Be Seeing You
stef[@|.]genesix.org

embedded AVR-Ada - Setup

preliminary edition for Paris Open Source Summit 2017

https://stef.genesix.org/pub/embedded_AVR-Ada_-_Setup_POSS2017.pdf

Weinberg's Second Law : If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.
Gerald Weinberg

stef[@|.]genesix.org

Ed.	*Release*	*Comments*	
1	20050505	Initial release in Texinfo.	sr
2	20120104	First windows setup with JTAGICE MkII.	sr
3	20171114	Port from GNU Texinfo format to ISO OpenDocument format.	sr
23	20171122	Preliminary edtion for Paris Open Source Summit 2017.	sr
24	20171204	Add appendix, Linux build section. Some typos corrected.	sr
27			

Étapes de mise à jour du tableau d'historique. Avant toute modification du document :

- Positionner le curseur sur l'avant dernière ligne du tableau (celle au dessus de « Édition courante ») ;
- Créer une nouvelle ligne dans le tableau ;
- Sélectionner et copier la dernière ligne, de « Validation » à « Email » (tout sauf la première colonne) ;
- Positionner le curseur sur l'avant dernière ligne, dans la colonne « Validation » ;
- Coller ;

Reporter l'indice de la dernière ligne dans la nouvelle ligne.

Printed	-
Édition	SR27- 25:30:16

stef[@|.]genesix.org

❑ Acknowledgments

I'm very grateful to a lot of people without whom this document could not simply exist.

I would like to thank, at one hand, the AVR-Ada project leader Rolf Ebert and his dream team : Tero Koskinen, Warren W. Gay (VE3WWG[1]), Bernd Trog and, at the other hand, John Leimon, author of a valuable AVR-Ada fork[2].

All my gratitude to some others very fine Ada related people : Enrique Bellido, Ludovic Brenta, Jeffrey Creem, Daniel Feneuille, Gautier de Montmollin, Pascal Obry, Jean-Pierre Rosen & Luigi Zaffalon.

Stéphane Richard translated the section three of the first chapter, which comes from his work on the AIDE[3] manual.

❑ Excuse me, I'm French

We are essentially famous as frog eaters. Surprisingly, frog eaters have recently discovered that others languages are somewhat largely used on earth. I'm a frog eater. So I've try to make up some stuff in this foreign dialect loosely known under the name of English. However, frogs don't really speak English. So your help is welcome to correct this bloody manual, for the sake of the Wildebeests, and Penguins too.

❑ Contact

This manual is released by Stephane Riviere (F1USA[4]).

stef@genesix.org
stef.genesix.org

1 International amateur radio call sign - https://en.wikipedia.org/wiki/Amateur_radio.

2 https://github.com/evilspacepirate/avr-ada

3 Ada Instant Development Environment - A true Ada 95 instant environment for Windows - https://stef.genesix.org/aide/aide.html

4 International amateur radio call sign - https://en.wikipedia.org/wiki/Amateur_radio.

stef[@|.]genesix.org

Table des matières

stef[@|.]genesix.org

stef[@|.]genesix.org

Releases

stef[@|.]genesix.org

stef[@|.]genesix.org

Introduction

One can write neatly in any language, including C. One can write badly in any language, including Ada. But Ada is the only language where it is more tedious to write badly than neatly.
Jean-Pierre Rosen

1 This manual

This is an attempt to present the developments opportunities in Ada on AVR 8 bits microcontroller Atmel, in an agnostic way, from the point of view of the operating system used, whether Linux or Windows. We do not want to impose the choice of one system or another, but on the contrary, we want to show the possibilities of development in bothes.

This document was originally a Texinfo based manual, created under Windows with the help of the AIDE (Ada Instant Development Environment) distribution[5]. For various reasons, this manual has been updated to the OpenDocument ISO format[6], with the help of the LibreOffice software.

This edition is a reboot of the previous ones, which were mostly Windows centric. It could also be seen as a research work about :
- All the possibilities offered to develop projects in Ada with Atmel AVR microcontrollers ;
- Implementing a full functional, up-to-date, Linux Ada 2012[7] toolchain with true real-time debugging capabilities.

Under Linux, the system of choice is Debian 8, preferably hosted as a virtual machine on a Debian 9 host, and the preferred Windows system is Seven, although XP or 10 should be fine also.

For the clarity and comfort of the reader, the organization of the manual has been kept as similar as possible between the Linux and Windows[8] chapters.

The author is not an Ada expert, nor a well-informed GNU free tools user, nor a fluent english writer : suggestions in order to improve this manual are very welcome.

2 Syntax notation

Inside a command line :
- A parameter between brackets [ ] is optional ;
- Two parameters separated by | are mutually exclusives.

[5] https://stef.genesix.org/aide/aide.html

[6] ISO/IEC 26300

[7] Last Ada compiler level (ISO/IEC 8652-2012)

[8] The Windows implementation described was functional in 2012 (and should always be) but has not been tested again, by lack of interest, time and means : we don't use Windows anymore but volonteers are welcome to update this part.

3 Ada

3.1 Introduction

This language is not known enough yet, at least not to the majority of us, much to the detri ment of many potential users for that matter. Compared to the fashionable languages, Ada is more portable, more readable, allows for higher abstraction levels and has features and functionalities unseen in other languages. Ada also allows a more comfortable experience in system programming[9] and proves itselt light enough to be usable on low class 8 bit processors[10].

Ada is the name of the first programmer to ever exist in humanity. And this first programmer was a woman : Augusta Ada Byron King, Countess of Lovelace, born in 1815, daughter of Byron, the great poet, Charles Babbage's assistant, she wrote programs destined to run on his famous machine.

Ada is an american military norm[11] as well as an international civil norm[12], it is the first object oriented language to be standarized at an international level. All Ada compilers must strictly adhere to the standard. There are hundreds of compilers destined to run on that many platforms but all of them will produce a code that runs identically.

Ada is used everywhere security is critical: Airbus (A3xx civil series and A400 military), Alsthom (High speed train), Boeing (777 and 787), EADS (Eurofighter, Ariane, ATV, many spaces probes), STS (line 14 Meteor), NASA (Electric power supply of the International Space Station). The list goes on and on. Everywhere reliability and security must come first, Ada is the language of choice.

3.2 Why use Ada

Ada was created because software engineering is a human activity. Humans make mistakes, the Ada compiler is friend to developers. Ada is also friend to project managers for large scale development. An Ada application is written, expanded and maintained very naturally. For these reasons, Ada is also friend to executives. Ada is the language of happy programmers, managers and users.

Because Ada is a confortable language by it's expressiveness and a restful language by it's reliability, humans involved with Ada also reflect the image of their language. The Ada community is a very comfortable community to visit and most meetings are very enlighting. Free libraries are numerous and are usually of a very high quality. Finally, the Ada community is very highly active and by now growing again.

3.3 The ending word

When Boeing decided, two decades ago, that all software for the 777[13] would be exclusively written in Ada, the corportate associates of the constructor made the remark that they were using,

[9] Thanks to it's representation clauses that obliterates the need to use bit masking for XORed for bit manipulation. This functionality *essential to system programming* is simply not there in C or even in Assembly language.

[10] Components that have at their disposal a couple dozen bytes of RAM and a couple Kilobytes of programming memory.

[11] MIL-STD-1815

[12] ISO/IEC 8652

for a long time, languages such as C, C++ and assembly language and that they were fully satisfied with them.

Boeing simply answered that only firms that could provide Ada software would be considered in contracts offerings. Therefore, the firms converted themselves to Ada.

Today, the development of software for the Boeing 777 nicknamed « The Ada Plane », has been performed and it is essentially thanks to the very big commercial success of this plane that Boeing was able to maintain the revenues created by its civil activities.

And what do the Boeing partner firms do from now on ? They continue to develop their new software in none other than... Ada, and here's why :

- They noticed that the length of time to convert developers to Ada is usually rather short. In a week, the developer is confortable enough to write software in Ada and in less than a month, he feels totally comfortable with the language ;
- These firms did their accounting : written in Ada, sofware costs less, present less anomalies, are ready sooner and are easier to maintain.

Today, these observations can still be verified at Boeing : The main language of the 787 « Dream-liner », it's new two engine state-of-art plane, is once again Ada.

4 Ada Toolchains

There is essentially four toolchains availables :
- The historical multi-platform LGPL project : AVR-Ada, a project from some very fine people, not easily buildable from scratch but which comes with functional binairies for Windows and Linux ;
- A Linux centric LGPL AVR-Ada fork from John Leimon, further named AVR-Ada-JLF. It is an up-to-date, easily buildable project but without available binairies ;
- The GNU/Linux Debian centric LGPL AIDE-AVR8, from Stéphane Rivière, fully based from John Leimon fork and original work from AVR-Ada team. It comes instantly ready-to-use, with affordable documentation, Linux binairies and true real-time debugging capabilities.
- The Windows Gnat-GPL-2012-Avr, an ended project from Adacore.

4.1 Levels of functionnalities

According to the amazing document « Debian policy for Ada[14] », from Ludovic Brenta :

Project	*GCC release*	*Debian GCC eq.*	*Equivalent*	*Norm compliance*
AVR-Ada	4.7.2	7 (Wheezy)	Gnat-GPL-2011	Ada 2005
AVR-Ada-JLF	4.9.2	8 (Jessie)	Gnat-GPL-2014	Ada 2012
AVR-Ada-SRF	4.9.2	8 (Jessie)	Gnat-GPL-2014	Ada 2012
Gnat-GPL-2012-AVR	undetermined[15]	n/a	n/a	Ada 2005

[13] The Boeing 777 is the world's biggest two engines plane and the first civil Boeing having electrical flight commands, 10 years later the Airbus A320.

[14] https://people.debian.org/~lbrenta/debian-ada-policy.html

Regarding AVR-Ada Ada coverage, the implementation level (Ada 2005 or Ada 2012) does not really matter and, similarly, the relative age of some available implementations does not matter either.

4.2 AVR-Ada (cross-compiler)

https://sourceforge.net/projects/avr-ada

Software	*Release*	*Comments*
gcc	4.7.2	
binutils	2.20	
avrlibc	1.8	
gdb	none	

Capabilities	*no*	*yes*	*Comments*
LGPL		X	Code can be used in a commercial environment without disclosing the sources.
Alive		X	Not active since some years but projects's team are allways responding.
Easily buildable	X		
Linux binairies		X	
Windows Binairies		X	
Real-time debug	X		
Full documentation	X		The documentation is scarce and disseminated.

Mailing list (devel) : https://lists.sourceforge.net/lists/listinfo/avr-ada-devel
Mailing list (cvs/ro) : https://lists.sourceforge.net/lists/listinfo/avr-ada-cvs-commit
Mailing list (devel) Web archive : https://sourceforge.net/p/avr-ada/mailman/avr-ada-devel

4.3 AVR-Ada-JLF (John Leimon Fork)

https://github.com/evilspacepirate/avr-ada

Software	*Release*	*Comments*
gcc	4.9.2	
binutils	2.20	
avrlibc	1.8	
gdb	none	

Capabilities	*no*	*yes*	*Comments*
LGPL		X	Code can be used in a commercial environment without disclosing the sources.
Alive		X	Not active since some years but project's leader is active.
Easily buildable		X	
Linux binairies	X		
Windows Binairies	X		

[15] Seems to be equivalent to GCC 4.8.x and partly Ada 2012 compliant but, frome other sources, GPL 2012 appears to be built against GCC 4.5.x. The real implement level has not been invastigated further, by lack of interest.

Capabilities	no	yes	Comments
Real-time debug	X		
Full documentation	X		The documentation is scarce and disseminated.

Comparison by directory with AVR-Ada original work :
- avr-ada/gcc-?.?-rts : contains all GCC runtimes, until 4.9
- avr-ada/apps : more applications
- avr-ada/avr : more controllers supported, new debug and test directories
- avr-ada/avr/avr-lib : new host directory
- avr-ada/doc : should be updated
- avr-ada/patches/binutils : contains all binutil patches, until 2.24
- avr-ada/patches/gcc : contains all GCC patches, until 4.9.2
- avr-ada/tools : new test/lcd-sim directory
- avr-ada/tools : deleted build and packaging directories

Notes :
- avr-ada/gcc-?.?-rts : run time system
- avr-ada/avr : support librairies
- avr-ada/patches : compiler and binutils patches

❑ Runtime diff between 4.7.2 et 4.9.2

Only one change, in system.ads, line124, an Atomic_Sync_Default definition is added :

```
../gcc-4.9.-rts/adainclude/system.ads

...
   ---------------------------------------
   -- System Implementation Parameters --
   ---------------------------------------

   --  These parameters provide information about the target that is used
   --  by the compiler. They are in the private part of System, where they
   --  can be accessed using the special circuitry in the Targparm unit
   --  whose source should be consulted for more detailed descriptions
   --  of the individual switch values.

   Atomic_Sync_Default       : constant Boolean := False;

...
```

➪ « targparm » = target parameter

4.4 AIDE-AVR8 (distribution)

https://stef.genesix.org

Software	Release	Comments
gcc	4.9.2	
binutils	2.20	
avrlibc	1.8	
gdb		

Capabilities	no	yes	Comments
LGPL		X	Code can be used in a commercial environment without disclosing the sources.
Alive		X	
Easily buildable		X	
Linux binairies		X	
Windows Binairies	X		The AVR-Ada project has ready to use Windows binairies
Real-time debug		X	
Full documentation		X	

<<<TODO>>>

⇨ This is an ongoing and unfinished project.

4.5 Gnat-GPL-2012-AVR (distribution)

https://www.adacore.com/download

Software	Release	Comments
gcc	4.5.x	
binutils	2.20	
avrlibc	none	
gdb	7.4	

Capabilities	no	yes	Comments
LGPL	X		Code can't be used in a commercial environment without disclosing the sources.
Alive	X		
Easily buildable	X		
Linux binairies	X		
Windows Binairies		X	
Real-time debug		X	Full realtime Atmel JTAGICE MKII debugging probe compatibility
Full documentation		X	Huge & multi-format documentation

This Adacore project has appeared in 2010, has been followed by two releases in 2011 & 2012, and since has been terminated.

This environment comes « ready to use » with an installer, all tools and an full featured IDE (GPS).

5 Atmel Resources

Atmel has been acquired by Microchip in 2016.

Home page : www.atmel.com/products/microcontrollers/avr
Documentation : http://www.atmel.com/webdoc/index.html

5.1 Atmel AVR Toolchain (C/C++ only)

Software	*Release*	*Comments*
gcc	4.9.2	
binutils	2.26	
avrlibc	2.0	
gdb	7.8	

Windows : http://www.atmel.com/tools/ATMELAVRTOOLCHAINFORWINDOWS.aspx
Linux : http://www.atmel.com/tools/ATMELAVRTOOLCHAINFORLINUX.aspx

http://distribute.atmel.no/tools/opensource/Atmel-AVR-GNU-Toolchain/3.5.4

Device Family Packs (DFP) : http://packs.download.atmel.com

How to use Atmel DFPs with Standalone toolchain :
- Download DFP from here (e.g. Atmel.ATmega_DFP.1.0.86.atpack) ;
- Unzip .atpack to packs directory (/home/packs/) ;
- Invoke avr-gcc with additional option -B to tell GCC where to look for device specific information. Add device header include path using -I option.

```
root@system : avr-gcc -mmcu=atmega328pb -B /home/packs/Atmel.ATmega_DFP.1.0.86/gcc/dev/atmega328pb/
-I /home/packs/Atmel.ATmega_DFP.1.0.86/include/
```

5.2 Studio 7

Home page : http://www.microchip.com/avr-support/atmel-studio-7
Documentation : http://www.atmel.com/webdoc/atmelstudio/index.html

The huge (887MB) & Windows only Studio 7, appears to be mandatory to upgrading Atmel probes firmwares.

There is two instructions sets to control Atmel probes. The basic one, used by open source Avrdude, and the extended one, use by Studio 7.

It is probably not wise to use Studio 7 for firmware updates if you need to use older windows tools or linux tools, as it updates firmware probe and windows drivers to a level somewhat incompatible with previous tools.

Choosing an Operating System

Investment in C programs reliability will increase up to exceed the probable cost of errors or until someone insists on recoding everything in Ada.
Gilb's laws synthesis

1 Linux

Linux must be preferred when you don't planned to use Windows tools. In our case, it is possible to get rid of Windows programs.

Install and use are straightforward.

Build could be more or less difficult, but it's not a criteria for common users, as ready-to-use setup programs are available.

2 Windows

Windows must be preferred when you use other tools only available under Windows. Indeed, in some cases, using Windows may be advantageous because, in electronics, many proprietary programs only exist on Windows.

Install and use are straightforward.

Build may be awfully ramdom, but it's not a criteria for common users, as ready-to-use setup programs are available.

stef[@|.]genesix.org

Choosing a Probe

The last bug isn't fixed until the last user is dead.
Sidney Markowitz

1 Discussion around real-time hardware debugging

For common users, real-time hardware debugging can be avoided, most of the time, with a serial link and/or a LCD display. However, once you work with real-time programs (in which step by step can not be performed) or your work on complex ones (like cyphers or protocols stacks), the practise of breakpoint debugging and step by step debugging become the rule. In theses latter cases simulator or traditional « poor man » debugging practises can't always help.

As the AVR micro-controllers come with all debugging circuitry inside the chip, there are no real emulators for the Atmel AVR-family. The link between the chip debugging circuitry and the software debugger is achieved by a JTAG interface. JTAG is a standardized interface control in the electronics industry (synchronous serial bainterface, for example). It is impossible to debug step by step without a JTAG probe. Basically, three hardware breakpoints are available.

Some low-cost JTAG probes are available for AVR. I've found one at the bottom of one of my drawers. But I can't make it at work with theses more recent ATmega AVR series (my Seedunino card comes with an ATmega1280 on-board). Some others JTAG probes should work with ATmega1280, but the price is only 30 % less than genuine Atmel probes.

In 2012, at the time I have to choose a realtime capable debugging probe, The Atmel Studio IDE of the time was very heavy and mostly buggy, and the Atmel up-to-date and affordable probe JTAGICE 3 had some problems too.

I've finally choosed to buy a JTAGICE mkII[16], which is a field proof genuine Atmel probe, ugly looking and expensive, with USB and (important to my point of view) serial links.

Luckily, I avoid the hardware revision zero and got a hardware revision one, which supports the PDI and aWire interfaces (serial number starting with A09-0041 or B0... indicate a hardware rev one).

2 JTAGICE mkII Probe

2.1 Introduction

Key features are[17] :
- Supports up to 3 hardware program breakpoints or 1 maskable data breakpoint ;
- Supports symbolic debug of complex data types including scope information ;
- Supports up to 128 software breakpoints ;
- Includes on-board 512kB SRAM for fast statement-level stepping ;
- Level converters support 1.8V to 5.5V target operation ;
- Uploads 256Kb code in ~30 seconds (XMEGA using JTAG interface) ;

[16] Still available in late 2017 as compatible stuff for the half of its price - worth 250 € in late 2011 (search « jtagice mkII amazon »).

[17] According to http://www.atmel.com/tools/AVRJTAGICEMKII.aspx

stef[@|.]genesix.org

- Full-speed USB 2.0 compliant (12 MB/s) and RS-232 host interfaces ;
- Externally or USB powered.

Probe page : www.atmel.com/tools/AVRJTAGICEMKII.aspx
Documentation : http://www.atmel.com/webdoc/jtagicemkii/index.html
Protocol : http://www.atmel.com/webdoc/protocoldocs

2.2 Powering

As state in the probe manual, the JTAGICE mkII probe is able to operate using an external power supply providing 9...15V DC or it can be powered directly from the USB bus (if the plug can deliver a minimum of 500mA, as a self powered hub). An internal switch will default select the power from the external power supply. However, if this is not connected, or the external power supply drops below a usable level the power will be taken from the USB (if connected).

Although any polarity will work, the preferred polarity of the DC jack is negative-centre due to the power switch grounding.

When powering up the JTAGICE mkII, the power LED should illuminate immediately. If the LED does not light up, check that an adequate power supply is being used.

Always switch off the target application power before switching off the Atmel AVR JTAGICE mkII.

Never leave a powered-down JTAGICE mkII connected to a powered application as current may leak from the application and result in damage to the emulator.

If the target application uses the JTAG pins for general purpose I/O, the JTAGICE mkII can still be used to program the target via the JTAG pins (provided that the JTAG enable fuse is set.) However, be sure to connect the RESET pin of the target device to the nSRST pin of the emulator. Without this connection, the target application cannot be prevented from running after programming. If the application drives the JTAG pins as outputs, there will be signal contention with the emulator, which may result in damage to the emulator and/or target.

2.3 Tips

❑ Wiring

Do not forget to wire the RST line of the JTAGICE mkII otherwise the programming is not done and the debugger does not work.

❑ Software

One can program before the debugging session, via the --erase --program flags (which must now go together). However, this causes the loss of the bootloader.

2.4 Programming firmware

❑ Linux

There is no mean to update firmware with Linux. You have to create a Windows 7 VM (2Go RAM and 30 Go HD min) and then follow the Windows programming firmware installation.

In a Windows 7 VM hosted on Linux, Studio 7 install well and then, when using it, behave well.

- Programming firmware from a VM

However, using a VM to update firmware through an USB cable come with some tricks you should aware of, thanks to Clawson, an user of the AvrFreaks well known forum, Some mysteries are solved[18].

As he noticed, I also experiment the « Windows bong-bing noise » alerting to the fact that the USB device had disconnected. If he went to the « Devices-USB Devices menus » of the Virtual Machine and re-ticked the JTAGICE to reconnect it he then just ended up with the firmware upgrader saying the upgrade had failed.

After a while he realised what's going on : the JTAGICEmkII can actually appear as TWO different USB devices. One called « Atmel JTAGICE mkII » and one called « Atmel AVRBLDR » which is clearly the bootloader in the JTAGICE mkII that is used to apply the firmware update.

So what he ended up doing was applied two USB filter rules to the VM. Using a filter means that if this device ever enumerates to the host (Linux in our case) it is immediately captured and connected to the virtual machine.

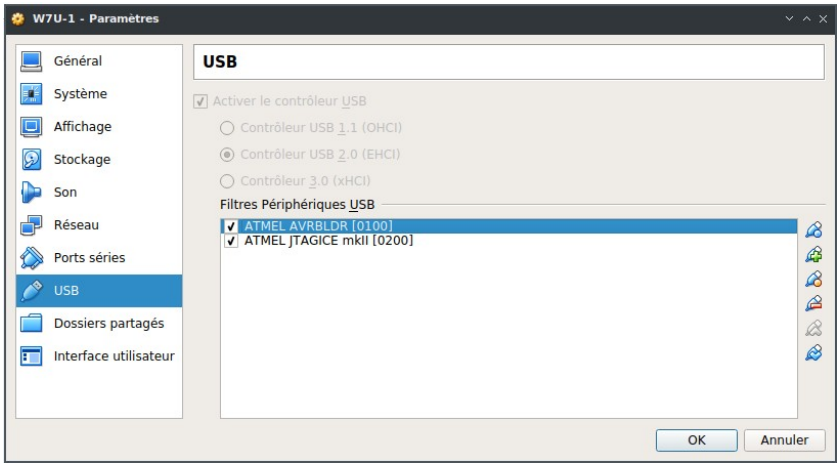

Fig. 1: VirtualBox / Peripherals / USB Parameters... / USB Peripherals Filters

Filters details :

[18] http://www.avrfreaks.net/forum/tuthardusing-jtacicemkii-atmel-tools-virtualbox?name=PNphpBB2&file=viewtopic&t=126054

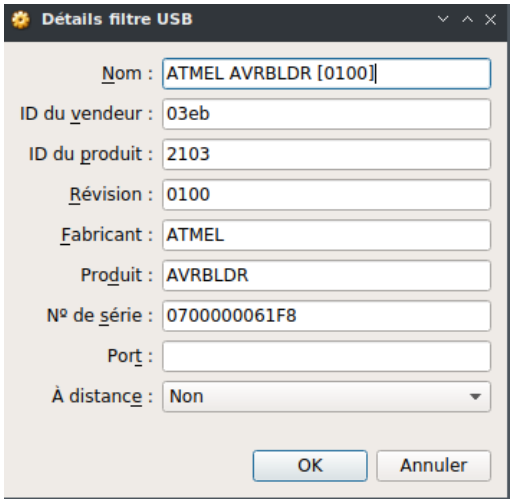

Fig. 2: Bootloader filter

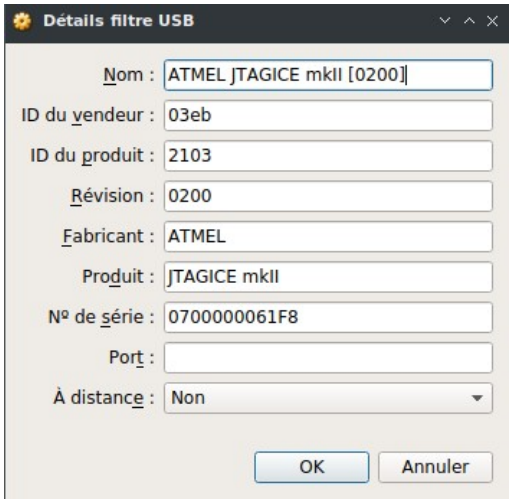

Fig. 3: Probe filter

❑ Windows

An old Studio 4 under XP is ok to update the JTAGICE MkII firmware. In 2013, JTAGICE MkII firmware 6.06 (release included in Studio 4) was ok for Avrdude. But if you need the latest firmware, you need Studio 7 and Windows 7 at least.

Install AVR Studio 7 (as-installer-7.0.1645-full.exe, 887 MB !). It is an incredebly lengthy process. You need .NET 4 dotNetFx40_Full_x86_x64.exe and the KB Update Windows6.1-KB2999226-x64.msu. You have to reboot. And even after, the installation process is not finished ! As a result, your hard disk has « gained » 8231 files, 1238 forlders and loss 1113 MB ! This piece of software is incredibly fat, pretty slow and complex.

In return, it offers a very complete environment, has many helpers, and could be used to get acquainted with the Atmel AVR family. The interface is neat and well organized.

Connect the JTAGICE mkII via USB and check the USB driver installation.

- Using Studio 4 (release 6.06)

Launch AVR studio and go to Tools / Jtagice mkII upgrade...
Tools / Extension and updates
To update, click OK and the firmware is updated in less than 20 seconds. The update procedure does not check if the firmware is already up-to-date and always updates.

- Using Studio 7 (release 7.27)

According to the manual, Studio 7 will automatically upgrade the tool's firmware when needed. A potential firmware upgrade is triggered once you start using a tool. Examples: the first time you launch a debug session or the first time you select the tool in the Device Programming dialog.

The tool can not be used by Atmel Studio if the user chooses not to upgrade.

You can also check for firmware used and upgrades availables by using View / Available Atmel Tools, the click right Upgrade... You get On Tool and On Disk releases, as well as the firmware status (up-to-date or need to update). If you decide to upgrade, you need to click on the Upgrade button, or quit the window with the Close button.

- Using Console

To our taste, this way is a better one.

```
-- Tools

C:> C:\Program Files (x86)\Atmel\Studio\7.0\tools\JTAGICEmkII

-- Check the current version

C:> <Atmel>\7.0\atbackend\atfw.exe -t jtagicemkii -r

Found jtagicemkii:0700000061F8
Master MCU Version: 6.11
Slave MCU Version: 6.11

-- Updating (if the processs freeze, without quitting, a straight probe on/off could help)

C:> <Atmel>\7.0\atbackend\atfw.exe -a c:\Users\sr\jtagicemkii_fw.zip -t jtagicemkii -s 0700000061F8

Upgrading jtagicemkii:0700000061F8
Upgrading Main MCU: [==========]
Upgrading Slave MCU: [==========]
Successful upgrade

-- Check new version

C:> <Atmel>\7.0\atbackend\atfw.exe -t jtagicemkii -r

Found jtagicemkii:0700000061F8
Master MCU Version: 7.27
Slave MCU Version: 7.27
```

2.5 Using GDB

The GDB debugger is able to use the Atmel JTAGICE mkII probe.

Only connection through JTAG is supported.
Only use 3 breakpoints are supported.

For some operations (such as the next command), GDB uses temporary breakpoints.

Do not forget to correctly program the fuses (be sure OCD and JTAG are enabled).

To use the probe, you simply have to select the « atmel-mkii » target :

```
$ gdb myprog
(gdb) target atmel-mkii
(gdb) load
(gdb) run
```


stef[@|.]genesix.org

As shown above, GDB is able to erase and program the AVR with the load command.[19]

When configured for debugging the Atmel AVR, GDB supports the following AVR-specific commands : info io_registers. This command displays information about the AVR I/O registers. For each register, GDB prints its number and value.

2.6 Debugging the debugger

❑ JTAG clock

According to the JTAGICE mkII manual, the target clock frequency must be accurately specified in the software front-end before starting a debug session. For synchronization reasons, the JTAG TCK signal must be less than one fourth of the target clock frequency for reliable debugging.

Setting the target clock frequency too high will cause failure of a debug session shortly after programming completes. This may be accompanied by several spurious SLEEP, WAKEUP, or IDR messages being displayed. When programming via the JTAG interface, the TCK frequency is limited by the maximum frequency rating of the target device, and not the actual clock frequency being used.

When using the internal RC oscillator, be aware that the frequency may vary from device to device and is affected by temperature and VCC changes. Be conservative when specifying the target clock frequency.

❑ GCC related information

Single stepping GCC-generated code in source-level may not always be possible. Set optimization level to lowest for best results, and use the disassemble view when necessary.

The JTAGICE mkII is optimized to work the way AVR Studio works, and that way is fairly different than the way GDB works. Thus, the JTAGICE mkII communication in AVaRICE causes much more traffic than it does in AVR Studio. This is most notable when single-stepping. As an alternative, you might consider using temporary breakpoints ("tbreak" or just "tb" in GDB).

Also, AVaRICE is sometimes not as efficient as it could be when communicating with the JTAGICE mkII. More manpower would be needed to clean things like that up (volunteers welcome !).

❑ Identification

To identify the JTAGICE mkII probe, just plug its USB cable :

```
root@system : dmesg

[1559152.221003] usb 1-3.4.4: new full-speed USB device number 14 using xhci_hcd
[1559152.335051] usb 1-3.4.4: New USB device found, idVendor=03eb, idProduct=2103
[1559152.335054] usb 1-3.4.4: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[1559152.335057] usb 1-3.4.4: Product: JTAGICE mkII
```

[19] Excerps from GNAT User's Guide - Supplement for Cross Platforms - Copyright © 1995-2011, Free Software Foundation.

```
[1559152.335059] usb 1-3.4.4: Manufacturer: ATMEL
[1559152.335060] usb 1-3.4.4: SerialNumber: 0700000061F8
```


stef[@|.]genesix.org

AVR-Ada-JLF Linux Toolchain

With the Wildebeest and the Penguin, there's no Bull.
Number Six

1 Introduction

This is the toolchain setup intended for AIDE-AVR8. This latter comes with its own manual.

2 System

Install a Debian 8 « Netinst » in a VirtualBox VM. And then put some light graphic environment like OpenBox or WindowMaker.

All this stuff is precisely described in the *french* manual[20] « station Debian 9 - Installation », available on my website[21], which included a Debian 8 VM creation example. Even foreign readers could make profit opening it, as the console instructions are sufficiently explanatory.

The station should have at least one or two serial links and/or one or two free USB 2.0 ports with USB/Serial converters.

3 Base setup

3.1 Atmel tool chain

<<<TODO>>>

3.2 AIDE-AVR8 setup

<<<TODO>>>

4 Ada Toolchain with Arduino card validation

<<<TODO>>>

5 Ada Toolchain with Jtagice mkII

<<<TODO>>>

6 Using

<<<TODO>>>

[20] Of course, translaters volunteers are welcome again.

[21] https://stef.genesix.org - new and updated site planned in 2018, first quarter.

stef[@|.]genesix.org

7 Compiler - Build

7.1 Introduction

The goal is to build a Gnat Ada compiler from a GCC 4.9.2 base.

The Debian 8 Jessie comes with a 4.9.2 GCC toolchain.

7.2 Build - Step 1 : system environment

Prepare the system :

```
root@system : aptitude install build-essential make git gnat-4.9 gnat-4.9-doc bison flex libc6-dev
libmpc-dev libgmp-dev libmpfr-dev zlib1g-dev libtool texinfo
```

You need up to :
- 10 GB of free HD space ;
- 2 GB RAM ;
- A couple of hours, depending of your workstation power.

You also need a well cooled rig. My NUC Intel ran from 42° to 85° C[22] !

You have two build script available :
- The original build.sh, subject to some modifications ;
- The new build-avr-ada-aide.sh script, a more user friendly and modular one, with a horodated log file. The script handle all the stuff, including the cloning of the AVR-Ada-JLF project. Contrary to the original build script, this new one handle all needed archives, files and patches before starting to build the project.

The new script user interface :

```
BUILD-AVR-Ada-JLF - A Linux oriented script to build AVR-Ada-JLF - Version 1.0.
Copyright (C) Tero Kostinen, John Leimo, Stephane Riviere 2014-2017 under the GPLv3 licence

Usage : ./build-avr-ada-jlf.sh --[build|fresh|aide]

--build : Resume from previous build, if exist
--fresh : Erase previous build, then make a fresh one
--aide  : Build AIDE-AVR8 distribution
```

7.3 Build - Step 2 : build with the original script *build.sh*

Choose a base directory for the project :

```
root@system : cd /usr/local/bin
```

Get the AVR-Ada-JLF project, and go into it :

[22] Celcius or Centigrades degrees, 49° to 85° Celcius is eq. to 120,2 ° to 185 ° Fahrenheit.

```
root@system : git clone https://github.com/evilspacepirate/avr-ada

root@system : cd ./avr-ada
```

You must then alter the script build.sh this way :

```
./build.sh

...

Line 65, replace
  echo " build_gcc47()"
with
  echo " build_gcc-${GCC_VERSION}()"

...

Line 274, replace
  wget http://download.savannah.gnu.org/releases/avr-libc/avr-libc-1.8.0.tar.bz2
with
  wget http://download.savannah.gnu.org/releases/avr-libc/old-releases/avr-libc-1.8.0.tar.bz2

...

Line 298, replace
  # Search for the texinfo patch #
with
  # Search for the avr-threads.diff patch #

...
```

You are now ready to build the AVR-Ada-JLF project.

7.4 Build - Step 2 : build with the new build-avr-ada-aide.sh

Choose a base directory for the project :

```
root@system : cd /usr/local/bin
```

Get the script https://stef.genesix.org/pub/build-avr-ada-aide.sh :

```
root@system : wget https://stef.genesix.org/pub/build-avr-ada-aide.sh
```

You are now ready to build the AVR-Ada-JLF project.

7.5 Really build (or not)

Now you can launch either build.sh or build-avr-ada-aide.sh.

But, before doing that, be sure to remember the introduction. Also it took time, could be more than a couple of hours, with a small sized VM or an old workstation. Have confidence in cooling your PC, even a well cooled one (see below).

```
root@system : sensors

coretemp-isa-0000
Adapter: ISA adapter
Physical id 0:  +85.0°C  (high = +100.0°C, crit = +100.0°C)
Core 0:         +85.0°C  (high = +100.0°C, crit = +100.0°C)
Core 1:         +78.0°C  (high = +100.0°C, crit = +100.0°C)

pch_skylake-virtual-0
Adapter: Virtual device
temp1:        +74.0°C
```

If you have follow the instructions, you should obtain this :

```
20171203-204712  - MESSAGE :  ========================================================================
20171203-204712  - MESSAGE :  BUILD START
20171203-204712  - MESSAGE :  ========================================================================
20171203-204712  - MESSAGE :  Erasing previous build.
20171203-204715  - MESSAGE :  ----------------------------------------
20171203-204715  - MESSAGE :  CHECK SYSTEM
20171203-204715  - MESSAGE :  ----------------------------------------
20171203-204715  - MESSAGE :  Distribution : Debian
20171203-204715  - MESSAGE :  Log file : /usr/local/bin/20171203-2047.log
20171203-204715  - MESSAGE :  Build directory : /usr/local/bin/avr-ada
20171203-204715  - MESSAGE :  Build type : Build AVR-Ada-JLF - Full erase previous build, then do a
fresh one.
20171203-204715  - MESSAGE :  GNAT version : 4.9.2
20171203-204715  - MESSAGE :  ----------------------------------------
20171203-204715  - MESSAGE :  CHECK FILES
20171203-204715  - MESSAGE :  ----------------------------------------
20171203-205000  - MESSAGE :  gcc-4.9.2.tar.gz already downloaded, skipping.
20171203-205000  - MESSAGE :  binutils-2.20.1a.tar.bz2 already downloaded, skipping.
20171203-205000  - MESSAGE :  avr-libc-1.8.0.tar.bz2 already downloaded, skipping.
20171203-205000  - MESSAGE :  ----------------------------------------
20171203-205000  - MESSAGE :  BUILD GCC-4.9.2
20171203-205000  - MESSAGE :  ----------------------------------------
20171203-205007  - MESSAGE :  Texinfo is already patched, skipping patch with texinfo.patch
20171203-205007  - MESSAGE :  Texinfo is already patched, skipping patch with texinfo2.patch
20171203-205007  - MESSAGE :  LIBRARY_PATH : '/usr/lib/x86_64-linux-gnu/'
20171203-205007  - MESSAGE :  C_INCLUDE_PATH : '/usr/include/x86_64-linux-gnu'
20171203-205007  - MESSAGE :  CPLUS_INCLUDE_PATH : '/usr/include/x86_64-linux-gnu'
20171203-205007  - MESSAGE :  CPATH : '/usr/include/x86_64-linux-gnu'
20171203-215727  - MESSAGE :  ----------------------------------------
20171203-215727  - MESSAGE :  BUILD AVR-BINUTILS
20171203-215727  - MESSAGE :  ----------------------------------------
20171203-215900  - MESSAGE :  ----------------------------------------
20171203-215900  - MESSAGE :  BUILD AVR-GCC-GNAT
20171203-215900  - MESSAGE :  ----------------------------------------
20171203-215900  - MESSAGE :  GNATPREFIX   : '/usr/local/bin/avr-ada/gnat-native-49'
20171203-215900  - MESSAGE :  AVRADAPREFIX : '/usr/local/bin/avr-ada/avr-ada-49'
20171203-215900  - MESSAGE :  Removing old gcc-4.9.2 directory.
20171203-221655  - MESSAGE :  ----------------------------------------
20171203-221655  - MESSAGE :  BUILD AVR-LIBC
20171203-221655  - MESSAGE :  ----------------------------------------
20171203-221833  - MESSAGE :  ----------------------------------------
20171203-221833  - MESSAGE :  BUILD AVR-ADA
20171203-221833  - MESSAGE :  ----------------------------------------
20171203-221900  - MESSAGE :  ----------------------------------------
20171203-221900  - MESSAGE :  AVR-Ada BUILD COMPLETE.
20171203-221900  - MESSAGE :
20171203-221900  - MESSAGE :  Binaries are located at :
20171203-221900  - MESSAGE :  /usr/local/bin/avr-ada/avr-ada-49/bin
20171203-221900  - MESSAGE :  ----------------------------------------
20171203-221900  - MESSAGE :  Build end with success.
20171203-221900  - MESSAGE :  ----------------------------------------
20171203-221900  - MESSAGE :  ========================================================================
20171203-221900  - MESSAGE :  BUILD END
20171203-221900  - MESSAGE :  ========================================================================
```

Congratulations !

stef[@|.]genesix.org

8 Tools - Build

8.1 Avarice

❑ Introduction

According to the Avarice https://packages.debian.org page, AVaRICE is a program which interfaces the GNU Debugger with the AVR JTAGICE mkI and mkII and other debuggers available from Atmel.

It connects to GDB via a TCP socket and communicates via GDB's « serial debug protocol ». This protocol allows GDB to send commands like « set/remove breakpoint » and « read/write memory ». AVaRICE translates this commands into the Atmel protocol used to control the JTAG ICE (or other) debugger.

Because the GDB-AVaRICE connection is via a TCP socket, the two programs do not need to run on the same machine.

The currently supported debuggers are:
- JTAGICE mkI ;
- JTAGICE mkII ;
- AVR Dragon.

❑ Build

We must compile the last 2.13 release and need additional package :

```
root@system : aptitude install libbfd-dev
```

Download :

```
root@system : wget https://sourceforge.net/projects/avarice/files/avarice/avarice-2.13/avarice-2.13.tar.bz2/download
-- Unarchive to ./Avarice-2-13
root@system : tar xvf download
root@system : cd ./avarice-2.13
```

Compile & test :

```
root@system : ./configure
root@system : make
root@system : cd ./src
root@system : ./avarice --help
```


stef[@|.]genesix.org

```
AVaRICE version 2.13, Nov 30 2017 14:51:32

Usage: ./avarice [OPTION]... [[HOST_NAME]:PORT]

Options:
  -h, --help                  Print this message.
  -1, --mkI                   Connect to JTAG ICE mkI (default)
  -2, --mkII                  Connect to JTAG ICE mkII
  -B, --jtag-bitrate <rate>   Set the bitrate that the JTAG box communicates
                                with the avr target device. This must be less
                                than 1/4 of the frequency of the target. Valid
                                values are 1000/500/250/125 kHz (mkI),
                                or 22 through 6400 kHz (mkII).
                                (default: 250 kHz)
  -C, --capture               Capture running program.
                                Note: debugging must have been enabled prior
                                to starting the program. (e.g., by running
                                avarice earlier)
  -c, --daisy-chain <ub,ua,bb,ba> Daisy chain settings:
                                <units before, units after,
                                bits before, bits after>
  -D, --detach                Detach once synced with JTAG ICE
  -d, --debug                 Enable printing of debug information.
  -e, --erase                 Erase target.
  -E, --event <eventlist>     List of events that do not interrupt.
                                JTAG ICE mkII and AVR Dragon only.
                                Default is "none,run,target_power_on,target_sleep,target_wakeup"
  -g, --dragon                Connect to an AVR Dragon rather than a JTAG ICE.
                                This implies --mkII, but might be required in
                                addition to --debugwire when debugWire is to
                                be used.
  -I, --ignore-intr           Automatically step over interrupts.
                                Note: EXPERIMENTAL. Can not currently handle
                                devices fused for compatibility.
  -j, --jtag <devname>        Port attached to JTAG box (default: /dev/avrjtag).
  -k, --known-devices         Print a list of known devices.
  -L, --write-lockbits <ll>   Write lock bits.
  -l, --read-lockbits         Read lock bits.
  -P, --part <name>           Target device name (e.g. atmega16)

  -r, --read-fuses            Read fuses bytes.
  -R, --reset-srst            External reset through nSRST signal.
  -V, --version               Print version information.
  -w, --debugwire             For the JTAG ICE mkII, connect to the target
                                using debugWire protocol rather than JTAG.
  -W, --write-fuses <eehhll>  Write fuses bytes.
  -x, --xmega                 AVR part is an ATxmega device, using JTAG.
  -X, --pdi                   AVR part is an ATxmega device, using PDI.
HOST_NAME defaults to 0.0.0.0 (listen on any interface).
":PORT" is required to put avarice into gdb server mode.

e.g. ./avarice --erase --program --file test.bin --jtag /dev/ttyS0 :4242
```

8.2 Avrdude

❑ Introduction

Avrdude is a tool to program ATMEL AVR devices, through many available hardware interfaces.

The currently supported probes and subsystems are:
- JTAG ICE mkI ;
- JTAG ICE mkII ;
- AVR Dragon ;
- And many more devices : AVR ISP mkII, STK600, AVR JTAGICE3, AVR JTAGICE3 (v3.x), Xplained Pro board, Atmel-ICE, ATxmega32A4U DFU, Cactus RF60 DFU, atmega8u2 DFU, atmega16u2 DFU, atmega32u2 DFU, at32uc3a3 DFU, atmega16u4 DFU, atmega32u4 DFU, at32uc3b0/1 DFU, at90usb82 DFU, at32uc3a0/1 DFU, at90usb646/647 DFU, at90usb162 DFU,

at90usb1286/1287 AVR DFU, at90usb1286/1287 AVR DFU, at89c5130/c5131 DFU, at89c5132/c51snd1c DFU, PICkit2, USBasp, NIBObee and USBtinyISP.

❑ Build

We must compile the last 6.3 release and need some additional packages :

```
root@system :  aptitude  install  libusb-dev  libelf-dev  libftdi-dev  libusb-1.0-0-dev  libftdi1
libhidapi-dev
```

Download :

```
root@system : wget http://download.savannah.gnu.org/releases/avrdude/avrdude-6.3.tar.gz

-- Unarchive to ./avrdude-6.3

root@system : tar xvf avrdude-6.3.tar.gz

root@system : cd ./avarice-2.13
```

Compile & test :

```
root@system : ./configure

root@system : make

root@system : cd ./src

root@system : ./avrdude

Usage: avrdude [options]
Options:
  -p <partno>                Required. Specify AVR device.
  -b <baudrate>              Override RS-232 baud rate.
  -B <bitclock>              Specify JTAG/STK500v2 bit clock period (us).
  -C <config-file>           Specify location of configuration file.
  -c <programmer>            Specify programmer type.
  -D                         Disable auto erase for flash memory
  -i <delay>                 ISP Clock Delay [in microseconds]
  -P <port>                  Specify connection port.
  -F                         Override invalid signature check.
  -e                         Perform a chip erase.
  -O                         Perform RC oscillator calibration (see AVR053).
  -U <memtype>:r|w|v:<filename>[:format]
                             Memory operation specification.
                             Multiple -U options are allowed, each request
                             is performed in the order specified.
  -n                         Do not write anything to the device.
  -V                         Do not verify.
  -u                         Disable safemode, default when running from a script.
  -s                         Silent safemode operation, will not ask you if
                             fuses should be changed back.
  -t                         Enter terminal mode.
  -E <exitspec>[,<exitspec>] List programmer exit specifications.
  -x <extended_param>        Pass <extended_param> to programmer.
  -y                         Count # erase cycles in EEPROM.
  -Y <number>                Initialize erase cycle # in EEPROM.
  -v                         Verbose output. -v -v for more.
  -q                         Quell progress output. -q -q for less.
  -l logfile                 Use logfile rather than stderr for diagnostics.
  -?                         Display this usage.

avrdude version 6.3, URL: <http://savannah.nongnu.org/projects/avrdude/>
```

❑ Install

Create /lib/udev/rules.d/60-avrdude.rules :

/lib/udev/rules.d/60-avrdude.rules

```
# JTAG ICE mkII
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2103", GROUP="plugdev", MODE="0660"
# AVR ISP mkII
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2104", GROUP="plugdev", MODE="0660"
# STK600
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2106", GROUP="plugdev", MODE="0660"
# AVR Dragon
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2107", GROUP="plugdev", MODE="0660"
# AVR JTAGICE3
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2110", GROUP="plugdev", MODE="0660"
# Xplained Pro board
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2111", GROUP="plugdev", MODE="0660"
# AVR JTAGICE3 (v3.x)
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2140", GROUP="plugdev", MODE="0660"
# Atmel-ICE
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2141", GROUP="plugdev", MODE="0660"
# ATxmega32A4U DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2fe4", GROUP="plugdev", MODE="0660"
# Cactus V6 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2fe6", GROUP="plugdev", MODE="0660"
# Cactus RF60 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2fea", GROUP="plugdev", MODE="0660"
# atmega8u2 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2fee", GROUP="plugdev", MODE="0660"
# atmega16u2 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2fef", GROUP="plugdev", MODE="0660"
# atmega32u2 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff0", GROUP="plugdev", MODE="0660"
# at32uc3a3 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff1", GROUP="plugdev", MODE="0660"
# atmega16u4 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff3", GROUP="plugdev", MODE="0660"
# atmega32u4 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff4", GROUP="plugdev", MODE="0660"
# at32uc3b0/1 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff6", GROUP="plugdev", MODE="0660"
# at90usb82 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff7", GROUP="plugdev", MODE="0660"
# at32uc3a0/1 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff8", GROUP="plugdev", MODE="0660"
# at90usb646/647 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ff9", GROUP="plugdev", MODE="0660"
# at90usb162 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ffa", GROUP="plugdev", MODE="0660"
# at90usb1286/1287 AVR DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ffb", GROUP="plugdev", MODE="0660"
# at89c5130/c5131 DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2ffd", GROUP="plugdev", MODE="0660"
# at89c5132/c51snd1c DFU
SUBSYSTEM=="usb", ATTR{idVendor}=="03eb", ATTR{idProduct}=="2fff", GROUP="plugdev", MODE="0660"
# PICkit2
SUBSYSTEM=="usb", ATTR{idVendor}=="04d8", ATTR{idProduct}=="0033", GROUP="plugdev", MODE="0660"
# USBasp
SUBSYSTEM=="usb", ATTR{idVendor}=="16c0", ATTR{idProduct}=="05dc", GROUP="plugdev", MODE="0660"
# NIBObee
SUBSYSTEM=="usb", ATTR{idVendor}=="16c0", ATTR{idProduct}=="092f", GROUP="plugdev", MODE="0660"
# USBtinyISP
SUBSYSTEM=="usb", ATTR{idVendor}=="1781", ATTR{idProduct}=="0c9f", GROUP="plugdev", MODE="0660"
```

Copy configuration file :

```
root@system : cp ./avrdude/avrdude.conf /etc
```


stef[@|.]genesix.org

9 Documentation

9.1 Compiler

Build the GCC documentation from scratch appears to be more difficult than build the compiler.

Don't have time to investigate and pick the documentation here : https://gcc.gnu.org/onlinedocs

9.2 Tools

PDF documentation has been collected for most of the tools.

9.3 Missing information

Please contact the author of this manual if you feel there is a lack of documentation.

AVR-Ada Windows toolchain

My Operating System is Emacs and Windows is its driver.
Anonymous

1 Introduction

This whole chapter was written in 20120104, against AVR-Ada 1.1.0 release on Windows XP and must be updated to AVR-Ada 1.2.2 on Windows Seven at least.

2 System

Windows does not require any special settings. The workstation should have at least one or two serial ports and one or two free USB 2.0 ports.

3 Base setup

3.1 Arduino

If you use Arduino hardware, you may choose to validate your hardware setup with Arduino IDE.

❑ Install toolchain

At first, download and install :

```
http://arduino.googlecode.com/files/arduino-1.0-windows.zip
```

Install FTDI driver and FTDI port.

The driver is located in : <arduino root>\arduino\drivers\ftdi usb drivers.

❑ Toolchain validation

Launch Arduino IDE and configure it :

```
Set COM port/Tools/Serial Port/Comx
Set Arduino board type : Tools/Board/Arduino Mega (ATmega1280)

Open the blink example : File/Open/examples /1.Basics/blink
Build blink : Sketch/Verify / Compile
Flash the micro-controller : File/Upload
```

The toolchain is considered valid if one LED flashes every second. Your card has been flashed and the program is running fine : our congratulations !

The link is not established if you got a message like :

```
error "avrdude: stk500 getsync(): not in sync: resp=0x00"
```

If you use a Seeeduino, check the switch M RST A must be set to A (pos. inside the card). This switch allows automated RESET, which is mandatory to program the card.

3.2 AVR-Ada

❑ WinAVR

At first, download :

```
http://sourceforge.net/projects/winavr/files/WinAVR/20100110/WinAVR-20100110-install.exe/download
```

Launch WinAVR-20100110-install.exe and install it in c:\WinAVR.

❑ AVR-Ada

Then, download :

```
http://sourceforge.net/projects/avr-ada/files/avr-ada/bin_windows/AVR-Ada-1.1.0.exe/download
```

Launch AVR-Ada-1.1.0.exe and install it in c:\WinAVR.

❑ Using ATmega1280

In order to build ATmega1280 runtime, you must :

```
C:> cd c:\WinAVR-20100110\lib\gnat
C:> mkdir avr_lib\atmega1280\lib
C:> avr-gnatmake -XMCU=atmega1280 -P avr.gpr
```

4 Ada Toolchain with Arduino card validation

4.1 Important note

When flashing AVR-Atmel, the common problem is around the serial link speed between the workstation (i.e. the AvrDude flasher) and your card. For many of them, the speed is 19200 bps, but for the Seeeduino Mega, the speed must be 57600 bps. If the speed is incorrect, you will see in the console a messsage like :

```
error "avrdude: stk500_getsync(): not in sync: resp=0x00"
```

4.2 First Ada test

Create a directory tree like c:\ada\blinky and write thoses following files inside it.

stef[@|.]genesix.org

❑ Source

c:\ada\blinky\blinky.adb

```
with AVR.MCU;
with AVR.Wait;
use AVR;
procedure Blinky is
procedure Delay_MS(MS : Natural) is
begin
for X in 1..MS loop
AVR.Wait.Wait_4_Cycles(8000);
end loop;
end;
-- Diode "D13" sur la pin 7 du port B
LED : Boolean renames MCU.PortB_Bits(7);
begin
MCU.DDRB_Bits := (others => DD_Output);
loop
LED := True;
Delay_MS(100);
LED := False;
Delay_MS(100);
end loop;
end Blinky;
```

❑ Project

c:\ada\blinky\blinky.gpr

```
with "avr.gpr";
project Blinky is
package Compiler renames AVR.Compiler;
package Builder renames AVR.Builder;Chapter 5: Install and using under Windows
package Binder
package Linker
8
renames AVR.Binder;
renames AVR.Linker;
for Exec_Dir use ".";
for Source_Files use ("blinky.adb");
for Main use ("blinky.adb");
end Blinky;
```

❑ Makefile

c:\ada\blinky\makefile

```
# Makefile Blinky
all:
avr-gnatmake -g -XMCU=atmega1280 -Pblinky.gpr
avr-objcopy -O ihex blinky.elf blinky.hex
clean:
rm -f b~*.ad[sb] *.o *.ali *.hex *.elf
burn:
avrdude -C c:/WinAVR/bin/avrdude.conf -v -p m1280 \
-c arduino -P COM4 -b 57600 \
-F -U flash:w:blinky.hex
dump:
avr avr-objdump -mavr:5 -d blinky.elf
# eof
```

➪ You should modify makefile to your needs adjusting parameters -XMCU=, -p, -P, -b.

stef[@|.]genesix.org

❑ Compile

WinAVR has all console tools needed inside and pathes are updated at install, so you just have to open a console inside c:\ada\blinky and run the command make :

```
c:\ada\blinky> make

avr-gnatmake -g -XMCU=atmega1280 -Pblinky.gpr avr-gcc -c \
--RTS=rts/avr5 -gnatec=C:\WinAVR\lib\gnat\gnat.adc -gdwarf-2 \
-gnatwp -gnatwu -gnatn -gnatp -gnatVn -Os -gnatef -fverbose-asm \
-frename-registers -mmcu=atmega1280 -gnateDMCU=atmega1280 \
-fdata-sections -ffunction-sections -g -I- -gnatA c:\blinky.adb

avr-gnatbind --RTS=rts/avr5 -freestanding -I- -x c:\blinky\blinky.ali

avr-gnatlink c:\blinky\blinky.ali -Wl,--gc-sections -gdwarf-2 \
-Wl,--relax "--GCC=avr-gcc -Os -mmcu=atmega1280 --RTS=rts/avr5 \
-fdata-sections -ffunction-sections" -g \
-LC:\WinAVR\lib\gnat\avr_lib\atmega1280\lib -lavrada -o c:\blinky\blinky.elf

avr-objcopy -O ihex blinky.elf blinky.hex
```

❑ Run

```
c:\ada\blinky> avrdude -C C:\WinAVR\bin\avrdude.conf -v -v -p m1280 -c arduino -P COM4 -b 57600 -F
-U flash:w:blinky.hex

avrdude: Version 5.10, compiled on Jan 19 2010 at 10:45:23
         Copyright (c) 2000-2005 Brian Dean, http://www.bdmicro.com/
         Copyright (c) 2007-2009 Joerg Wunsch

         System wide configuration file is "C:\WinAVR\bin\avrdude.conf"

         Using Port                    : COM4
         Using Programmer              : arduino
         Overriding Baud Rate          : 57600
         AVR Part                      : ATMEGA1280
         Chip Erase delay              : 9000 us
         PAGEL                         : PD7
         BS2                           : PA0
         RESET disposition             : dedicated
         RETRY pulse                   : SCK
         serial program mode           : yes
         parallel program mode         : yes
         Timeout                       : 200
         StabDelay                     : 100
         CmdexeDelay                   : 25
         SyncLoops                     : 32
         ByteDelay                     : 0
         PollIndex                     : 3
         PollValue                     : 0x53
         Memory Detail                 :

                                  Block Poll               Page                       Polled
           Memory Type Mode Delay Size  Indx Paged  Size   Size #Pages MinW  MaxW   ReadBack
           ----------- ---- ----- ----- ---- ------ ------ ---- ------ ----- ----- ---------
           eeprom        65    10     8    0 no       4096    8      0  9000  9000 0x00 0x00
           flash         65    10   256    0 yes    131072  256    512  4500  4500 0x00 0x00
           lfuse          0     0     0    0 no          1    0      0  9000  9000 0x00 0x00
           hfuse          0     0     0    0 no          1    0      0  9000  9000 0x00 0x00
           efuse          0     0     0    0 no          1    0      0  9000  9000 0x00 0x00
           lock           0     0     0    0 no          1    0      0  9000  9000 0x00 0x00
           calibration    0     0     0    0 no          1    0      0     0     0 0x00 0x00
           signature      0     0     0    0 no          3    0      0     0     0 0x00 0x00

         Programmer Type : Arduino
         Description     : Arduino
         Hardware Version: 2
         Firmware Version: 1.16
         Vtarget         : 0.0 V
         Varef           : 0.0 V
         Oscillator      : Off
```


stef[@|.]genesix.org

```
         SCK period      : 0.1 us

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.05s

avrdude: Device signature = 0x1e9703
avrdude: safemode: lfuse reads as 0
avrdude: safemode: hfuse reads as 0
avrdude: safemode: efuse reads as 0
avrdude: NOTE: FLASH memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "blinky.hex"
avrdude: input file blinky.hex auto detected as Intel Hex
avrdude: writing flash (346 bytes):

Writing | ################################################## | 100% 0.16s

avrdude: 346 bytes of flash written
avrdude: verifying flash memory against blinky.hex:
avrdude: load data flash data from input file blinky.hex:
avrdude: input file blinky.hex auto detected as Intel Hex
avrdude: input file blinky.hex contains 346 bytes
avrdude: reading on-chip flash data:

Reading | ################################################## | 100% 0.14s

avrdude: verifying ...
avrdude: 346 bytes of flash verified

avrdude: safemode: lfuse reads as 0
avrdude: safemode: hfuse reads as 0
avrdude: safemode: efuse reads as 0
avrdude: safemode: Fuses OK

avrdude done.  Thank you.
```

You should see now a beautiful blinking LED !

❑ Output

Refer to the appendix for a disasembled listing of this program.

4.3 Mixed language test

Ada is at its best when we need to connect heterogeneous languages together. Thanks to Ada, link Fortran, Cobol and Pascal together - as we do every day ;) - is a routine task.

The only language that matters to me is the C (I will not re-implement in Ada a vocoder, crypto stuff or signal processing functions written in C that have been validated).

However, we find more and more stuff in Ada for embedded (a full FS FAT12/16/32, all the current crypto stuff AES, DES, etc..., even some light - dBase compatible - Database - , and interfaces routines with common components like LCD displays, environment sensors, ethernet chip, etc...

Same example as the first test but this time a blinking period variable is exported from Ada to C, then this variable is read by a C function called from Ada.

❑ Source Ada

blinky.adb

```
with AVR.MCU;
```

```
with AVR.Wait;

use AVR;

procedure Blinky is

    -- D??clare un entier Ada Delay_Value
    -- et exporte un entier C correspondant delay_value_from_ada
    with AVR.MCU;
with AVR.Wait;

use AVR;

procedure Blinky is

    -- D??clare un entier Ada Delay_Value
    -- et exporte un entier C correspondant delay_value_from_ada

    Delay_Value : Integer := 500;
    pragma Export (C, Delay_Value, "delay_value_from_ada");

    -- D??clare une sp??cification de fonction Ada Get_Delay_From_C
    -- et utilise une fonction C read_delay comme impl??mentation.

    function Get_Delay_From_C return Integer;
    pragma Import (C, Get_Delay_From_C, "read_delay");


    procedure Delay_MS(MS : Natural) is
    begin
        for X in 1..MS loop
            AVR.Wait.Wait_4_Cycles(8000);
        end loop;
    end;

    LED : Boolean renames MCU.PortB_Bits(7);

begin

    MCU.DDRB_Bits := (others => DD_Output);

    loop
        LED := True;
        Delay_MS(Get_Delay_From_C);
        LED := False;
        Delay_MS(Get_Delay_From_C);
    end loop;

end Blinky;
```

❑ Source C

foreign.c

```
// foreign.c

// delay_value_from_ada is declared in blinky.adb

extern int delay_value_from_ada;

int read_delay (void)
    {
    return delay_value_from_ada;
    }

// eof foreign.c
```

❑ Project

blinky.gpr

```
with "avr.gpr";

project Blinky is
```


stef[@|.]genesix.org

```
    package Compiler renames AVR.Compiler;
    package Builder  renames AVR.Builder;
    package Binder   renames AVR.Binder;
    package Linker   renames AVR.Linker;

    for Exec_Dir use ".";
    for Source_Files use ("blinky.adb");
    for Main use ("blinky.adb");

end Blinky;
```

❑ Makefile

```
makefile

# Makefile Blinky

all:
    avr-gcc -c foreign.c
    avr-gnatmake -g -XMCU=atmega1280 -Pblinky.gpr -largs foreign.o
    avr-objcopy -O ihex blinky.elf blinky.hex

clean:
    rm -f b~*.ad[sb] *.o *.ali *.hex *.elf

burn:
    avrdude -C c:/WinAVR/bin/avrdude.conf -v -p m1280 \
        -c arduino -P COM4 -b 57600 \
        -F -U flash:w:blinky.hex

dump:
    avr-objdump -mavr:5 -d blinky.elf

# End
---------- Buffer: makefile ----------
```

The makefile is not there to manage the compilation and the links edition but is rather used as « comfort » function. All the complex mechanics (generally devolved to this conceptual dummy automake when one dev in C) is managed by gnatmake.

The dependencies are automatically managed. It is impossible in Ada to create a binary with dependency issues. C and Java developers will appreciate.

5 Ada Toolchain with Jtagice mkII

5.1 Introduction

In this section, the Jtagice MkII is used for programming and tracing, using serial link instead of USB (more error prone).

If you use Arduino boards, the Arduino bootloader will be deleted for sure.

5.2 Example

Changes to Cygwin cause the standard com port names like « com1 » to crash avarice. This standard naming convention worked in earlier versions. However later builds after 2004 will crash with a « JTAG ICE communication failed: Inappropriate ioctl for device » error when using this old syntax.

Under Windows, command strings to avarice must follow the com port naming convention that Cygwin understands. Cygwin support 16 com ports and these are « zero based » rather than « one based » like Windows. Which means :
- For com1 use /dev/ttyS0 ;
- For com16 use /dev/ttyS15

Debug session example with COM5 (/dev/ttyS4) port com :

```
C:> avarice --file blinky.elf --part atmega1280 --jtag /dev/ttyS4 :4242 --reset-srst --mkII

AVaRICE version 2.9, Jan  7 2010 22:42:57

Defaulting JTAG bitrate to 250 kHz.

JTAG config starting.
Found a device: JTAGICEmkII
Serial number:  07:00:00:00:61:f8
Reported JTAG device ID: 0x9703
Configured for device ID: 0x9703 atmega1280 -- Matched with atmega1280
JTAG config complete.
Preparing the target device for On Chip Debugging.

Disabling lock bits:
  LockBits -> 0xff

Enabling on-chip debugging:
  Extended Fuse byte -> 0xfd
      High Fuse byte -> 0x1a
       Low Fuse byte -> 0xff
WARNING: The default behaviour has changed.
Programming no longer erases by default. If you want to erase and program
in a single step, use the --erase in addition to --program. The reason for
this change is to allow programming multiple sections (e.g. application and
bootloader) in multiple passes.

Downloading FLASH image to target...

Download complete.
Waiting for connection on port 4242.
Connection opened by host 127.0.0.1, port 3760.
```

For reference, under Windows, USB debug session test (serial 61F8) have failed with a command as below :

```
C:> avarice --file blinky.elf --part atmega1280 --jtag usb:61F8 --reset-srst --mkII :4242
```

6 Build

Anything that begins well ends badly. Anything that begins badly ends worse.
Pudder's laws

Therefore, if everything is going right, something is wrong.
Anonymous

6.1 Introduction

Some little things to remember : build under Windows could be tricky and, also, build under Windows could be tricky. From now on, anything may happen. Even you may have success (or

not). Retry with another workstation, with another Windows, with another mouse, with another feeling. I can't explain © The Who.

6.2 Build - Step 1 : basic environment

Create two new directories and go inside the first of it :

```
C:> mkdir c:\MinGW-install1
C:> mkdir c:\MinGW-install2
C:> c:
C:> cd \MinGW-install1
```

Download http://downloads.sourceforge.net/mingw/MinGW-5.1.4.exe.
Then install it with full version option in c:\MinGW.

Download http://downloads.sourceforge.net/mingw/MSYS-1.0.10.exe.
Then install it in c:\MinGW. At the end of the setup, choose *post install = y*.

In order to avoid errors, you may (if you need it) limit PATH. In line 18 of c:\minGW\etc\profile, replace :

c:\minGW\etc\profile

```
if [ $MSYSTEM == MINGW32 ]; then
  export PATH=".:/usr/local/bin:/mingw/bin:/bin:$PATH"
else
  export PATH=".:/usr/local/bin:/bin:/mingw/bin:$PATH"
fi

by

if [ $MSYSTEM == MINGW32 ]; then
  export PATH=".:/usr/local/bin:/mingw/bin:/bin"
else
  export PATH=".:/usr/local/bin:/bin:/mingw/bin"
fi
```

6.3 Build - Step 2 : compiler environment

Go inside the second directory :

```
C:> c:
C:> cd \MinGW-install2
```

In that order (an important detail) download and install theses packages in c:\MinGW, confirm all overwrite files :

http://downloads.sourceforge.net/mingw/msysCORE-1.0.11-20080826.tar.gz
http://downloads.sourceforge.net/mingw/msysDTK-1.0.1.exe
http://downloads.sourceforge.net/mingw/gcc-4.3.0-20080502-mingw32-alpha-bin.7z
http://downloads.sourceforge.net/mingw/cvs-1.11.22-MSYS-1.0.11-1-bin.tar.gz
http://downloads.sourceforge.net/mingw/flex-2.5.33-MSYS-1.0.11-1.tar.bz2
http://downloads.sourceforge.net/mingw/bison-2.3-MSYS-1.0.11-1.tar.bz2
http://downloads.sourceforge.net/mingw/regex-0.12-MSYS-1.0.11-1.tar.bz2

http://downloads.sourceforge.net/mingw/gettext-0.16.1-MSYS-1.0.11-1.tar.bz2
http://downloads.sourceforge.net/mingw/texinfo-4.11-MSYS-1.0.11-1.tar.bz2
http://downloads.sourceforge.net/sourceforge/mingw/tar-1.13.19-MSYS-2005.06.08.tar.bz2
http://downloads.sourceforge.net/mingw/msys-autoconf-2.59.tar.bz2
http://downloads.sourceforge.net/mingw/msys-automake-1.8.2.tar.bz2
http://downloads.sourceforge.net/mingw/crypt-1.1-1-MSYS-1.0.11-1.tar.bz2[23]
http://sourceforge.net/projects/win32svn/files/1.7.2/svn-win32-1.7.2.zip/download[24]

❑ Wget

Download : https://sourceforge.net/project/downloading.php?group_id=2435&filename=@/wget-1.9.1-mingwPORT.tar.bz2&a=85590697

Then extract the file wget-1.9.1\mingwPORT\wget.exe from archive to c:\MinGW\bin.

6.4 Build - Step 3 : Prepare the build

Create a place to host the build :

```
C:> mkdir c:\avr-ada
C:> mkdir c:\avr-ada\src
C:> c:
C:> cd \avr-ada
```

Lauch Msys, then at the bash prompt go inside it :

```
-- Note the /c/ notation in place of c:\
root:~> cd /c/avr-ada
```

Then get an up-to-date build script : http://avr-ada.svn.sourceforge.net/viewvc/avr-ada/tags/avr-ada-1_1_0/tools/build/build-avr-ada-gcc-4.3.x.sh?revision=1025

This script must be located at c:\avr-ada\build-avr-ada-gcc-4.3.x.sh

You also need binutils-2.19.tar.bz2, but this release has been replaced by binutils-2.19.1.tar.bz2 almost everywhere. Here two valid sources (at the beginning of 2012) :

http://pkgs.fedoraproject.org/repo/pkgs/msp430-binutils/binutils-2.19.tar.bz2/17a52219dee5a76c1a9d9b0bfd337d66/binutils-2.19.tar.bz2

https://www.fooe.net/trac/llvm-msp430/browser/trunk/mspgcc/gnu/binutils-2.19.tar.bz2?rev=79

Finally, You must place binutils-2.19.tar.bz2 in c:\avr-ada\src.

[23] To avoid msys-crypt-0.dll missing error.

[24] To allow patches extract.

At the line 140 from the file build/build-avr-ada-gcc-4.3..sh, replace :

```
CC=gcc-4.3
by
CC=gcc
```

At the line 279 from the file build/build-avr-ada-gcc-4.3.x.sh, you may delete the line below or you might prefer leave this line unaffected provided that c:\avr-ada\src\binutils-2.19.tar.bz2 exist.

$WGET "ftp://anonymous:fireftp@@mirrors.kernel.org/gnu/binutils/$FILE_BINUTILS.tar.bz2"

6.5 Build - Step 4 : Really build (or not)

Now, you can launch build-avr-ada-gcc-4.3.x.sh

But, before doing that, be sure to remember the introduction. Also it took time, could be more than a couple of hours with an old workstation. Have confidence in cooling your pc, even a well cooled one (see below).

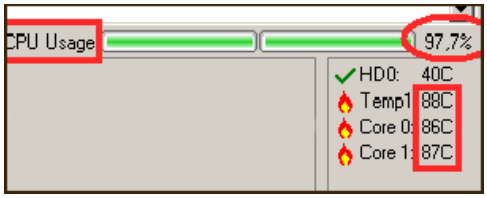

Fig. 4: Some like it hot

stef[@|.]genesix.org

Real world testbed

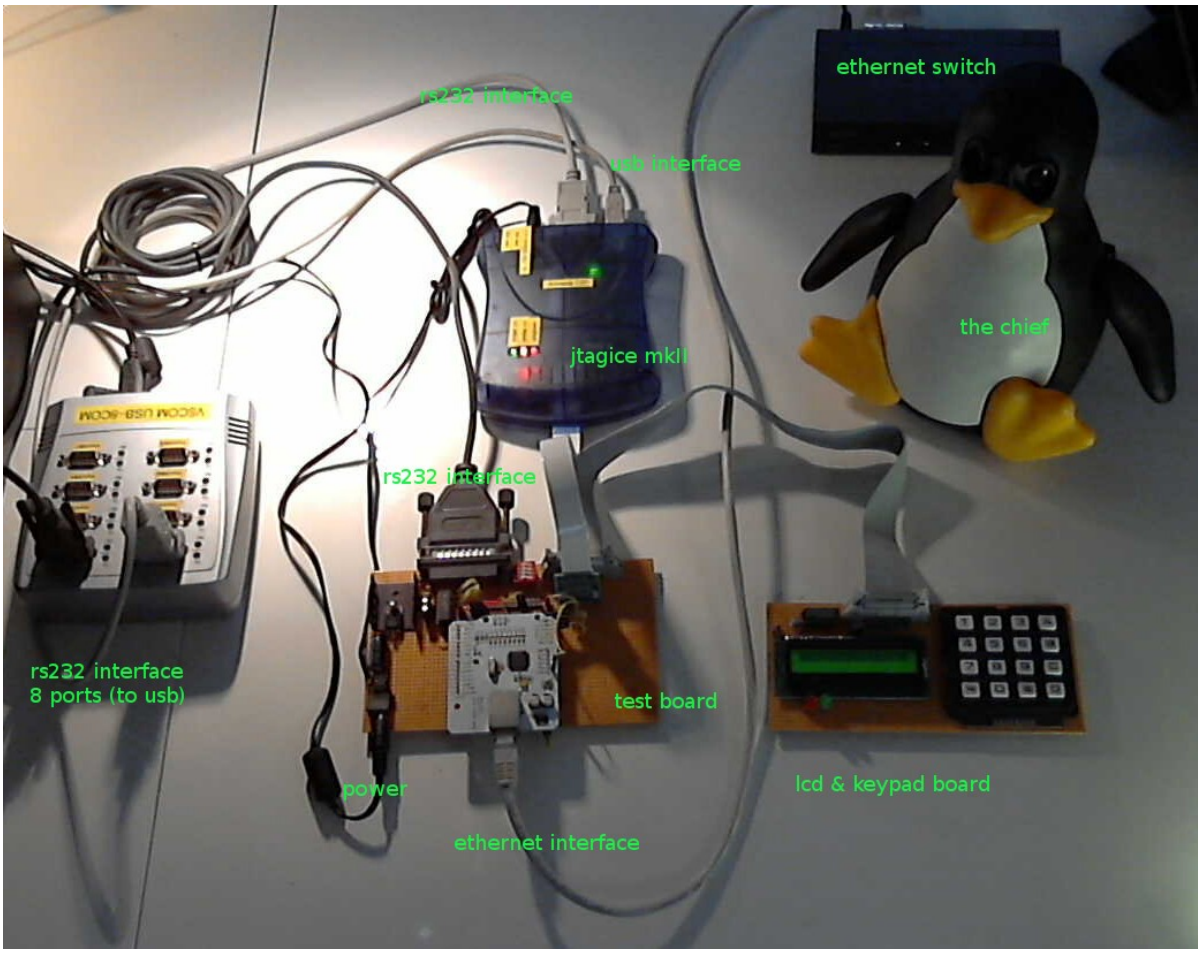

Fig. 5: A real world testbed, supervised by the chef himself.

<<<TODO>>>

References

There are 10 types of people in the world : those who understand binary and those who don't.
Anonymous

1 Links

1.1 Ada

https://en.wikibooks.org/wiki/Ada_Programming

1.2 Hardware

❑ Seeduino

If you are considering buying an Arduino Mega, look at the Seeeduino Mega which owns more I/O (in fact, a lot more) and comes with other goodies.

Browse http://www.seeedstudio.com for details.

❑ Ethernet

You may use board with the so-called Wiz5100. This chip comes with a full built-in IP/TCP/UDP layer inside ! Tero has wrote Ada code for it.

1.3 Compiling Linux toolchain from sources

https://sourceforge.net/p/avr-ada/wiki/Setup

http://arduino.ada-language.com/building-avr-gnat-for-avr-ada.html
http://arduino.ada-language.com/avr-ada-package-for-ubuntu-1404.html
https://bitbucket.org/tkoskine/avr-ada-build-script/src

https://www.weeoak.com/content/building-an-ada-toolchain-for-avr-arduino-on-ubuntu-1404-lts-x64

http://lvcetada.e-monsite.com/pages/installation/installation-avr-ada-sur-ubuntu-12-04.html

http://avr-ada-devel.narkive.com/7nn9RTDd/avr-ada-getting-to-square-one-on-debian-ubuntu

http://arduino.ada-language.com/avr-ada-package-for-debian-wheezy-7x-on-raspberry-pi-1.html

1.4 Programming and debugging

Very useful link (toolchain compilation and use) :
http://uracoli.nongnu.org/avrtools.html

AVR Debugging on Linux (with debugWire) :
http://www.luniks.net/avr-debug.jsp

A video using DDD as a graphic frontend to GDB with JTAGICE mkII :
http://delvsie.com/download/IwPWq9m0M6w/debugging-an-avr-under-linux-using-the-jtag-ice-mkii-and-gdb

Mode videos about GDB :
http://delvsie.com/list/gdb-debugger-unix

Detailled JTAGICE mkII fuse setting, programming and debugging session :
http://ardupilot.org/dev/docs/jtag.html

Problem with variables not updated in debug session (test it to validate debugging with AVR) :
https://groups.google.com/forum/#!topic/comp.lang.ada/Z3FPId6nKMY

1.5 Goodies

Fuse calculator :
http://www.engbedded.com/fusecalc

Useful tips & trics :
http://www.ladyada.net/resources/ucannoyances.html

1.6 Others

Avr Freaks : http://www.avrfreaks.net

Adacore Github : https://github.com/AdaCore
Adacore Papers : https://www.adacore.com/papers
Adacore Gems : https://www.adacore.com/gems
Adacore Books : https://www.adacore.com/books
Adacore GPL : https://www.adacore.com/community

http://www.tldp.org/HOWTO/Avr-Microcontrollers-in-Linux-Howto/x207.html

http://www.avrfreaks.net/forum/i-didnt-know-you-could-get-ada-avr

1.7 People

❑ Ludovic Breta

Ada Debian Maintainer, and a good friend too.
https://people.debian.org/~lbrenta/debian-ada-policy.html

❑ Rolf Ebert

The AVR-Ada project leader, since the beginning.

https://sourceforge.net/projects/avr-ada

❑ Tero Koskinen

Tero is very involved in AVR-Ada development. The resources he provides are of great value.

Home site : http://tkoskine.me
Ada related site : http://ada-language.com
Ada related site :http://stronglytyped.org
Sources repository : https://bitbucket.org/tkoskine

http://ubuntu.ada-language.com/14.04/avr-ada_1.2.2_amd64.deb

❑ Stéphane Carrez

Member of Ada-France.

Blog : https://blog.vacs.fr
Sources repository : https://github.com/stcarrez

❑ Frédéric Praca

Member of Ada-France.

Blog : https://plus.google.com/111458939618003185072

stef[@|.]genesix.org

Appendix

1 build-avr-ada-aide.sh listing

./buil-avr-ada-aide.sh

```
#!/bin/bash
#-------------------------------------------------------------------------------
# build-avr-ada-aide.sh - A script to build AVR-Ada-JLF and AIDE-AVR8 projects
#-------------------------------------------------------------------------------
#
# Original project : AVR-Ada - A GCC Ada environment for AVR-Atmel
# Copyright (C) 2003...2017 Rolf Ebert and AVR-Ada team - https://sourceforge.net/projects/avr-ada
#
# For project : AVR-Ada-JLF (John Leimon Fork) - An up-to-date GCC Ada environment based on AVR-Ada
# Copyright (C) 2015 John Leimon - https://github.com/evilspacepirate/avr-ada
#
# This file : build-avr-ada-aide.sh
# Copyright (C) 2014, 2015 Tero Kostinen - https://bitbucket.org/tkoskine
# Copyright (C) 2015 John Leimon - https://github.com/evilspacepirate/avr-ada
# Copyright (C) 2017 Stephane Riviere - https://stef.genesix.org
#
#-------------------------------------------------------------------------------
#
#  Permission to use, copy, modify, and/or distribute this software for any purpose
#  with or without fee is hereby granted, provided that the above copyright notice and
#  this permission notice appear in all copies.
#
#  THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
#  TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN
#  NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR
#  CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR
#  PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
#  ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
#
#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# INIT
#-------------------------------------------------------------------------------

VER=1.0

#--- Uncomment for script verbose mode
#set -x

#--- Preferred directory is...
#cd /usr/local/bin

#--- Build root directory (current used)
ORGDIR=$PWD

#--- Target dir (gitlab project name)
TGTDIR=avr-ada

#--- Build root dir
TOPDIR=$ORGDIR/$TGTDIR

#-------------------------------------------------------------------------------
# service functions
#-------------------------------------------------------------------------------

pause()
{
read -n1 -r -p "Press any key to continue..." key
}

log_msg()
{
  echo `date +%Y%m%d-%H%M%S` " - MESSAGE : " $@
  echo `date +%Y%m%d-%H%M%S` " - MESSAGE : " $@ >> $LOGFILE
}

log_err()
{
  echo `date +%Y%m%d-%H%M%S` " - ERROR   : " $@
  echo `date +%Y%m%d-%H%M%S` " - ERROR   : " $@ >> $LOGFILE
  log_end
  exit 1
```

```
}

log_ini()
{

#--- Create log file
LOGFILE=$ORGDIR/`date +%Y%m%d-%H%M`.log
echo "" > $LOGFILE

log_msg "================================================================================"
log_msg "BUILD START"
log_msg "================================================================================"
}

log_end()
{
log_msg "================================================================================"
log_msg "BUILD END"
log_msg "================================================================================"

cd $ORGDIR
}

#-------------------------------------------------------------------------------
check_system()
#-------------------------------------------------------------------------------

{
  log_msg "-----------------------------------------"
  log_msg "CHECK SYSTEM"
  log_msg "-----------------------------------------"

  #--- Search for distribution
  LSB_REL=`lsb_release -i|tr "\t" " "|tr -s " "|cut -f3 -d" "`
  if [ x"$LSB_REL" != x"" ]; then
     DISTRO=$LSB_REL;
  else
     DISTRO=unknown
  fi

  #--- Check for a Gnat compiler
  gnat > /dev/null 2>&1
  if [ $? -ne 0 ]
  then
    log_err "GNAT missing, building AVR-Ada requires a GNAT compiler."
  fi

  CONFIG_GNAT=`gnat 2>/dev/null | sed -n 1p | grep -wo -E "[0-9]*\.{1}[0-9]*\.*[0-9]*"`
  CONFIG_GNAT_MAJOR=`gnat 2>/dev/null | sed -n 1p | grep -wo -E "[0-9]*\.{1}[0-9]*"`
  CONFIG_GNAT_SHORT=`echo $CONFIG_GNAT_MAJOR | sed 's/\.//'`

  GNATPREFIX=$TOPDIR/gnat-native-$CONFIG_GNAT_SHORT
  AVRADAPREFIX=$TOPDIR/avr-ada-$CONFIG_GNAT_SHORT

  GCC_VERSION=$CONFIG_GNAT
  AVR_GCC_VERSION=$CONFIG_GNAT

  #--- System and build information
  log_msg "Distribution : "$DISTRO"."
  log_msg "Log file : $LOGFILE"
  log_msg "Build directory : $TOPDIR"
  log_msg "Build type : $BUILDTYPE"
  log_msg "GNAT version : $CONFIG_GNAT"
}

#-------------------------------------------------------------------------------
check_files()
#-------------------------------------------------------------------------------

{
  log_msg "-----------------------------------------"
  log_msg "CHECK FILES"
  log_msg "-----------------------------------------"

  #--- Search for AVR-ADA-JLF project
  if [ ! -d $TOPDIR/avr-ada ]; then
    git clone --verbose https://github.com/evilspacepirate/avr-ada
    if [[ "$?" != 0 ]]; then
       log_err "Error downloading AVR-Ada-JLF project."
    fi
```


stef[@|.]genesix.org

```
  else
    log_msg "AVR-Ada-JLF project already downloded, skipping."
  fi

  #-- Go to build directory
  cd $TOPDIR

  #--- Search for TEXINFO.PATCH
  if [ ! -f $TOPDIR/texinfo.patch ]; then
    log_err "Could not find texinfo.patch."
  fi

  #--- Search for TEXINFO2.PATCH
  if [ ! -f $TOPDIR/texinfo2.patch ]; then
    log_err "Could not find texinfo2.patch."
  fi

  #--- Search for AVR-THREADS.DIFF
  if [ ! -f $TOPDIR/avr-threads.diff ]; then
    log_err "Could not find 'avr-threads.diff' patch."
  fi

  #--- Search for GCC
  if [ ! -f gcc-${GCC_VERSION}.tar.gz ]; then
     wget ftp://ftp.mirrorservice.org/sites/sourceware.org/pub/gcc/releases/gcc-${GCC_VERSION}/gcc-$
{GCC_VERSION}.tar.gz
    if [[ "$?" != 0 ]]; then
       log_err "Error downloading file gcc-${GCC_VERSION}.tar.gz"
    fi
  else
    log_msg "gcc-${GCC_VERSION}.tar.gz already downloaded, skipping."
  fi

  #--- Note that in this release GCC_VERSION=AVR_GCC_VERSION so this part could be deleted
  if [ ! -f gcc-${AVR_GCC_VERSION}.tar.gz ]; then
                    wget     ftp://ftp.mirrorservice.org/sites/sourceware.org/pub/gcc/releases/gcc-$
{AVR_GCC_VERSION}/gcc-${AVR_GCC_VERSION}.tar.gz
    if [[ "$?" != 0 ]]; then
       log_err "Error downloading file gcc-${AVR_GCC_VERSION}.tar.gz"
    fi
  else
    log_msg "gcc-${AVR_GCC_VERSION}.tar.gz already downloaded, skipping."
  fi

  #--- Search for BINUTILS
  if [ ! -f binutils-2.20.1a.tar.bz2 ]; then
    wget https://ftp.gnu.org/gnu/binutils/binutils-2.20.1a.tar.bz2
    if [[ "$?" != 0 ]]; then
       log_err "Error downloading file binutils-2.20.1a.tar.bz2"
    fi
  else
    log_msg "binutils-2.20.1a.tar.bz2 already downloaded, skipping."
  fi

  #--- Search for AVR-LIBC
  if [ ! -f avr-libc-1.8.0.tar.bz2 ]; then
    wget http://download.savannah.gnu.org/releases/avr-libc/old-releases/avr-libc-1.8.0.tar.bz2
    if [[ "$?" != 0 ]]; then
       log_err "Error downloading file avr-libc-1.8.0.tar.bz2"
    fi
  else
    log_msg "avr-libc-1.8.0.tar.bz2 already downloaded, skipping."
  fi
}

#-------------------------------------------------------------------------------
build_gcc()
#-------------------------------------------------------------------------------

{
  log_msg "----------------------------------------"
  log_msg "BUILD GCC-${GCC_VERSION}"
  log_msg "----------------------------------------"

  if [ -f "$GNATPREFIX/bin/gnatmake" ]; then
    log_msg "GCC ${GCC_VERSION} is already build, skipping."
    return 0
  fi

  #--- Uncompress GCC archive
```

```
  if [ ! -d gcc-${GCC_VERSION} ]; then
    tar zxf gcc-${GCC_VERSION}.tar.gz
  else
    log_msg "GCC gcc-${GCC_VERSION} exists, skipping un-archive."
  fi

  cd gcc-${GCC_VERSION}

  #--- Determine if we need to apply texinfo.patch
  patch -p0 --dry-run --silent < $TOPDIR/texinfo.patch 2>&1 > /dev/null
  if [ $? -eq 0 ]; then
    # Apply the patch #
    log_msg "Applying texinfo.patch"
    patch -p0 -N < $TOPDIR/texinfo.patch || log_err "GCC texinfo.patch not applied."
  else
    log_msg "Texinfo is already patched, skipping patch with texinfo.patch"
  fi

  #--- Determine if we need to apply texinfo2.patch
  patch -p0 --dry-run --silent < $TOPDIR/texinfo2.patch 2>&1 > /dev/null
  if [ $? -eq 0 ]; then
    # Apply the patch #
    log_msg "Applying texinfo2.patch"
    patch -p0 -N < $TOPDIR/texinfo2.patch || log_err "GCC texinfo2.patch not applied."
  else
    log_msg "Texinfo is already patched, skipping patch with texinfo2.patch"
  fi

  cd ..

  rm -rf gcc-obj-${GCC_VERSION}
   mkdir gcc-obj-${GCC_VERSION} && cd gcc-obj-${GCC_VERSION} || log_err "cd gcc-obj-${GCC_VERSION}
failed."

  #--- i386 Hardware
  if [ -d /usr/lib/i386-linux-gnu/ ]; then
    export LIBRARY_PATH=/usr/lib/i386-linux-gnu/
  fi
  if [ -d /usr/include/i386-linux-gnu ]; then
    export C_INCLUDE_PATH=/usr/include/i386-linux-gnu
    export CPATH=/usr/include/i386-linux-gnu
    export CPLUS_INCLUDE_PATH=/usr/include/i386-linux-gnu
  fi

  #--- x86_64 Hardware
  if [ -d /usr/lib/x86_64-linux-gnu/ ]; then
    export LIBRARY_PATH=/usr/lib/x86_64-linux-gnu/
  fi
  if [ -d /usr/include/x86_64-linux-gnu ]; then
    export C_INCLUDE_PATH=/usr/include/x86_64-linux-gnu
    export CPATH=/usr/include/x86_64-linux-gnu
    export CPLUS_INCLUDE_PATH=/usr/include/x86_64-linux-gnu
  fi

  log_msg "LIBRARY_PATH       = '$LIBRARY_PATH'"
  log_msg "C_INCLUDE_PATH     = '$C_INCLUDE_PATH'"
  log_msg "CPLUS_INCLUDE_PATH = '$CPLUS_INCLUDE_PATH'"
  log_msg "CPATH              = '$CPATH'"

  case "$DISTRO" in
    Debian|Ubuntu|LinuxMint)
          ADAC=gcc-$CONFIG_GNAT_MAJOR  CC=gcc-$CONFIG_GNAT_MAJOR  ../gcc-${GCC_VERSION}/configure
--enable-languages=c,ada  --prefix=$GNATPREFIX  --disable-multilib  --without-cloog  --without-ppl  ||
log_err "gcc: configure"
      ;;
    *)
        ../gcc-${GCC_VERSION}/configure  --enable-languages=c,ada  --prefix=$GNATPREFIX  ||  log_err
"gcc: configure"
      ;;
  esac

  make bootstrap || log_err "GCC make bootstrap failed."
  make install || log_err "GCC make install failed."

  cd ..

}

#-----------------------------------------------------------------------------
build_avrbinutils()
```


stef[@|.]genesix.org

```
#-------------------------------------------------------------------------------

{
  log_msg "----------------------------------------"
  log_msg "BUILD AVR-BINUTILS"
  log_msg "----------------------------------------"

  cd $TOPDIR
  if [ -f $AVRADAPREFIX/bin/avr-as ]; then
    log_msg "BINUTILS already installed, skipping."
    return 0
  fi

  pwd
  ls -l

  tar jxf binutils-2.20.1a.tar.bz2 || log_err "BINUTILS un-archive failed."
  cd binutils-2.20.1
  for a in ../avr-ada/patches/binutils/2.20.1/*.patch;
  do
    patch -p0 < $a || log_err "BINUTILS patch $a failed."
  done
  sed -i -e 's/@colophon/@@colophon/' -e 's/doc@cygnus.com/doc@@cygnus.com/' bfd/doc/bfd.texinfo

  cd ..

  rm -rf binutils-obj
  mkdir binutils-obj && cd binutils-obj || log_err "BINUTILS mkdir/cd failed."

  ../binutils-2.20.1/configure --target=avr \
                               --program-prefix=avr- \
                               --disable-shared \
                               --disable-nls \
                               --enable-commonbfdlib=no \
                               --disable-werror \
                               --prefix=$AVRADAPREFIX || log_err "BINUTILS configure failed."

  sed -i -e 's/SUBDIRS = doc po/SUBDIRS = po/' gas/Makefile
  sed -i -e 's/SUBDIRS = doc po/SUBDIRS = po/' ../binutils-2.20.1/gas/Makefile.am
  sed -i -e 's/SUBDIRS = /SUBDIRS = #/' ../binutils-2.20.1/gas/Makefile.in

  make || log_err "BINUTILS make failed."
  make install || log_err "BINUTILS make failed."
}

#-------------------------------------------------------------------------------
build_avrgcc()
#-------------------------------------------------------------------------------

{
  log_msg "----------------------------------------"
  log_msg "BUILD AVR-GCC-GNAT"
  log_msg "----------------------------------------"

  log_msg "GNATPREFIX   : '$GNATPREFIX'"
  log_msg "AVRADAPREFIX : '$AVRADAPREFIX'"

  cd $TOPDIR
  export PATH="$GNATPREFIX/bin":"$AVRADAPREFIX/bin":$PATH

  mkdir avr
  if [ -d gcc-${AVR_GCC_VERSION} ]; then
    log_msg "Removing old gcc-${AVR_GCC_VERSION} directory."
    rm -rf gcc-${AVR_GCC_VERSION}
  fi

  if [ -f "$AVRADAPREFIX/bin/avr-gnatmake" ]; then
    log_msg "AVR-GCC already installed, skipping."
    return 0
  fi

  cd $TOPDIR/avr

  #--- i386 Hardware
  if [ -d /usr/lib/i386-linux-gnu/ ]; then
    export LIBRARY_PATH=/usr/lib/i386-linux-gnu/
  fi
  if [ -d /usr/include/i386-linux-gnu ]; then
    export C_INCLUDE_PATH=/usr/include/i386-linux-gnu
    export CPATH=/usr/include/i386-linux-gnu
```


stef[@|.]genesix.org

```
    export CPLUS_INCLUDE_PATH=/usr/include/i386-linux-gnu
  fi

  #--- x86_64 Hardware
  if [ -d /usr/lib/x86_64-linux-gnu/ ]; then
    export LIBRARY_PATH=/usr/lib/x86_64-linux-gnu/
  fi
  if [ -d /usr/include/x86_64-linux-gnu ]; then
    export C_INCLUDE_PATH=/usr/include/x86_64-linux-gnu
    export CPATH=/usr/include/x86_64-linux-gnu
    export CPLUS_INCLUDE_PATH=/usr/include/x86_64-linux-gnu
  fi

  tar zxf $TOPDIR/gcc-${AVR_GCC_VERSION}.tar.gz || log_err "AVR-GCC-GNAT un-archive failed."
  cd gcc-${AVR_GCC_VERSION} || log_err "AVR-GCC-GNAT cd failed."

  pwd

  for a in ../../avr-ada/patches/gcc/$AVR_GCC_VERSION/*.patch;do patch -p0 < $a;done

  cd ..

  rm -rf avr-gcc-obj

  mkdir avr-gcc-obj && cd avr-gcc-obj || log_err "AVR-GCC-GNAT mkdir/cd failed."

  ../gcc-${AVR_GCC_VERSION}/configure --target=avr \
                                      --program-prefix=avr- \
                                      --disable-shared \
                                      --disable-nls \
                                      --disable-libssp \
                                      --with-system-zlib \
                                      --disable-libada \
                                      --enable-multilib \
                                      MAKEINFO=missing \
                                      --enable-languages=ada,c,c++ \
                                      --enable-cpp \
                                      --with-dwarf2 \
                                      --enable-version-specific-runtime-libs \
                                          --prefix=$AVRADAPREFIX || log_err "AVR-GCC-GNAT configure
failed."

  sed -i -e 's/$(STAMP)/echo timestamp >/' ../gcc-${AVR_GCC_VERSION}/gcc/config/avr/t-avr

  make || log_err "AVR-GCC-GNAT make failed"
  make install || log_err "AVR-GCC-GNAT make install failed."

  cd $TOPDIR

  if [ ! -f "$AVRADAPREFIX/bin/avr-gnatmake" ]; then
    log_err "AVR-GCC-GNAT build failed."
  fi
}

#---------------------------------------------------------------------------
build_avrlibc()
#---------------------------------------------------------------------------

{
  log_msg "---------------------------------------"
  log_msg "BUILD AVR-LIBC"
  log_msg "---------------------------------------"

  cd $TOPDIR
  export PATH="$AVRADAPREFIX/bin":$PATH

  if [ -f "$AVRADAPREFIX/avr/lib/avr3/libc.a" ]; then
    log_msg "AVR-LIBC installed, skipping."
    return 0
  fi

  rm -rf avr-libc-1.8.0

  tar jxf avr-libc-1.8.0.tar.bz2 || log_err "AVR-LIBC un-archive failed."

  cd avr-libc-1.8.0 || log_err "avr-libc: cd"
  ./configure --host=avr --prefix=$AVRADAPREFIX || log_err "AVR-LIBC configure failed."

  make || log_err "AVR-LIBC make failed."
  make install || log_err "AVR-LIBC make install failed."
```


stef[@|.]genesix.org

```
  cd ..
}

#-------------------------------------------------------------------------------
build_avrada()
#-------------------------------------------------------------------------------

{
  log_msg "----------------------------------------"
  log_msg "BUILD AVR-ADA"
  log_msg "----------------------------------------"

  cd $TOPDIR
  export PATH="$AVRADAPREFIX/bin":$PATH

  cd avr-ada
  patch -p1 --dry-run --silent < $TOPDIR/avr-threads.diff 2>&1 > /dev/null
  if [ $? -eq 0 ]; then
    # Apply the patch #
    echo "Applying AVR threads patch"
    patch -p1 -N < $TOPDIR/avr-threads.diff || log_err "AVR-Ada threads diff failed."
  else
    echo "AVR-ADA threads is already patched, skipping."
  fi
  cd ..

  cd avr-ada || log_err "avr-ada: cd"
  sed -i -e 's/stamp-libs: $(LIB_LIST) $(BOARD_LIB_LIST) $(THREAD_LIB_LIST)/stamp-libs: $(LIB_LIST)
$(BOARD_LIB_LIST)/' avr/avr_lib/Makefile

  ./configure                                          # OPERATION_00

  make GPRCONFIG=/usr/bin/gprconfig GPRBUILD=/usr/bin/gprbuild || log_err "AVR-ADA make failed." #
OPERATION_01

  # Clean up from a previous build #
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr25
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr3
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr31
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr35
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr4
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr5
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr51
  rm -rfv  $AVRADAPREFIX/lib/gcc/avr/$CONFIG_GNAT/avr6

  make install_rts || log_err "AVR-ADA make install_Rts failed."    # OPERATION_02

  cd avr/avr_lib || log_err "AVR-ADA cd avr/avr_lib failed."
  make || log_err "avr-ada: make lib"                       # OPERATION_03

  cd ..
  make install_libs || log_err "AVR-ADA make install_libs failed."  # OPERATION_04

  cd ..
}

#-------------------------------------------------------------------------------
check_build()
#-------------------------------------------------------------------------------

{
  log_msg ""
  log_msg "----------------------------------------"
  if [ -f $AVRADAPREFIX/bin/avr-gnatmake ];then
    log_msg " AVR-Ada BUILD COMPLETE."
    log_msg ""
    log_msg " Binaries are located at :"
    log_msg " $AVRADAPREFIX/bin"
    log_msg "----------------------------------------"
    log_msg "Build end with success."
  else
    log_msg "AVR-Ada BUILD DID NOT COMPLETE SUCESSFULLY"
    log_msg "----------------------------------------"
    log_msg "Build end with error(s)."
  fi
  log_msg "----------------------------------------"
  log_msg ""
}
```


stef[@|.]genesix.org

```
#-------------------------------------------------------------------------------
build_all()
#-------------------------------------------------------------------------------
{

check_system
check_files
build_gcc
build_avrbinutils
build_avrgcc
build_avrlibc
build_avrada
check_build
}

#-------------------------------------------------------------------------------
erase_all()
#-------------------------------------------------------------------------------
{
log_msg "Erasing previous build."
rm -rf $TOPDIR
}

#-------------------------------------------------------------------------------
build_aide()
#-------------------------------------------------------------------------------
{
log_msg "Not implemented yet."
}

#-------------------------------------------------------------------------------
# Main program
#-------------------------------------------------------------------------------

echo ""
echo "BUILD-AVR-Ada-AIDE - A script to build AVR-Ada-JLF and AIDE-AVR8 projects - Version $VER."
echo "Copyright (C) Tero Kostinen, John Leimon, Stephane Riviere 2014-2017 under the GPLv3 licence"
echo ""

case "$1" in
    --build)
        BUILDTYPE="Build AVR-Ada-JLF - Resume from previous build, if exist."
        log_ini
        build_all
        log_end
        ;;
    --fresh)
        BUILDTYPE="Build AVR-Ada-JLF - Full erase previous build, then do a fresh one."
        log_ini
        erase_all
        build_all
        log_end
        ;;
    --aide)
        BUILDTYPE="Build AIDE-AVR8 distribution - An AVR-Ada-JLF built must exist."
        log_ini
        build_aide
        log_end
        ;;
    *)
    echo "Usage : $0 --[build|fresh|aide]"
    echo ""
    echo "--build : Resume from previous build, if exist"
    echo "--fresh : Erase previous build, then make a fresh one"
    echo "--aide  : Build AIDE-AVR8 distribution"
    echo ""
    exit 1
    ;;
esac

exit 0

#-------------------------------------------------------------------------------
# EOF
#-------------------------------------------------------------------------------
```

2 blinky.adb disasembled listing

```
c:\ada\blinky> avr-gnatmake -g -S -fverbose-asm -XMCU=atmega1280 -Pblinky.gpr

blinky.elf:     file format elf32-avr

Disassembly of section .text:

00000000 <__vectors>:
   0:    71 c0           rjmp    .+226        ; 0xe4 <__ctors_end>
   2:    00 00           nop
   4:    8c c0           rjmp    .+280        ; 0x11e <__bad_interrupt>
   6:    00 00           nop
   8:    8a c0           rjmp    .+276        ; 0x11e <__bad_interrupt>
   a:    00 00           nop
   c:    88 c0           rjmp    .+272        ; 0x11e <__bad_interrupt>
   e:    00 00           nop
  10:    86 c0           rjmp    .+268        ; 0x11e <__bad_interrupt>
  12:    00 00           nop
  14:    84 c0           rjmp    .+264        ; 0x11e <__bad_interrupt>
  16:    00 00           nop
  18:    82 c0           rjmp    .+260        ; 0x11e <__bad_interrupt>
  1a:    00 00           nop
  1c:    80 c0           rjmp    .+256        ; 0x11e <__bad_interrupt>
  1e:    00 00           nop
  20:    7e c0           rjmp    .+252        ; 0x11e <__bad_interrupt>
  22:    00 00           nop
  24:    7c c0           rjmp    .+248        ; 0x11e <__bad_interrupt>
  26:    00 00           nop
  28:    7a c0           rjmp    .+244        ; 0x11e <__bad_interrupt>
  2a:    00 00           nop
  2c:    78 c0           rjmp    .+240        ; 0x11e <__bad_interrupt>
  2e:    00 00           nop
  30:    76 c0           rjmp    .+236        ; 0x11e <__bad_interrupt>
  32:    00 00           nop
  34:    74 c0           rjmp    .+232        ; 0x11e <__bad_interrupt>
  36:    00 00           nop
  38:    72 c0           rjmp    .+228        ; 0x11e <__bad_interrupt>
  3a:    00 00           nop
  3c:    70 c0           rjmp    .+224        ; 0x11e <__bad_interrupt>
  3e:    00 00           nop
  40:    6e c0           rjmp    .+220        ; 0x11e <__bad_interrupt>
  42:    00 00           nop
  44:    6c c0           rjmp    .+216        ; 0x11e <__bad_interrupt>
  46:    00 00           nop
  48:    6a c0           rjmp    .+212        ; 0x11e <__bad_interrupt>
  4a:    00 00           nop
  4c:    68 c0           rjmp    .+208        ; 0x11e <__bad_interrupt>
  4e:    00 00           nop
  50:    66 c0           rjmp    .+204        ; 0x11e <__bad_interrupt>
  52:    00 00           nop
  54:    64 c0           rjmp    .+200        ; 0x11e <__bad_interrupt>
  56:    00 00           nop
  58:    62 c0           rjmp    .+196        ; 0x11e <__bad_interrupt>
  5a:    00 00           nop
  5c:    60 c0           rjmp    .+192        ; 0x11e <__bad_interrupt>
  5e:    00 00           nop
  60:    5e c0           rjmp    .+188        ; 0x11e <__bad_interrupt>
  62:    00 00           nop
  64:    5c c0           rjmp    .+184        ; 0x11e <__bad_interrupt>
  66:    00 00           nop
  68:    5a c0           rjmp    .+180        ; 0x11e <__bad_interrupt>
  6a:    00 00           nop
  6c:    58 c0           rjmp    .+176        ; 0x11e <__bad_interrupt>
  6e:    00 00           nop
  70:    56 c0           rjmp    .+172        ; 0x11e <__bad_interrupt>
  72:    00 00           nop
  74:    54 c0           rjmp    .+168        ; 0x11e <__bad_interrupt>
  76:    00 00           nop
  78:    52 c0           rjmp    .+164        ; 0x11e <__bad_interrupt>
  7a:    00 00           nop
  7c:    50 c0           rjmp    .+160        ; 0x11e <__bad_interrupt>
  7e:    00 00           nop
  80:    4e c0           rjmp    .+156        ; 0x11e <__bad_interrupt>
  82:    00 00           nop
  84:    4c c0           rjmp    .+152        ; 0x11e <__bad_interrupt>
  86:    00 00           nop
  88:    4a c0           rjmp    .+148        ; 0x11e <__bad_interrupt>
```


stef[@|.]genesix.org

```
  8a:    00 00           nop
  8c:    48 c0           rjmp    .+144        ; 0x11e <__bad_interrupt>
  8e:    00 00           nop
  90:    46 c0           rjmp    .+140        ; 0x11e <__bad_interrupt>
  92:    00 00           nop
  94:    44 c0           rjmp    .+136        ; 0x11e <__bad_interrupt>
  96:    00 00           nop
  98:    42 c0           rjmp    .+132        ; 0x11e <__bad_interrupt>
  9a:    00 00           nop
  9c:    40 c0           rjmp    .+128        ; 0x11e <__bad_interrupt>
  9e:    00 00           nop
  a0:    3e c0           rjmp    .+124        ; 0x11e <__bad_interrupt>
  a2:    00 00           nop
  a4:    3c c0           rjmp    .+120        ; 0x11e <__bad_interrupt>
  a6:    00 00           nop
  a8:    3a c0           rjmp    .+116        ; 0x11e <__bad_interrupt>
  aa:    00 00           nop
  ac:    38 c0           rjmp    .+112        ; 0x11e <__bad_interrupt>
  ae:    00 00           nop
  b0:    36 c0           rjmp    .+108        ; 0x11e <__bad_interrupt>
  b2:    00 00           nop
  b4:    34 c0           rjmp    .+104        ; 0x11e <__bad_interrupt>
  b6:    00 00           nop
  b8:    32 c0           rjmp    .+100        ; 0x11e <__bad_interrupt>
  ba:    00 00           nop
  bc:    30 c0           rjmp    .+96         ; 0x11e <__bad_interrupt>
  be:    00 00           nop
  c0:    2e c0           rjmp    .+92         ; 0x11e <__bad_interrupt>
  c2:    00 00           nop
  c4:    2c c0           rjmp    .+88         ; 0x11e <__bad_interrupt>
  c6:    00 00           nop
  c8:    2a c0           rjmp    .+84         ; 0x11e <__bad_interrupt>
  ca:    00 00           nop
  cc:    28 c0           rjmp    .+80         ; 0x11e <__bad_interrupt>
  ce:    00 00           nop
  d0:    26 c0           rjmp    .+76         ; 0x11e <__bad_interrupt>
  d2:    00 00           nop
  d4:    24 c0           rjmp    .+72         ; 0x11e <__bad_interrupt>
  d6:    00 00           nop
  d8:    22 c0           rjmp    .+68         ; 0x11e <__bad_interrupt>
  da:    00 00           nop
  dc:    20 c0           rjmp    .+64         ; 0x11e <__bad_interrupt>
  de:    00 00           nop
  e0:    1e c0           rjmp    .+60         ; 0x11e <__bad_interrupt>
    ...

000000e4 <__ctors_end>:
  e4:    11 24           eor     r1, r1
  e6:    1f be           out     0x3f, r1    ; 63
  e8:    cf ef           ldi     r28, 0xFF    ; 255
  ea:    d1 e2           ldi     r29, 0x21    ; 33
  ec:    de bf           out     0x3e, r29    ; 62
  ee:    cd bf           out     0x3d, r28    ; 61

000000f0 <__do_copy_data>:
  f0:    12 e0           ldi     r17, 0x02    ; 2
  f2:    a0 e0           ldi     r26, 0x00    ; 0
  f4:    b2 e0           ldi     r27, 0x02    ; 2
  f6:    ea e5           ldi     r30, 0x5A    ; 90
  f8:    f1 e0           ldi     r31, 0x01    ; 1
  fa:    00 e0           ldi     r16, 0x00    ; 0
  fc:    0b bf           out     0x3b, r16    ; 59
  fe:    02 c0           rjmp     .+4          ; 0x104 <__do_copy_data+0x14>
 100:    07 90           elpm     r0, Z+
 102:    0d 92           st      X+, r0
 104:    a0 30           cpi      r26, 0x00    ; 0
 106:    b1 07           cpc      r27, r17
 108:    d9 f7           brne     .-10         ; 0x100 <__do_copy_data+0x10>

0000010a <__do_clear_bss>:
 10a:    12 e0           ldi     r17, 0x02    ; 2
 10c:    a0 e0           ldi     r26, 0x00    ; 0
 10e:    b2 e0           ldi     r27, 0x02    ; 2
 110:    01 c0           rjmp     .+2          ; 0x114 <.do_clear_bss_start>

00000112 <.do_clear_bss_loop>:
 112:    1d 92           st      X+, r1

00000114 <.do_clear_bss_start>:
 114:    a0 30           cpi     r26, 0x00    ; 0
```


stef[@|.]genesix.org

```
 116:    b1 07           cpc    r27, r17
 118:    e1 f7           brne    .-8          ; 0x112 <.do_clear_bss_loop>
 11a:    02 d0           rcall    .+4          ; 0x120 <main>
 11c:    1c c0           rjmp    .+56         ; 0x156 <_exit>

0000011e <__bad_interrupt>:
 11e:    70 cf           rjmp    .-288        ; 0x0 <__vectors>

00000120 <main>:
 120:    00 d0           rcall    .+0          ; 0x122 <_ada_blinky>

00000122 <_ada_blinky>:
 122:    8f ef           ldi    r24, 0xFF    ; 255
 124:    84 b9           out    0x04, r24    ; 4
 126:    20 e4           ldi    r18, 0x40    ; 64
 128:    3f e1           ldi    r19, 0x1F    ; 31
 12a:    2f 9a           sbi    0x05, 7    ; 5
 12c:    81 e0           ldi    r24, 0x01    ; 1
 12e:    90 e0           ldi    r25, 0x00    ; 0
 130:    f9 01           movw    r30, r18
 132:    31 97           sbiw    r30, 0x01    ; 1
 134:    f1 f7           brne    .-4          ; 0x132 <_ada_blinky+0x10>
 136:    84 36           cpi    r24, 0x64    ; 100
 138:    91 05           cpc    r25, r1
 13a:    11 f0           breq    .+4          ; 0x140 <_ada_blinky+0x1e>
 13c:    01 96           adiw    r24, 0x01    ; 1
 13e:    f8 cf           rjmp    .-16         ; 0x130 <_ada_blinky+0xe>
 140:    2f 98           cbi    0x05, 7    ; 5
 142:    81 e0           ldi    r24, 0x01    ; 1
 144:    90 e0           ldi    r25, 0x00    ; 0
 146:    d9 01           movw    r26, r18
 148:    11 97           sbiw    r26, 0x01    ; 1
 14a:    f1 f7           brne    .-4          ; 0x148 <_ada_blinky+0x26>
 14c:    84 36           cpi    r24, 0x64    ; 100
 14e:    91 05           cpc    r25, r1
 150:    61 f3           breq    .-40         ; 0x12a <_ada_blinky+0x8>
 152:    01 96           adiw    r24, 0x01    ; 1
 154:    f8 cf           rjmp    .-16         ; 0x146 <_ada_blinky+0x24>

00000156 <_exit>:
 156:    f8 94           cli

00000158 <__stop_program>:
 158:    ff cf           rjmp    .-2          ; 0x158 <__stop_program>
```


stef[@|.]genesix.org

Releases

Doubling the number of programmers on a late project does not make than double the delay.
Second Brook's Law

1 AIDE-AVR8

Ed.	*Release*	*Comments*	
1.0.0	2017xxxx		jl

2 AVR-Ada-JLF

Ed.	*Release*	*Comments*	
1.3.0	20150429	GCC 4.9.2 Build - Full Ada 2012 compliant - Debian 8 compliant. Gather all GCC patches until 4.9, all GCC runtimes until 4.9.2 and all binutil patches, until 2.24. More controllers supported, new debug and test directories. More applications, More tests. Scripts updated.	jl

3 AVR-Ada

It have been slightly difficult to sort AVR-Ada releases, since there is no centralized version file in sources. Information mainly comes from : https://sourceforge.net/p/avr-ada/wiki/News & https://sourceforge.net/p/avr-ada/wiki/V1.2%20Release%20Notes & https://sourceforge.net/p/avr-ada/news. Corrections are welcome.

Ed.	*Release*	*Comments*	
1.2.2	20130518	The Run Time System now has optimized assembler code to generate string images of integer variables. The code is based on new code of the upcoming avr libc 1.8.1. You can directly access the function as System.Int_Img.U32_Img. The compile environment (Makefiles, gpr-files) received a long due overhaul, contributed by Jedi. The part specifications for the primary devices (attiny2313 atmega8 atmega168 atmega169 atmega32 atmega328p atmega644p atmega2560) were regenerated from the latest available AVR Studio 4 release XML files. AVR.Serial is a drop in replacement for the existing AVR.UART. It actually is a renaming of a generic instantiation of the new AVR.Generic_Text_IO. AVR.Generic_Text_IO factors out the reusable parts of the input and output routines. You still have to provide routines for sending and receiving single bytes as generic parameters. AVR.Strings.Text is an adoption of Dmitry Kazakov's Strings_Edit](http://www.dmitry-kazakov.de/ada/strings_edit.htm) packages for the small AVR processors. A new example shows the use of the relatively cheap humidity sensors DHT. A command interpreter makes use of the new support packages AVR.Serial and AVR.Strings.Text.	re
1.2.1	20121125	The build script uses gcc-4.7.2. The problem with a missing environment variable during build time is fixed by a patch. The build script also supports building on Darwin. The AVR.Timer packages also support attiny85 and attiny4313. The config file for gprbuild avr.cgpr gets created dynamically during install as it contains hard coded paths.	re

stef[@|.]genesix.org

Ed.	*Release*	*Comments*	
1.2.0	20120801	Here is a brief description of the changes compared to V1.1. Special thanks to Tero Koskinen and Warren W. Gay for contributing most of the new code. Compiler and Tools - The compiler is based on FSF gcc-4.7.2. The Ada compiler already contains many new Ada2012 features. The AVR backend received lots of bug fixes and enhancements, see http://gcc.gnu.org/gcc-4.7/changes.html#avr The binutils are based on binutils-2.20.1. Run Time System. The RTS is configured and extended to support local exceptions, ie. as long as the exceptions are handled in a local block, you can use exceptions now. Added (the types of) Interfaces.C to the RTS. New System.GCC_Builtins to import all AVR specific builtin functions. The compile environment (Makefiles, gpr-files) always use the inline preprocessor with a symbol of the MCU. No explicit preprocessing and explicit intermediate files needed anymore. Removed hardcoded recursive make calls and replaced them with MAKE variable, so building with make named as gmake should work. Supported Controllers Added primary device support for Attiny13, Attiny2313, Attiny4313, Attiny85 and Atmega2560 AVR Support Library The old avr.gpr is now separated into two files avr_tools.gpr and avr_app.gpr. New applications can now be built as an extension project of avr_app.gpr. We have some initial support for boards, ie. you don't specify the MCU but the board like e.g. Arduino2009 and the MCU and clock frequency are automatically set. AVR.ADC. Package for using the built-in A/D converters AVR.SPI. Packages for using SPI communication for master and slave modes. AVR.Threads. Import of Dean Ferreyra's avr-threads library and provide an Ada binding. AVR.TimerX. Synchronized the specs for Timer0, Timer1, and Timer2. Completed basic functionality for the primary MCUs. AVR.UART. Lots of named constants provide more readable configuration at the initialization of a serial line. The standard package supports both polled and interrupt driven receive mode depending on the Init routine that you call. In an actual application you either manually remove the part that you don't need or you declare Reveive_Mode as a constant at the top of avr-uart.adb. The compiler is smart enough to automatically remove the parts that are not needed anymore. External Support Libraries - The following libraries were contributed to AVR-Ada: CRC. Code for calculating 8 and 32 bit checksums. Moved existing code from the 1-Wire lib and added Warren's CRC32 code. SLIP. This package provides a generalized SLIP interface for a communication link lacking "packet boundaries". While it was designed primarily for asynchronous serial connections, it has broader application. All you need is one Transmit and Receive byte procedure for this SLIP interface to use. FATFS. This package provides a FAT16/32 file system capability, targeted for SD/MMC memory cards. Midi. Accessing Midi devices as receiver or transmitter. MCP4922. This package is for sending SPI data to the MCP4922 DAC chip.	re
1.1.0	20100226	Windows binary release of the V-1.1.0 compiler and library.	re
1.1.0	20100213	Initial release	re
1.0.1	20081121	Syntax sugar as one can now directly access (i.e. read and write) registers and IO ports as variables and you still get optimal code. Every register and IO port is declared as a 8 (or 16) bit wide modular integer and an overlapping array of 8 (or 16) booleans. That permits code like : OC1A_DDR (OC1A_Pin) := DD_Output; -- set pin5 as output OC1A_Port (OC1A_Pin) := False; -- clear pin5 Which compiles down to only two assembler instructions : .LSM4: sbi 49-32,5 ; ,, .LSM5: cbi 50-32,5 ; ,, Support libraries for Maxim-Dallas' One-Wire sensors, Sensirion temperature/humidity sensors and a standard LCD. The device description Ada specs are now based on Atmel's XML part description files of AVR Studio 4.15 beta (build 619). The patches for building the cross compiler require gcc 4.3.2.	re

stef[@|.]genesix.org

Ed.	*Release*	*Comments*	
0.5.0	20070923	Extensive package hierarchy in and below AVR.Real_Time with two alternative implementations for the Time type. The first alternative uses 6 bytes, one for year, month, day, hour, minute and second. No subsecond values available. It requires an external 32kHz quartz as in the AVR Butterfly. The second alternative uses 8 bytes and and can count milliseconds. It has not been intensively tested. Experimental implementation of standard Ada delay. Binaries based on gcc-4.2.2, avr-libc-1.4.6 and binutils-2.17.50 The part description Ada specs are now based on Atmel's XML part description files of AVR Studio 4.13 build 524. First package (bounded priority queues) in AVR.Containers8 Improved some routines in AVR.Int_Img (smaller code size). Preliminary implementations in AVR.Int_Val for converting string images to values.	re
0.4.1	20060601	The linker (avr-ld) now generates correct code for avr5 (16k-128K flash memory) targets when called with the --relax switch (default). Building static libraries with avr-gnatmake no longer requirers a host 'gcc' in the search path. Limited support for new devices: atmega2560 atmega2561 (beta) mk_ada_app.sh is now working directory independent, and part of the binary packages (i386-Linux, Windows). The Linux/Windows binaries are based on gcc-4.1.1, avr-libc-1.4.4 and binutils-2.17.50 and where configured for Ada, C and C++. avr-mem.sh now recognizes all supported devices.	re
0.4.0	20060511	The part description Ada specs are now based on Atmel's XML part description files of AVR Studio 4.12 build 473 SP 2 + Atmega644p. Limited support for new devices: at90can32 at90can64 at90pwm2 at90pwm3 at90usb1287 atmega1280 atmega1281 atmega164p atmega165 atmega165p atmega169p atmega324p atmega325 atmega3250 atmega329 atmega3290 atmega406 atmega640 atmega644 atmega644p atmega645 atmega6450 atmega649 atmega6490 attiny24 attiny25 attiny261 attiny44 attiny45 attiny461 attiny84 attiny85 attiny861 We build part specific runtime systems (RTS) now. Old Makefiles have to be adjusted! The AVR-lib has new packages: Watchdog, Sleep, Int_Img. (Note that not all packages have been ported to all devices). Updated scripts to build AVR-Ada with gcc-3.4.6 and gcc-4.1.0 are now located in tools/build/. New script (wizard) to generate a ready-to-compile project directory with all necessary files (tools/mk_ada_app/). New examples (largedemo, debounce) are located in apps/. New bug fixes and workarounds for gcc-3.4 and gcc-4.1 are located in patches/.	re
0.3.0	20050803	The part description spec files are completely new. The Ada specs are now generated directly from Atmel's XML part description files of AVR Studio 4.11 build 406 SP 2. In previous versions they were derived from the C header files of avr-libc. We changed the syntax for accessing single bits from bit numbering to masks in AVR.IO. Simple names in the part description specs now contain bit masks, the pin number is still available with the name xxx_Bit. E.g.: -- Port D Data Register bit 3 PORTD3_Bit : constant Bit_Number := 3; PORTD3 : constant Byte := 16#08#; The previous Set_IO routines in AVR.IO were renamed to simply Put omitting the _IO, the Get_IO have become Get. That is more in the style of other Ada IO packages like Text_IO. New packages for string handling and accessing constants in program space. They are actually a stripped down version of Ada.Strings.Bounded. We build part specific libraries now. That means that we can provide specific functionality adapted to a given part. It is currently used for the UART and EEPROM access only, though. The example programs for accessing a 1-wire devices and an LCD are not part of the distributions anymore. They are only available via CVS. There is also starting support for the atmega169 based AVR Butterfly in the CVS repository (UART, LCD, and dataflash).	re
0.2.0	20041028	We modified the compiler patches to fit cleanly to gcc-3.4.3. They probably also fit in previous gcc-3.4.x releases; I never tried. The "freestanding" patch is now considerably shorter since part of what it does (avoid some code in the binder file) is now provided directly in gcc. The corresponding binder options was previously called "standalone", but that expression is used elsewhere in GNAT. You can now use "Library Projects" with AVR-Ada. I.e. you can have gpr files with "Library_Name", etc. The only permitted value for "Library_Kind" is "static". This feature is used for the Avr library.	re
0.1.0	20031205	We are proud to announce the first release of AVR-Ada, one of the first GCC based Ada compilers targeting 8-bit microcontrolers.	re
	20031204	We released the first package of AVR-Ada. This is for testing the actual release V0.1.	re

stef[@|.]genesix.org

Ed.	Release	Comments	
	20031202	I have just released a new patch file gcc-3.3.2-AVR-Ada-2. It now incudes the essential parts of the standalone patch.	re

4 Gnat-GPL-2012-AVR Windows

Ed.	Release	Comments	
2012			
2011			
2010			

Ada, « it's stronger than you ».
Tribute to Daniel Feneuille, legendary french Ada teacher (and much more)[25]

[25] http://d.feneuille.free.fr