



AVRPascalDbg

Manual

version 29/03/2026

1. Introduction

AVRPascalDbg is a tool for debugging and emulating AVR microcontrollers in the AVRPascal environment. It is distributed as a plug-in for the application (*avrpasdbg.dll* on Windows, *libavrpasdbg.so* on Linux) along with a licence file (**.lic*), placed in the system using a dedicated installer.¹ These files should be located in the AVRPascal binaries directory, i.e.:

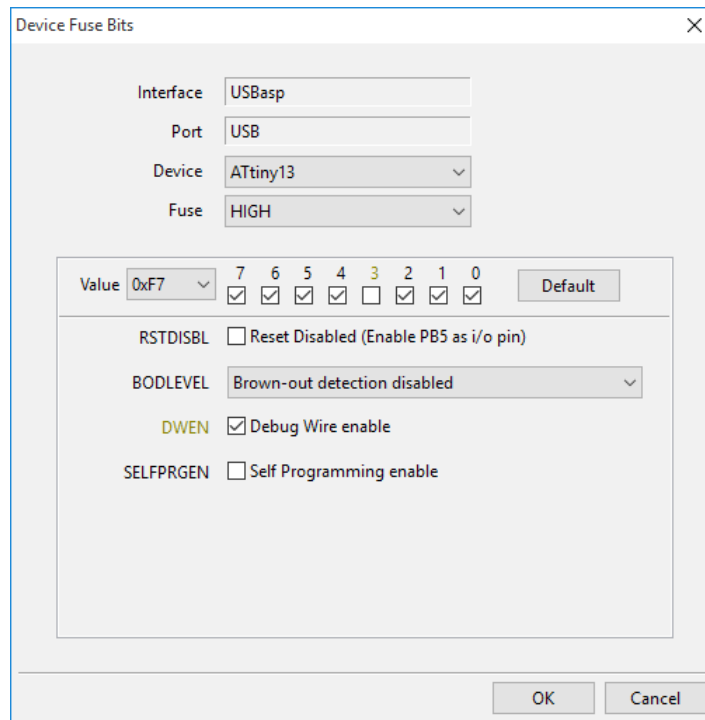
- for Windows: *c:\Program Files\AVRPascal\bin\win64*²
- for Linux: */opt/AVRPascal/bin/linux64*

2. Modes of operation

AVRPascalDbg operates in two modes: *Emulator* and *DebugWire*. The emulation mode does not require hardware (a physical microcontroller) and allows analysis of program logic. It supports programs compiled for any type of physical AVR microcontroller supported by the FPC compiler; however, compilation with AVRSIM as the virtual target device is recommended. In this case, additional diagnostic functionality is available, such as *write/writeln*, which display text in the AVRPascal *Messages* area³. The emulation mode does not support microcontroller peripherals and emulates only instruction logic.

In hardware debugging mode, AVRPascalDbg uses the DebugWire (DW) protocol to communicate with microcontrollers. To work with AVRPascalDbg, the microcontroller must have the DWEN fuse bit enabled. This can be done, for example, using the USBasp programmer within AVRPascal. After turning on DWEN, the programmer should be disconnected.⁴

-
- 1 The debugger version must be compatible with the version of the installed AVRPascal, i.e. the first two digits separated by a dot must be identical. In case of Linux systems, if both the AVRPascal editor and the AVRPascalDbg debugger are installed and there is a need to install a new version of the editor, you must first uninstall the debugger, otherwise the installer will inform you about the violation of dependencies between the packages. In Windows systems, you can easily overwrite previously installed programs, keeping in mind the version compatibility principle.
 - 2 The folder *C:\Program Files\AVRPascal* is only suggested during installation, the user can choose another one.
 - 3 For other target devices, the *write* and *writeln* functions produce no output.
 - 4 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.



3. Adapter (DebugWire mode)

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

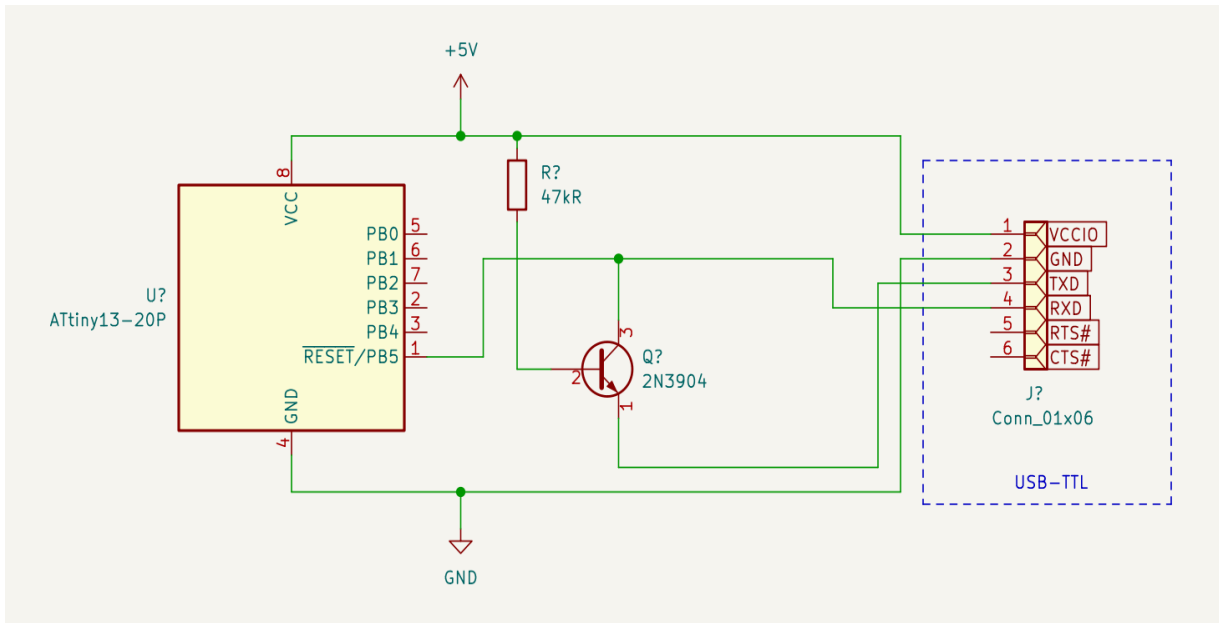
USB-UART Converter	Manufacturer	VID - PID
FT232	Waveshare	0x0403 - 0x6001
PL2303	Prolific	0x067b - 0x2303
CH340 (HL-340)	Quinheng	0x1a86 - 0x7523

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:

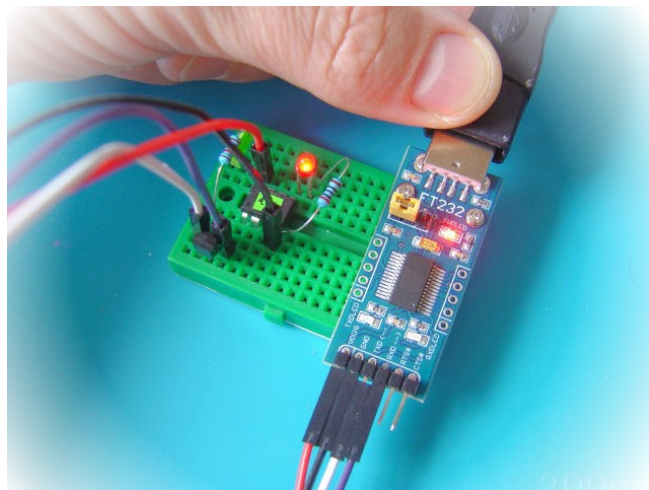
Waveshare: <https://www.waveshare.com>

Prolific: <https://www.prolific.com.tw>


Quinheng: <https://wch-ic.com>

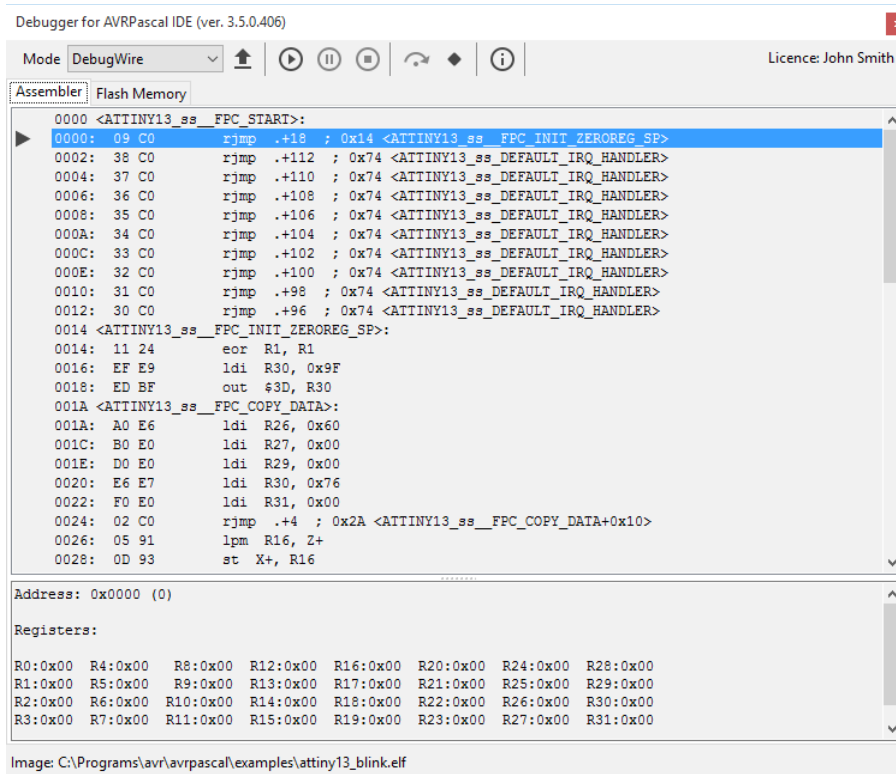


The VCC and GND outputs of the adapter should be connected to the corresponding VCC and GND pins of the microcontroller. The RXD output should be connected to the RESET pin through a 47 kΩ pull-up resistor, while the TXD output should be connected to the RESET pin through a 2N3904 transistor. The connection 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.

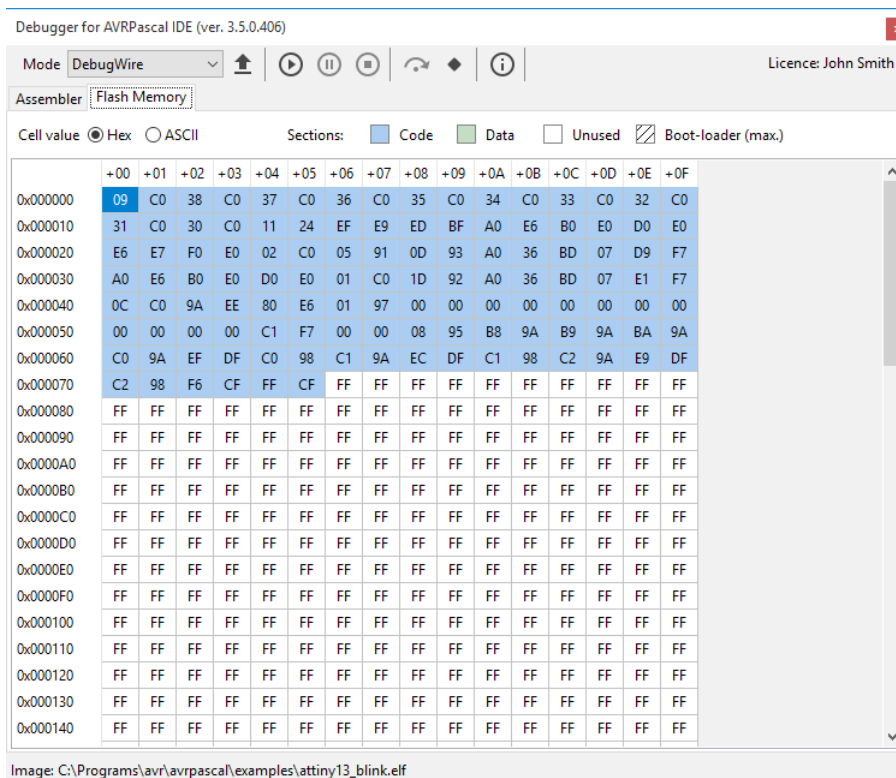


4. Debugger window

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.



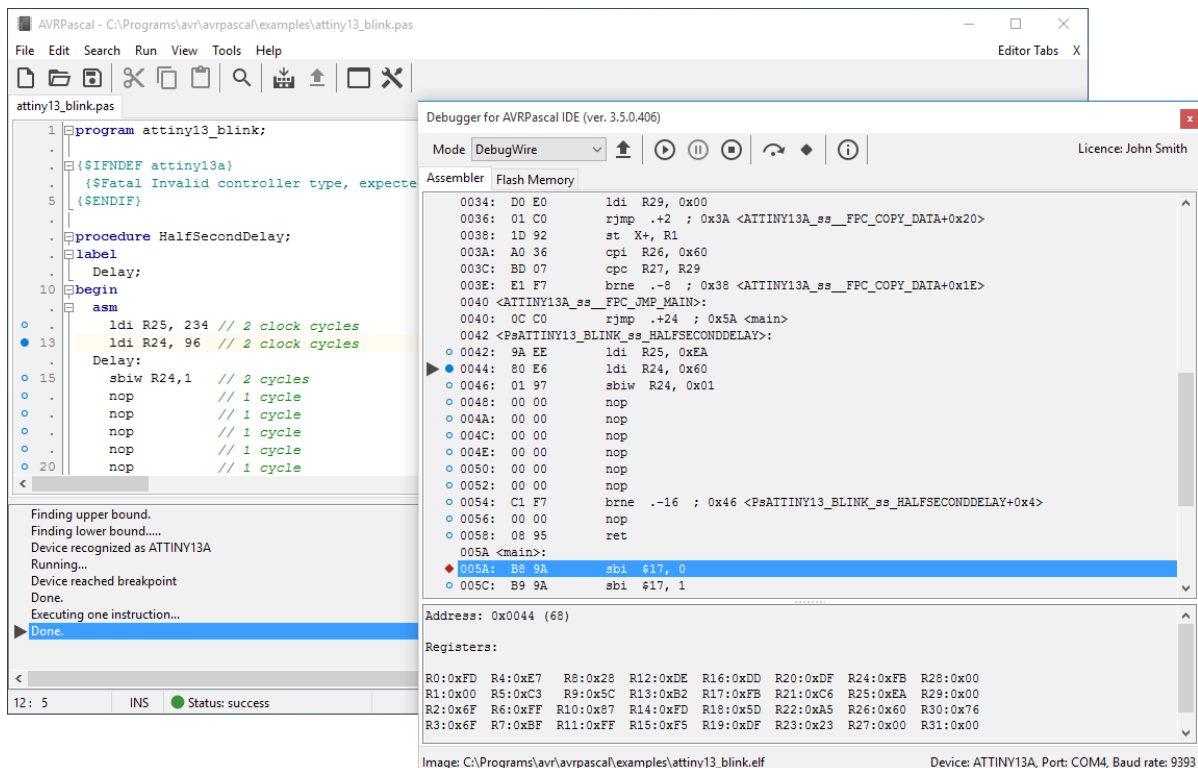
The *Flash Memory* tab displays the loaded flash memory image, differentiating code (blue background) and data (green background) sections. You can view memory bytes in either hexadecimal mode or as ASCII characters.



5. Debugger functions

The debugging tools are available in the toolbar and context menu. These tools include: *Upload* (⬆ - loading the flash memory image into the microcontroller), *Run* (▶ - starting the debugger), *Pause* (⏸ - pausing the debugger), *Stop and reset* (⏹ - stopping and resetting the debugger), *One step* (↻ - single step execution), *Breakpoint* (◆ - execution breakpoint) and *Config data* (ⓘ - microcontroller configuration data). The results of each action and any error information are displayed in the AVR Pascal Messages area.

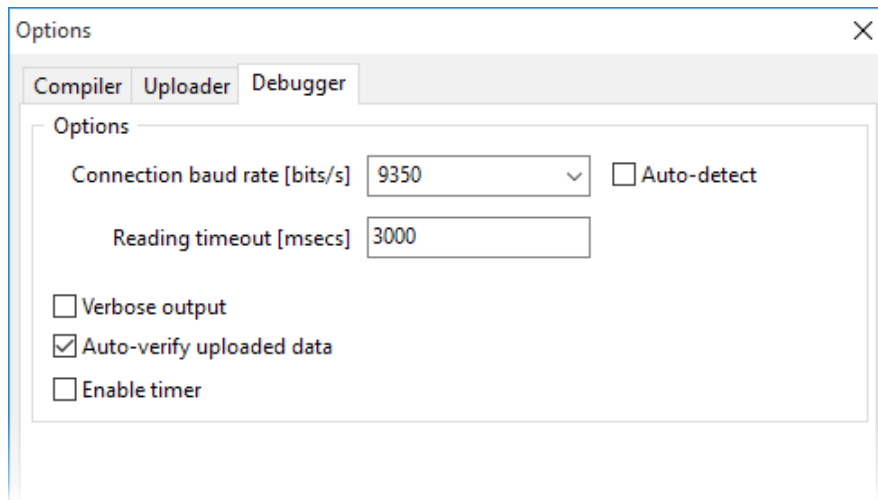
Before starting debugging, load the compiled program (referred to as the flash memory image) into the microcontroller using the *Upload* tool. This ensures that the microcontroller runs the exact code you compiled in AVR Pascal. Then, start the debugger using the *Run* tool. If no breakpoint is set, 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 is marked by an icon ◆ in the memory listing. 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 AVR Pascal tab, next to the corresponding address.



The *Stop and reset* function is used to stop 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.

6. Debugger options

AVRPascalDbg offers configuration options accessible in the *Debugger* tab of the AVRPascal *Options* window. Here, it is possible to configure the connection with the USB-UART adapter, indicating the baud 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 failure of the debugger⁵.

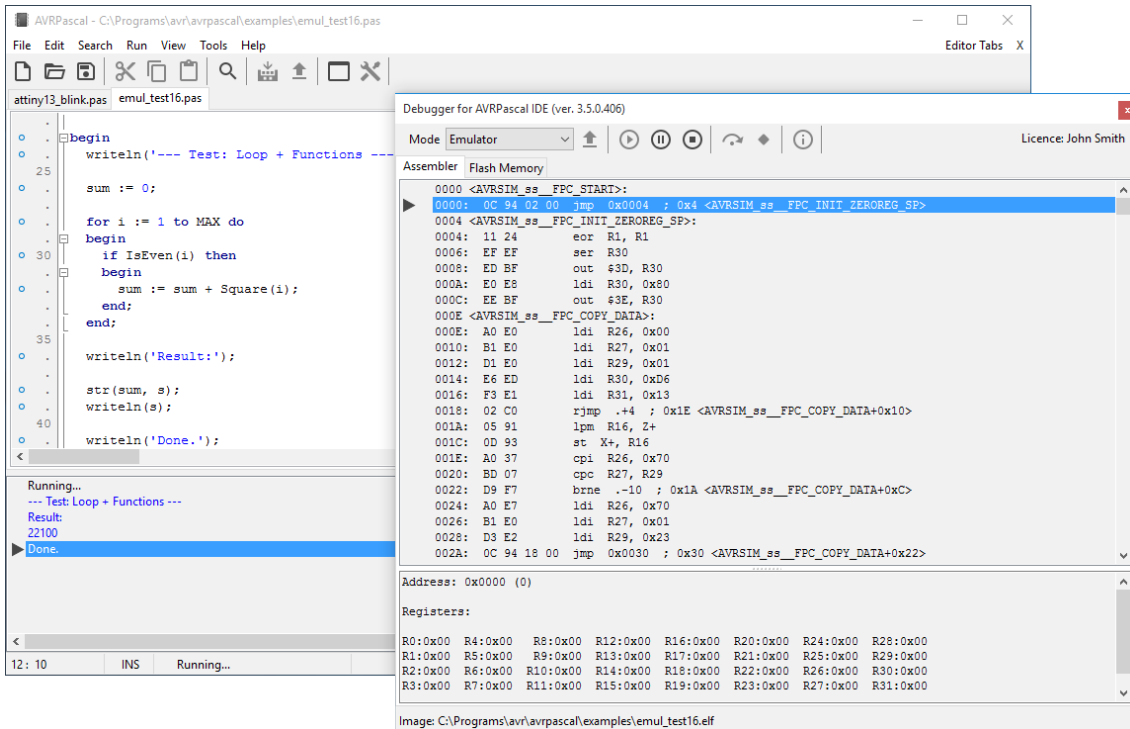
The next parameters concern the ability of the microcontroller to use a timer during debugging (*Enable timer*), automatic verification of data loaded into the microcontroller's flash memory (*Auto-verify uploaded data*) and the level of detail of messages sent to the *Messages* area of AVRPascal (*Verbose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

7. Emulator

The emulator allows program execution without the use of a physical microcontroller. The code is executed instruction by instruction based on the disassembled flash memory image. In emulation mode, the debugging functions described in section 5 are available, except for operations requiring a physical device, such as *Upload* and *Config data*.

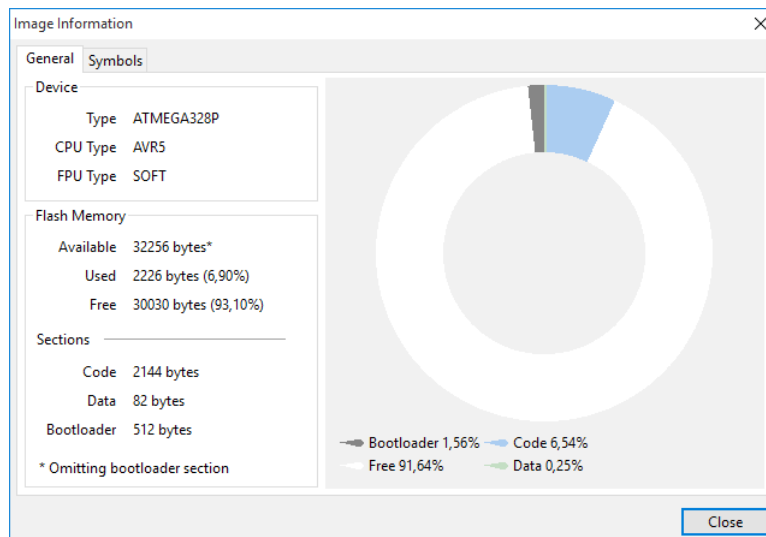
For programs compiled for the virtual AVRSIM device, the *write/writeln* functions can be used to display text or data in the *Messages* area. The output text is displayed in blue.

⁵ The optimal baud rate value is usually slightly higher than that determined automatically by the debugger.



8. Additional features

The debugger plugin also provides some statistical data for AVR Pascal. After successful compilation, it is possible to view flash memory usage statistics divided into sections along with a list of symbols generated by the compiler. This data is available in the *Image Information* window (*Run->Image Information...*)⁶.



⁶ These are code and data sections. However, the total size of both sections reported by the compiler in *Messages* is slightly larger than the actual size of the code and data in the binary file (*.bin). The difference is that the binary file does not contain the .bss section, which the compiler includes (along with .data) in the data section.

9. DEMO version

AVRPascalDbg is available on the website akarwowski.pl with a DEMO licence (file *000B462305076250.lic*). This is a demonstration version of the debugger in which debugging of programs exceeding a specified flash memory image size is not possible. The DEMO limit is 350 bytes for physical microcontrollers and 16 KB in AVRSIM mode. Detailed licensing information is available on the project website.

Andrzej Karwowski

email: ackarwow@gmail.com