AVRPascalDbg

Manual

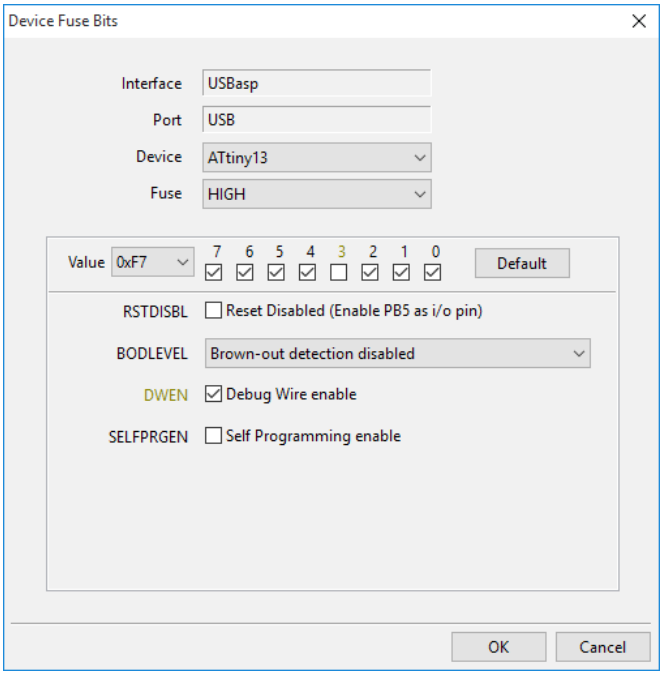
version 01/03/2024

1. Introduction

AVRPasDbg is a debugger for AVRPascal. It is distributed as a plug-in for the application (*avrpasdbg.dll* on Windows, *libavrpasdbg.so* on Linux) along with a license file (**.lic*). These files should be located in the AVRPascal binaries directory, i.e.:

- for Windows: *c:\Program Files\AVRPascal\bin\win64*[1]
- for Linux: */opt/AVRPascal/bin/linux64*

AVRPasDbg uses the DebugWire (DW) protocol to communicate with microcontrollers. To work with AVRPasDbg, the microcontroller must have the DWEN fuse bit enabled. This can be done, for example, using the UABasp programmer within AVRPascal. After turning on DWEN, the programmer should be disconnected[2].



1 The folder *c:\Program Files\AVRPascal* is only suggested during installation, the user can choose another one.

2 Enabling the DebugWire protocol using the DWEN fuse-bit prevents subsequent programming using USBasp. Therefore, dedicating a specific microcontroller only for debugging code seems like a good solution.

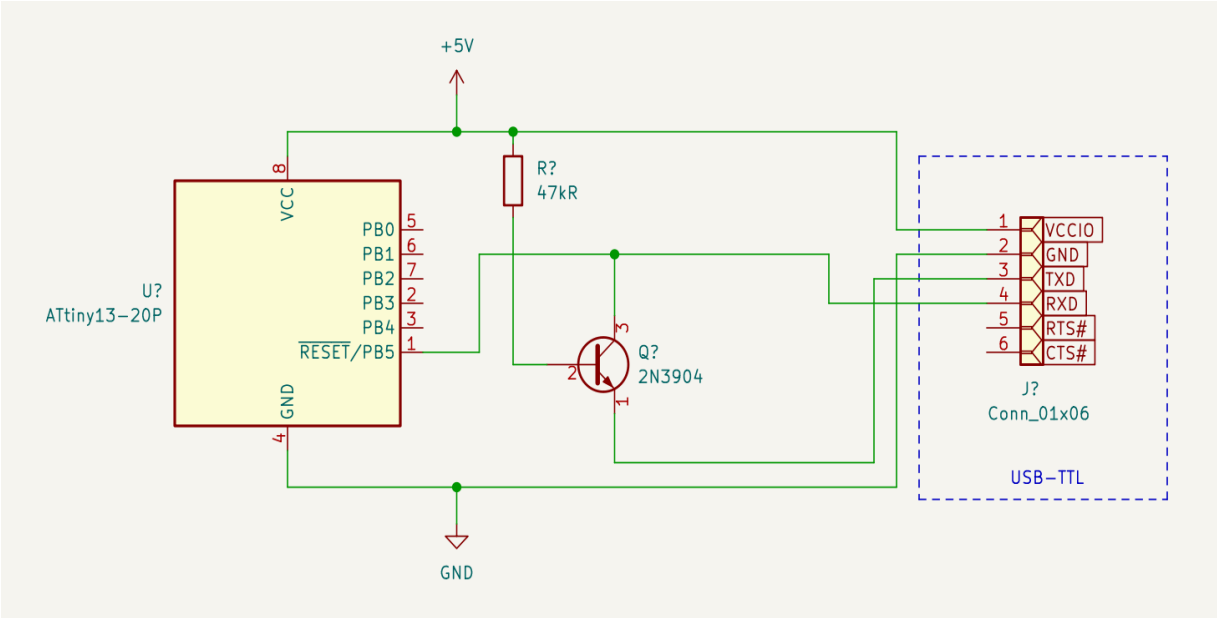
2. Adapter

AVRPasDbg communicates with the microcontroller using a USB-UART converter, which must be installed in the operating system as a virtual serial port (VCP). AVRPasDbg works with popular converters enabling the use of non-standard data transfer rates (baud-rates):

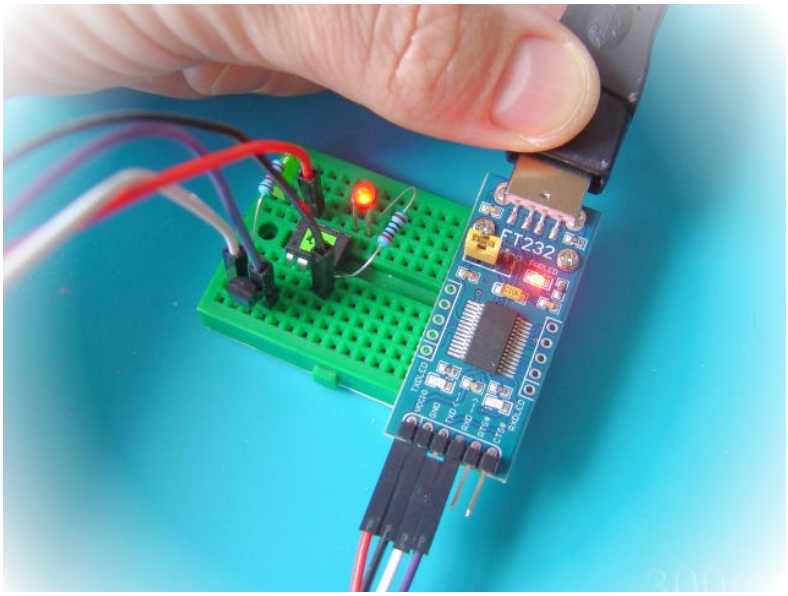
USB-UART Converter	Manufacturer	VID - PID
FT232	Waveshare	0x0403 - 0x6001
PL2303	Prolific	0x067b - 0x2303
CH340 (HL-340)	Quinheng	0x1a86 - 0x7523

For 64-bit Windows systems, using the above converters might require the installation of additional VCP drivers. These drivers are usually available for download from the manufacturers' websites:

Waveshare: https://www.waveshare.com
Prolific: https://www.prolific.com.tw
Quinheng: https://wch-ic.com

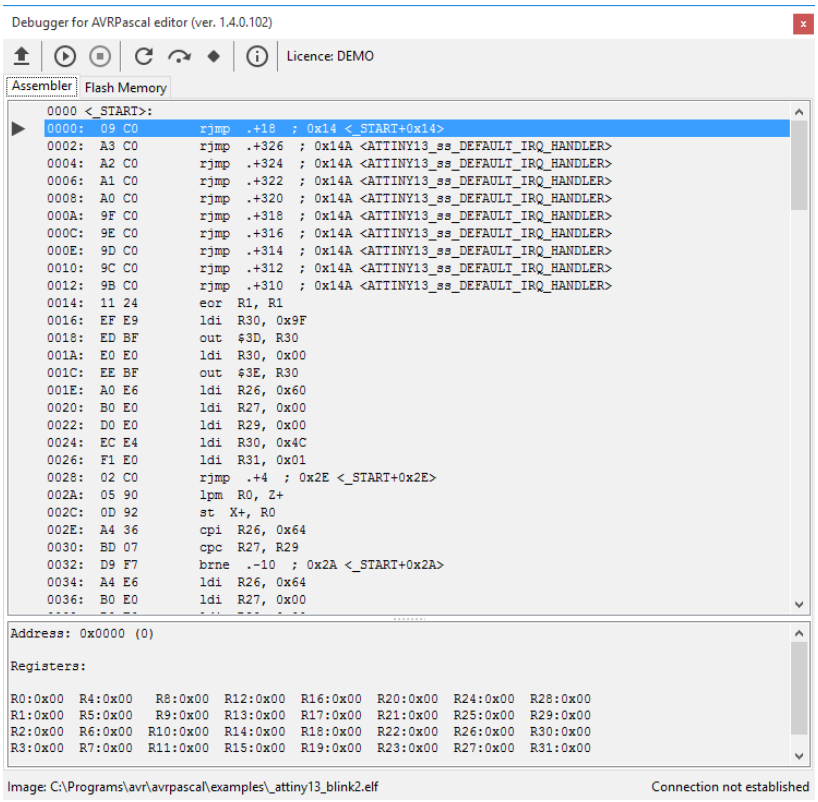


The VCC and GND outputs of the adapter should be connected to the corresponding VCC and GND pins of the microcontroller. The RXD output should be connected to the RESET pin through a 47 kΩ pull-up resistor, while the TXD output should be connected to the RESET pin through a 2N3904 transistor. The connection circuit (for example, with an ATtiny13 microcontroller) is shown above. The connection can be made on a breadboard and expanded with other elements useful during debugging, such as an LED.

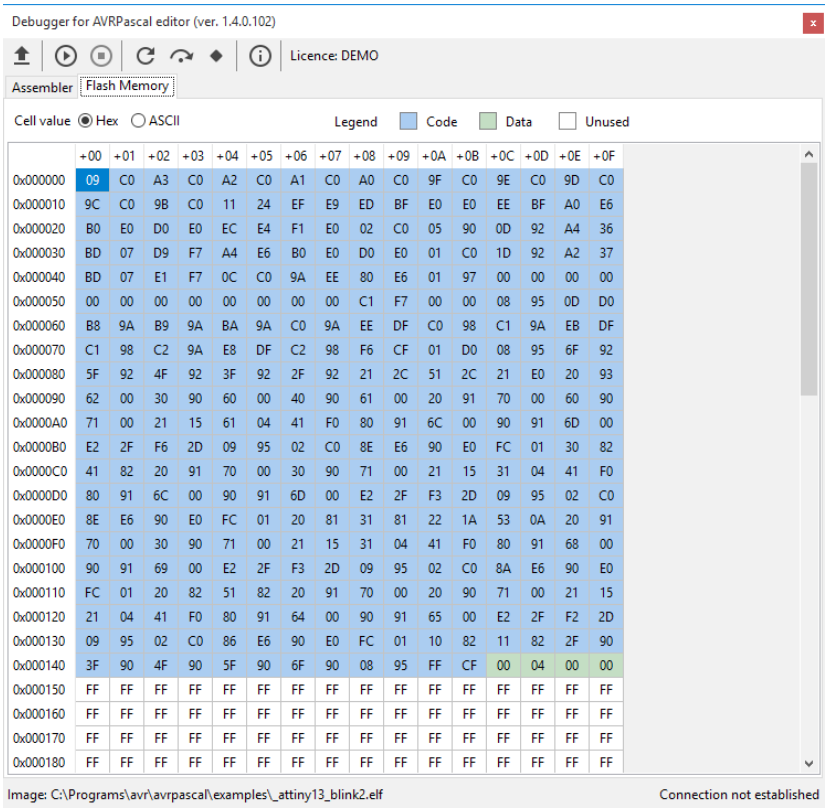


3. Debugger window

AVRPascal offers a debugger window accessible through the *View->Debug Window* menu. This window displays the loaded flash memory image from the *.elf file. The Assembler tab shows both the disassembled memory image and the current contents of the microcontroller registers. If a line of assembly code corresponds to a specific line of Pascal source code, an icon will appear next to the corresponding address in the disassembled view.



The *Flash Memory* tab displays the loaded flash memory image, differentiating code (blue background) and data (green background) sections. You can view memory bytes in either hexadecimal mode or as ASCII characters.

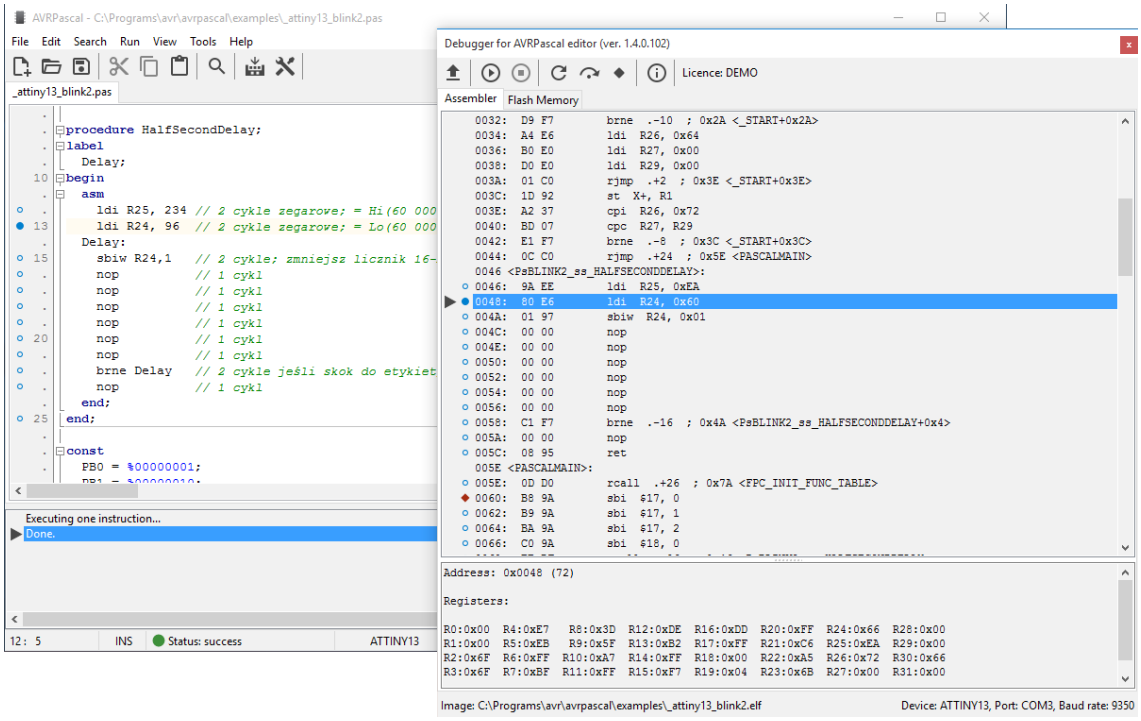


	+00	+01	+02	+03	+04	+05	+06	+07	+08	+09	+0A	+0B	+0C	+0D	+0E	+0F
0x000000	09	C0	A3	C0	A2	C0	A1	C0	A0	C0	9F	C0	9E	C0	9D	C0
0x000010	9C	C0	9B	C0	11	24	EF	E9	ED	BF	E0	E0	EE	BF	A0	E6
0x000020	B0	E0	D0	E0	EC	E4	F1	E0	02	C0	05	90	0D	92	A4	36
0x000030	BD	07	D9	F7	A4	E6	B0	E0	D0	E0	01	C0	1D	92	A2	37
0x000040	BD	07	E1	F7	0C	C0	9A	EE	80	E6	01	97	00	00	00	00
0x000050	00	00	00	00	00	00	00	00	C1	F7	00	00	08	95	0D	D0
0x000060	B8	9A	B9	9A	BA	9A	C0	9A	EE	DF	C0	98	C1	9A	EB	DF
0x000070	C1	98	C2	9A	E8	DF	C2	98	F6	CF	01	D0	08	95	6F	92
0x000080	5F	92	4F	92	3F	92	2F	92	21	2C	51	2C	21	E0	20	93
0x000090	62	00	30	90	60	00	40	90	61	00	20	91	70	00	60	90
0x0000A0	71	00	21	15	61	04	41	F0	80	91	6C	00	90	91	6D	00
0x0000B0	E2	2F	F6	2D	09	95	02	C0	8E	E6	90	E0	FC	01	30	82
0x0000C0	41	82	20	91	70	00	30	90	71	00	21	15	31	04	41	F0
0x0000D0	80	91	6C	00	90	91	6D	00	E2	2F	F3	2D	09	95	02	C0
0x0000E0	8E	E6	90	E0	FC	01	20	81	31	81	22	1A	53	0A	20	91
0x0000F0	70	00	30	90	71	00	21	15	31	04	41	F0	80	91	68	00
0x000100	90	91	69	00	E2	2F	F3	2D	09	95	02	C0	8A	E6	90	E0
0x000110	FC	01	20	82	51	82	20	91	70	00	20	90	71	00	21	15
0x000120	21	04	41	F0	80	91	64	00	90	91	65	00	E2	2F	F2	2D
0x000130	09	95	02	C0	86	E6	90	E0	FC	01	10	82	11	82	2F	90
0x000140	3F	90	4F	90	5F	90	6F	90	08	95	FF	CF	00	04	00	00
0x000150	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x000160	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x000170	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0x000180	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF

4. Debugging tools

The debugging tools are available in the toolbar and context menu. These tools include: *Upload* (- uploading the flash memory image into the microcontroller), *Run* (- starting the debugger), *Stop* (- stopping the debugger), *Reset* (- resetting the debugger), *One step* (- one step of the debugger), *Breakpoint* (- breakpoint of execution) and *Config data* (- microcontroller's configuration data). The results of each action and any error information are displayed in the AVRPascal *Messages* window

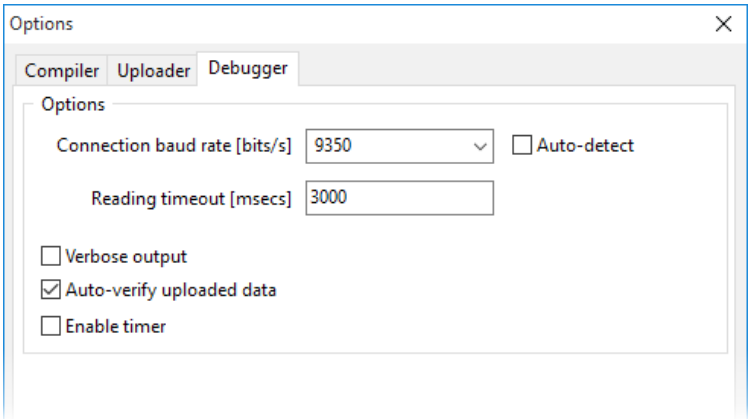
Before starting debugging, load the compiled program (referred to as the flash memory image) into the microcontroller using the *Upload* tool. This ensures that the microcontroller runs the exact code you compiled in AVRPascal. Then, start the debugger using the *Run* tool. If no breakpoint is set, the program will run continuously until you press *Stop*. If a breakpoint is set using the *Breakpoint* tool, the program will pause execution at that specific memory location. Only one breakpoint can be active at a time, and it's marked by an icon in the memory listing by icon. To resume an interrupted program, you can either use *Run* again or use *One Step*. The *One Step* tool executes the next instruction, moving the program to the following memory address. If the current execution point corresponds to a line of Pascal source code, an icon will appear in both the memory listing and the AVRPascal tab, next to the corresponding address.



The *Reset* tool is used to reset the microcontroller and return the program execution point to the beginning of the flash memory. In each case, the current content of the microcontroller registers is displayed in the lower panel of the Assembler tab. Finally, the *Config data* function shows the values of the microcontroller's fuse bits.

5. Debugger options

AVRPasDbg offers configuration options accessible in the *Debugger* tab of the AVRPascal *Options* window. Here, you can configure the connection with the USB-UART adapter, indicating the boud-rate of the transfer (in bits/second) or its automatic detection (*Auto-detect*). Additionally, you can define a read timeout, i.e. the time after which the connection will be interrupted if there is no data transferred.



These parameters are important because an incorrectly determined data transfer rate may result in data corruption and, consequently, errors or the inability of the debugger to function correctly[3].

The next parameters concern the ability of the microcontroller to use a timer during debugging (*Enable timer*), automatic verification of data loaded into the microcontroller's flash memory (*Auto-verify uploaded data*) and the level of detail of messages sent to the *Messages* window of AVRPascal (*Verbose output*).

6. *DEMO* version

AVRPasDbg is available on the website http://akarwowski.pl/ with a DEMO license (file *000B462305076250.lic*). This is a limited version of the debugger. Its only limitation is the maximum size of flash memory it can load into the microcontroller, which is 350 bytes.

Andrzej Karwowski

e-mail: ackarwow@gmail.com

3 The optimal baud-rate value is usually slightly higher than that determined automatically by the debugger.