Arduino	demoFloat	0.01	3.14	FLOAT	@demoFloat=0x02
2	ser1/My Arduino	demoFloat	0.01	1E-10	FLOAT	@demoFloat=0x02
3	ser1/My Arduino	demoFloat	0.01	0.1e-1	FLOAT	@demoFloat=0x02
4	ser1/My Arduino	demoDouble	6.8959	-1.2e-8	DOUBLE	@demoDouble=0x
5	ser1/My Arduino	demoDouble	6.8959	-5.6	DOUBLE	@demoDouble=0x
6	ser1/My Arduino	demoDouble	6.8959	6.8959	DOUBLE	@demoDouble=0x

```
Index  Time                      Node                   Message
000    29/07/2021 11:02:58.392   ser1/COM22/My Arduino  [Row 1] Illegal input value '3.14' for 'demoFloat' (bad value/bad value count)
001    29/07/2021 11:03:03.417   ser1/COM22/My Arduino  [Row 1] Illegal input value '3,14' for 'demoFloat' (bad value/bad value count)
002    29/07/2021 11:03:07.292   ser1/COM22/My Arduino  [Row 1] Illegal input value '3.14' for 'demoFloat' (bad value/bad value count)
003    29/07/2021 11:03:28.222   ser1/COM22/My Arduino  [Row 1] Writing variable demoFloat=3.14 (FLOAT) completed.
004    29/07/2021 11:03:31.335   ser1/COM22/My Arduino  [Row 2] Writing variable demoFloat=1e-10 (FLOAT) completed.
005    29/07/2021 11:03:35.504   ser1/COM22/My Arduino  [Row 3] Writing variable demoFloat=0.01 (FLOAT) completed.
006    29/07/2021 11:03:38.102   ser1/COM22/My Arduino  [Row 4] Writing variable demoDouble=-1.2e-08 (DOUBLE) completed.
007    29/07/2021 11:03:41.155   ser1/COM22/My Arduino  [Row 5] Writing variable demoDouble=-5.6 (DOUBLE) completed.
008    29/07/2021 11:03:44.236   ser1/COM22/My Arduino  [Row 6] Writing variable demoDouble=6.8959 (DOUBLE) completed.

Loop()  Autowrite on  No recorders active  ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 5247/5247
```

Fig. 3.12. Floating point formatting.

3.5.5 Write character format (CHAR)


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
ser1\My Arduino
Single variable
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable adc
1	ser1/My Arduino	demoChar	A	A	CHAR	@demoChar=0x0238 [1 byte]
2	ser1/My Arduino	demoInt	AB	AB	CHAR	@demoInt=0x0235 [2 byte] x
3	ser1/My Arduino	demoFloat	ABCD	ABCD	CHAR	@demoFloat=0x0227 [4 byte
4	ser1/My Arduino	demoDouble	ABCE	ABCE	CHAR	@demoDouble=0x0223 [4 by

```
Index  Time                      Node                   Message
000    29/07/2021 11:05:59.554   ser1/COM22/My Arduino  [Row 1] Writing variable demoChar=A (CHAR) completed.
001    29/07/2021 11:05:59.945   ser1/COM22/My Arduino  [Row 2] Writing variable demoInt=AB (CHAR) completed.
002    29/07/2021 11:06:00.157   ser1/COM22/My Arduino  [Row 3] Writing variable demoFloat=ABCD (CHAR) completed.
003    29/07/2021 11:06:00.340   ser1/COM22/My Arduino  [Row 4] Writing variable demoDouble=ABCE (CHAR) completed.

Loop()  Autowrite on  No recorders active  ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 5476/5476
```

Fig. 3.12. Char formatting.

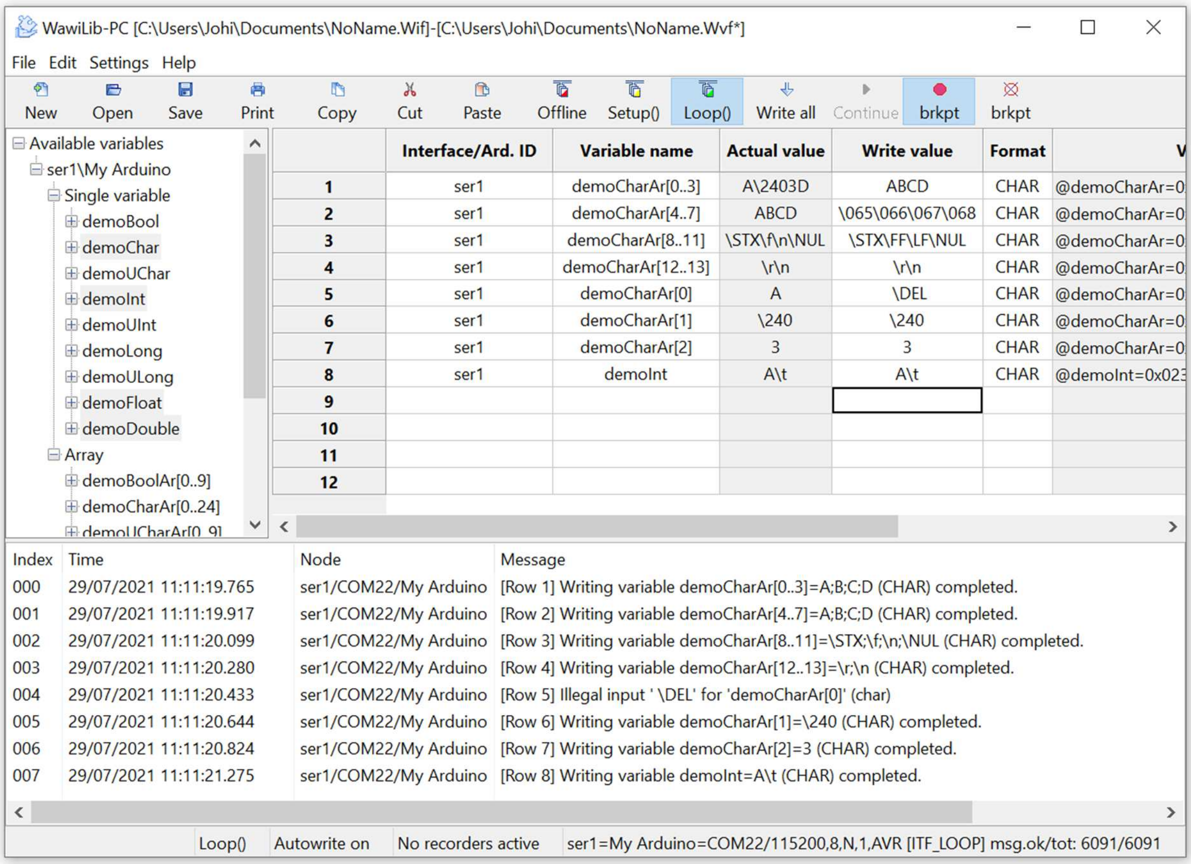
Characters input supports multiple input formats. Variables larger than 1 character can be written. The number of characters must match the length of the variable. You can see the size of the variable in the last column of the grid. Writing to arrays of non CHAR data types is supported.

WawiLib supports character input in different formats:

- Literal value, e.g. ABCD.
- ASCI code for all digits, e.g. \065,\066,\067,\068 (=ABCD).
- Code for special characters in the range 0..31, e.g. \ETX,\NUL (capital case, see ASCII table below).
- C Escape sequences, e.g. \r,\n,\t (lower case).

Some values of the ASCII table cannot be displayed as a letter, because they have a special function. WawiLib character output has the following precedence for displaying character values:

- Literal value, e.g. A.
- C Escape sequences, e.g. \r.
- Operand code for special characters in the range 0..31, e.g. \STX\ETX\NUL (! capital case).
- WawiLib ASCI code for all digits, e.g. \065.

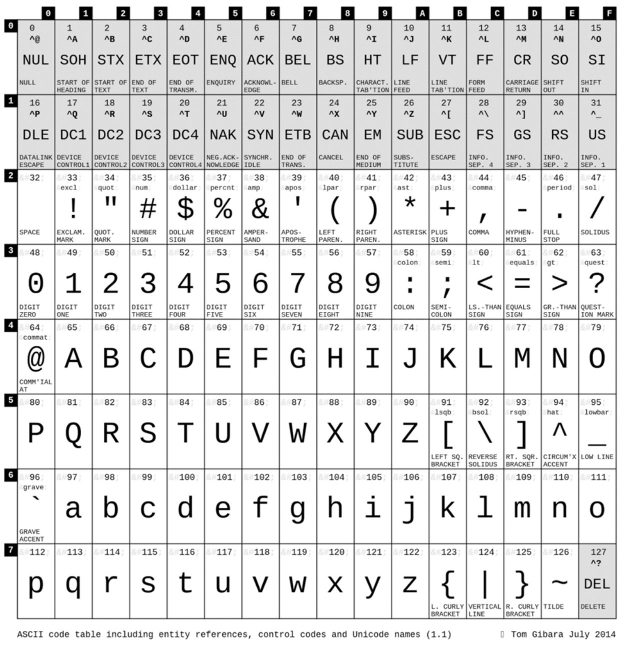


	Interface/Ard. ID	Variable name	Actual value	Write value	Format	V
1	ser1	demoCharAr[0..3]	A\2403D	ABCD	CHAR	@demoCharAr=0
2	ser1	demoCharAr[4..7]	ABCD	\065\066\067\068	CHAR	@demoCharAr=0
3	ser1	demoCharAr[8..11]	\STX\f\n\NUL	\STX\FF\LF\NUL	CHAR	@demoCharAr=0
4	ser1	demoCharAr[12..13]	\r\n	\r\n	CHAR	@demoCharAr=0
5	ser1	demoCharAr[0]	A	\DEL	CHAR	@demoCharAr=0
6	ser1	demoCharAr[1]	\240	\240	CHAR	@demoCharAr=0
7	ser1	demoCharAr[2]	3	3	CHAR	@demoCharAr=0
8	ser1	demoInt	A\t	A\t	CHAR	@demoInt=0x023
9						
10						
11						
12						

```
Index  Time                     Node                    Message
000    29/07/2021 11:11:19.765  ser1/COM22/My Arduino   [Row 1] Writing variable demoCharAr[0..3]=A;B;C;D (CHAR) completed.
001    29/07/2021 11:11:19.917  ser1/COM22/My Arduino   [Row 2] Writing variable demoCharAr[4..7]=A;B;C;D (CHAR) completed.
002    29/07/2021 11:11:20.099  ser1/COM22/My Arduino   [Row 3] Writing variable demoCharAr[8..11]=\STX;\f;\n;\NUL (CHAR) completed.
003    29/07/2021 11:11:20.280  ser1/COM22/My Arduino   [Row 4] Writing variable demoCharAr[12..13]=\r;\n (CHAR) completed.
004    29/07/2021 11:11:20.433  ser1/COM22/My Arduino   [Row 5] Illegal input ' \DEL' for 'demoCharAr[0]' (char)
005    29/07/2021 11:11:20.644  ser1/COM22/My Arduino   [Row 6] Writing variable demoCharAr[1]=\240 (CHAR) completed.
006    29/07/2021 11:11:20.824  ser1/COM22/My Arduino   [Row 7] Writing variable demoCharAr[2]=3 (CHAR) completed.
007    29/07/2021 11:11:21.275  ser1/COM22/My Arduino   [Row 8] Writing variable demoInt=A\t (CHAR) completed.
```

```
Loop()  Autowrite on  No recorders active  ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 6091/6091
```

Fig. 3.12. Char formatting of arrays.



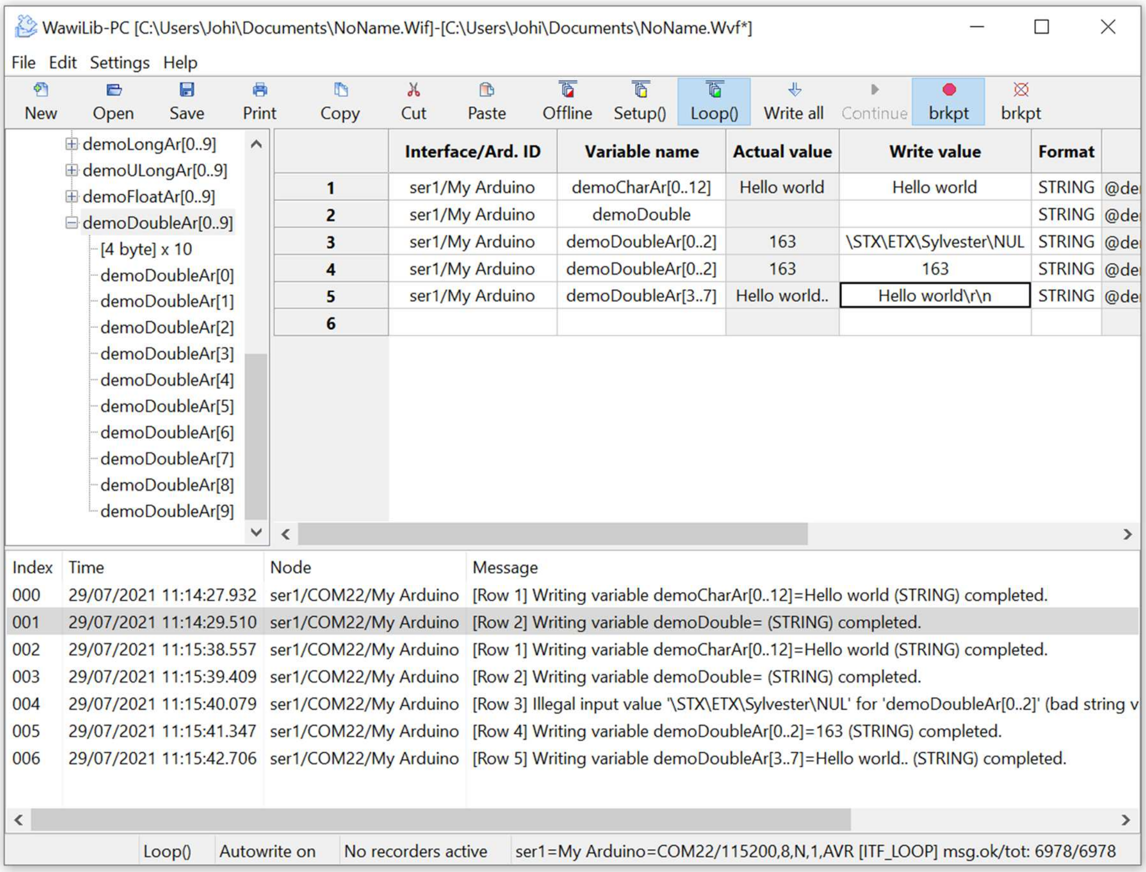
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0 ^@ NUL NULL	1 ^A SOH START OF HEADING	2 ^B STX START OF TEXT	3 ^C ETX END OF TEXT	4 ^D EOT END OF TRANSM.	5 ^E ENQ ENQUIRY	6 ^F ACK ACKNOWL-EDGE	7 ^G BEL BELL	8 ^H BS BACKSP.	9 ^I HT CHARACT. TAB'TION	10 ^J LF LINE FEED	11 ^K VT LINE TAB'TION	12 ^L FF FORM FEED	13 ^M CR CARRIAGE RETURN	14 ^N SO SHIFT OUT	15 ^O SI SHIFT IN
1	16 ^P DLE DATALINK ESCAPE	17 ^Q DC1 DEVICE CONTROL1	18 ^R DC2 DEVICE CONTROL2	19 ^S DC3 DEVICE CONTROL3	20 ^T DC4 DEVICE CONTROL4	21 ^U NAK NEG.ACK-NOWLEDGE	22 ^V SYN SYNCHR. IDLE	23 ^W ETB END OF TRANS.	24 ^X CAN CANCEL	25 ^Y EM END OF MEDIUM	26 ^Z SUB SUBS-TITUTE	27 ^[ESC ESCAPE	28 ^\ FS INFO. SEP. 4	29 ^] GS INFO. SEP. 3	30 ^^ RS INFO. SEP. 2	31 ^_ US INFO. SEP. 1
2	&#32; SPACE	&#33; &excl; ! EXCLAM. MARK	&#34; &quot; " QUOT. MARK	&#35; &num; # NUMBER SIGN	&#36; &dollar; $ DOLLAR SIGN	&#37; &percnt; % PERCENT SIGN	&#38; &amp; & AMPER-SAND	&#39; &apos; ' APOS-TROPHE	&#40; &lpar; (LEFT PAREN.	&#41; &rpar;) RIGHT PAREN.	&#42; &ast; * ASTERISK	&#43; &plus; + PLUS SIGN	&#44; &comma; , COMMA	&#45; - HYPHEN-MINUS	&#46; &period; . FULL STOP	&#47; &sol; / SOLIDUS
3	&#48; 0 DIGIT ZERO	&#49; 1 DIGIT ONE	&#50; 2 DIGIT TWO	&#51; 3 DIGIT THREE	&#52; 4 DIGIT FOUR	&#53; 5 DIGIT FIVE	&#54; 6 DIGIT SIX	&#55; 7 DIGIT SEVEN	&#56; 8 DIGIT EIGHT	&#57; 9 DIGIT NINE	&#58; &colon; : COLON	&#59; &semi; ; SEMI-COLON	&#60; &lt; < LS.-THAN SIGN	&#61; &equals; = EQUALS SIGN	&#62; &gt; > GR.-THAN SIGN	&#63; &quest; ? QUEST-ION MARK
4	&#64; &commat; @ COMM'IAL AT	&#65; A	&#66; B	&#67; C	&#68; D	&#69; E	&#70; F	&#71; G	&#72; H	&#73; I	&#74; J	&#75; K	&#76; L	&#77; M	&#78; N	&#79; O
5	&#80; P	&#81; Q	&#82; R	&#83; S	&#84; T	&#85; U	&#86; V	&#87; W	&#88; X	&#89; Y	&#90; Z	&#91; &lsqb; [LEFT SQ. BRACKET	&#92; &bsol; \ REVERSE SOLIDUS	&#93; &rsqb;] RT. SQR. BRACKET	&#94; &hat; ^ CIRCUM'X ACCENT	&#95; &lowbar; _ LOW LINE
6	&#96; &grave; ` GRAVE ACCENT	&#97; a	&#98; b	&#99; c	&#100; d	&#101; e	&#102; f	&#103; g	&#104; h	&#105; i	&#106; j	&#107; k	&#108; l	&#109; m	&#110; n	&#111; o
7	&#112; p	&#113; q	&#114; r	&#115; s	&#116; t	&#117; u	&#118; v	&#119; w	&#120; x	&#121; y	&#122; z	&#123; { L. CURLY BRACKET	&#124; | VERTICAL LINE	&#125; } R. CURLY BRACKET	&#126; ~ TILDE	127 ^? DEL DELETE

ASCII code table including entity references, control codes and Unicode names (1.1) Tom Gibara July 2014

Fig. 3.13. Ascii table.

These values are represented in the format \... where ... is an abbreviation of the function (example \STX). Alternatively can be used \ddd for values that have no symbol nor a special function (typically values above 127 (example \145 or \064):

3.5.6 Write string format (STRING)



WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	
1	ser1/My Arduino	demoCharAr[0..12]	Hello world	Hello world	STRING	@de
2	ser1/My Arduino	demoDouble			STRING	@de
3	ser1/My Arduino	demoDoubleAr[0..2]	163	\STX\ETX\Sylvester\NUL	STRING	@de
4	ser1/My Arduino	demoDoubleAr[0..2]	163	163	STRING	@de
5	ser1/My Arduino	demoDoubleAr[3..7]	Hello world..	Hello world\r\n	STRING	@de
6						

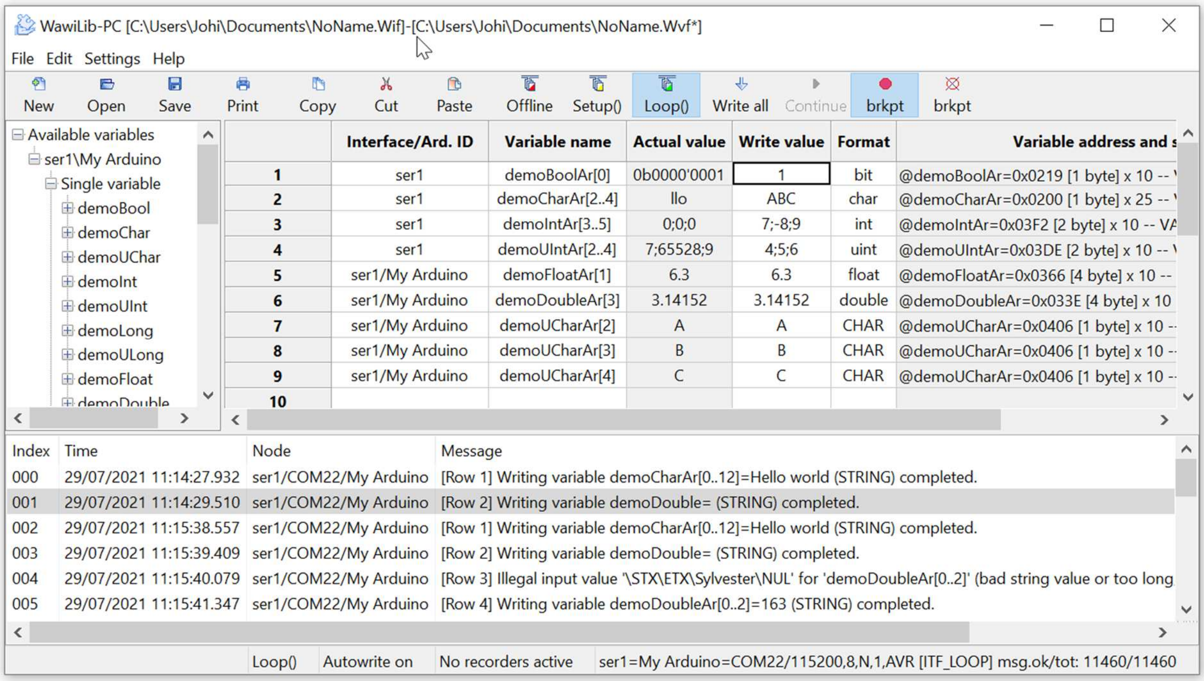
```
Index Time                     Node                  Message
000   29/07/2021 11:14:27.932  ser1/COM22/My Arduino [Row 1] Writing variable demoCharAr[0..12]=Hello world (STRING) completed.
001   29/07/2021 11:14:29.510  ser1/COM22/My Arduino [Row 2] Writing variable demoDouble= (STRING) completed.
002   29/07/2021 11:15:38.557  ser1/COM22/My Arduino [Row 1] Writing variable demoCharAr[0..12]=Hello world (STRING) completed.
003   29/07/2021 11:15:39.409  ser1/COM22/My Arduino [Row 2] Writing variable demoDouble= (STRING) completed.
004   29/07/2021 11:15:40.079  ser1/COM22/My Arduino [Row 3] Illegal input value '\STX\ETX\Sylvester\NUL' for 'demoDoubleAr[0..2]' (bad string v
005   29/07/2021 11:15:41.347  ser1/COM22/My Arduino [Row 4] Writing variable demoDoubleAr[0..2]=163 (STRING) completed.
006   29/07/2021 11:15:42.706  ser1/COM22/My Arduino [Row 5] Writing variable demoDoubleAr[3..7]=Hello world.. (STRING) completed.
```

```
Loop()  Autowrite on  No recorders active  ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 6978/6978
```

Fig. 3.14. Char formatting of strings (zero terminated char arrays).

Strings are typically arrays of characters, they will be treated in the next chapter. However, if you have a 4-byte variable, you can write a 3-character string + tailing 0 to this variable as on line 4. Characters in a string that are non digit will be displayed as '.'. If you want to see all characters in detail, switch to 'char' format.

3.5.7 Writing arrays of values


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
Available variables
ser1\My Arduino
Single variable
demoBool
demoChar
demoUChar
demoInt
demoUInt
demoLong
demoULong
demoFloat
demoDouble

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	Variable address and s
1	ser1	demoBoolAr[0]	0b0000'0001	1	bit	@demoBoolAr=0x0219 [1 byte] x 10 -- \
2	ser1	demoCharAr[2..4]	llo	ABC	char	@demoCharAr=0x0200 [1 byte] x 25 -- \
3	ser1	demoIntAr[3..5]	0;0;0	7;-8;9	int	@demoIntAr=0x03F2 [2 byte] x 10 -- VA
4	ser1	demoUIntAr[2..4]	7;65528;9	4;5;6	uint	@demoUIntAr=0x03DE [2 byte] x 10 -- \
5	ser1/My Arduino	demoFloatAr[1]	6.3	6.3	float	@demoFloatAr=0x0366 [4 byte] x 10 --
6	ser1/My Arduino	demoDoubleAr[3]	3.14152	3.14152	double	@demoDoubleAr=0x033E [4 byte] x 10
7	ser1/My Arduino	demoUCharAr[2]	A	A	CHAR	@demoUCharAr=0x0406 [1 byte] x 10 --
8	ser1/My Arduino	demoUCharAr[3]	B	B	CHAR	@demoUCharAr=0x0406 [1 byte] x 10 --
9	ser1/My Arduino	demoUCharAr[4]	C	C	CHAR	@demoUCharAr=0x0406 [1 byte] x 10 --
10						

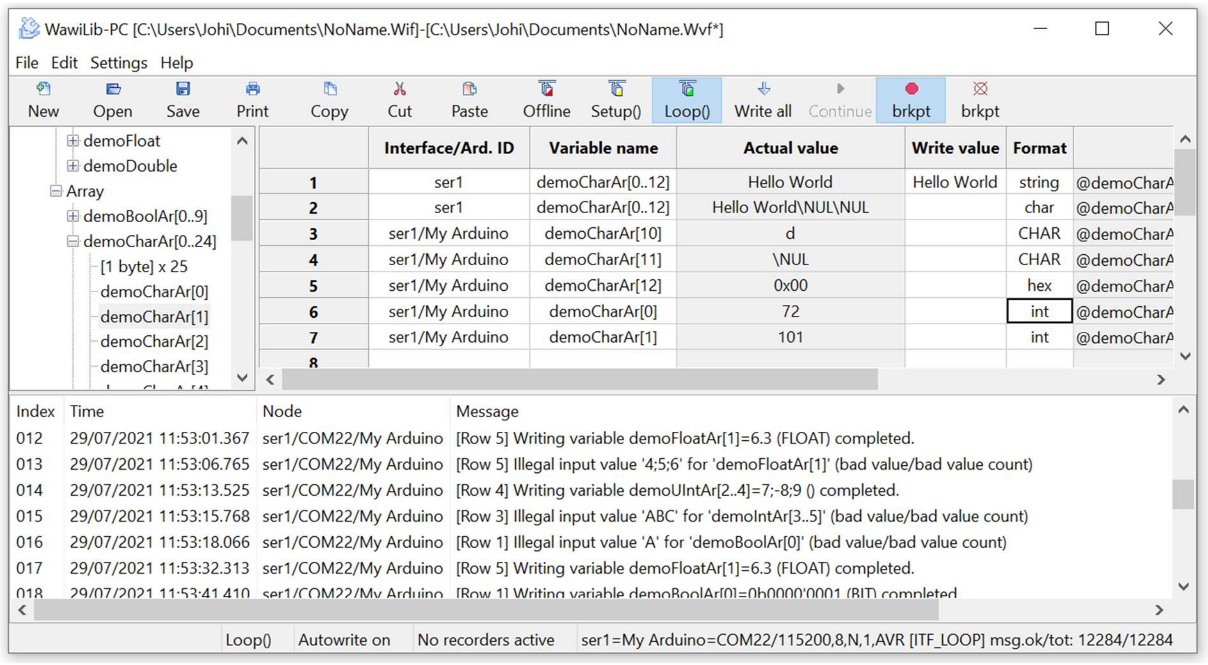
```
Index Time                    Node                    Message
000   29/07/2021 11:14:27.932 ser1/COM22/My Arduino   [Row 1] Writing variable demoCharAr[0..12]=Hello world (STRING) completed.
001   29/07/2021 11:14:29.510 ser1/COM22/My Arduino   [Row 2] Writing variable demoDouble= (STRING) completed.
002   29/07/2021 11:15:38.557 ser1/COM22/My Arduino   [Row 1] Writing variable demoCharAr[0..12]=Hello world (STRING) completed.
003   29/07/2021 11:15:39.409 ser1/COM22/My Arduino   [Row 2] Writing variable demoDouble= (STRING) completed.
004   29/07/2021 11:15:40.079 ser1/COM22/My Arduino   [Row 3] Illegal input value '\STX\ETX\Sylvester\NUL' for 'demoDoubleAr[0..2]' (bad string value or too long
005   29/07/2021 11:15:41.347 ser1/COM22/My Arduino   [Row 4] Writing variable demoDoubleAr[0..2]=163 (STRING) completed.
```

```
Loop()   Autowrite on   No recorders active   ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 11460/11460
```

Fig. 3.15. Writing arrays of values.

Array write values need to be separated in default by a ‘;’ (except for char arrays and strings). On line 2 in the table above, you can see an example of a write to 3 array elements [2,3,4] using a single value ABC in the write value column.

String variables can be written using plain text as you can see in the table below. Do not forget that the receiving array must also be able to contain the 0 character at the end of the string. WawiLib will automatically add the 0 character when writing to a variable represented in string format.


WawiLib-PC [C:\Users\Johi\Documents\NoName.Wif]-[C:\Users\Johi\Documents\NoName.Wvf*]
File Edit Settings Help
New Open Save Print Copy Cut Paste Offline Setup() Loop() Write all Continue brkpt brkpt
demoFloat
demoDouble
Array
demoBoolAr[0..9]
demoCharAr[0..24]
[1 byte] x 25
demoCharAr[0]
demoCharAr[1]
demoCharAr[2]
demoCharAr[3]

	Interface/Ard. ID	Variable name	Actual value	Write value	Format	
1	ser1	demoCharAr[0..12]	Hello World	Hello World	string	@demoCharA
2	ser1	demoCharAr[0..12]	Hello World\NUL\NUL		char	@demoCharA
3	ser1/My Arduino	demoCharAr[10]	d		CHAR	@demoCharA
4	ser1/My Arduino	demoCharAr[11]	\NUL		CHAR	@demoCharA
5	ser1/My Arduino	demoCharAr[12]	0x00		hex	@demoCharA
6	ser1/My Arduino	demoCharAr[0]	72		int	@demoCharA
7	ser1/My Arduino	demoCharAr[1]	101		int	@demoCharA
8						

```
Index Time                    Node                    Message
012   29/07/2021 11:53:01.367 ser1/COM22/My Arduino   [Row 5] Writing variable demoFloatAr[1]=6.3 (FLOAT) completed.
013   29/07/2021 11:53:06.765 ser1/COM22/My Arduino   [Row 5] Illegal input value '4;5;6' for 'demoFloatAr[1]' (bad value/bad value count)
014   29/07/2021 11:53:13.525 ser1/COM22/My Arduino   [Row 4] Writing variable demoUIntAr[2..4]=7;-8;9 () completed.
015   29/07/2021 11:53:15.768 ser1/COM22/My Arduino   [Row 3] Illegal input value 'ABC' for 'demoIntAr[3..5]' (bad value/bad value count)
016   29/07/2021 11:53:18.066 ser1/COM22/My Arduino   [Row 1] Illegal input value 'A' for 'demoBoolAr[0]' (bad value/bad value count)
017   29/07/2021 11:53:32.313 ser1/COM22/My Arduino   [Row 5] Writing variable demoFloatAr[1]=6.3 (FLOAT) completed.
018   29/07/2021 11:53:41.410 ser1/COM22/My Arduino   [Row 1] Writing variable demoBoolAr[0]=0b0000'0001 (BIT) completed.
```

```
Loop()  Autowrite on  No recorders active  ser1=My Arduino=COM22/115200,8,N,1,AVR [ITF_LOOP] msg.ok/tot: 12284/12284
```

Fig. 3.16. Writing arrays of values.

4 Further reading

This demo demonstrates how to read and write variables with WawiLib. Variables of different sizes can be read and written in various formats. Arrays and parts of arrays can also be read and written. Special attention is required when reading and writing strings as they have to be zero terminated. Char formats are supported in multiple forms.

Recording of variables can be executed “on change”, “on timer” or both. Recording can also be done with one file per hour or per day to make the generated files more manageable. WawiLib supports links via the USB programming interface, WiFi, cabled Ethernet, hardware serial, software serial and via USB to serial converters.

I hope you enjoyed this demo. Visit us on www.sylvestersolutions.com for more demos.