Integrating 8-bit AVR Micro-Controllers in Ada

Pablo Vieira Rego

Embraer, S.A., Av. Brigadeiro Faria Lima 2170, Putim, São José dos Campos, SP, Brazil;email: pvrego@gmail.com

Abstract

The increasing popularity of 8-bit AVR micro-controllers for using with robotics and low-complexity embedded applications has called the attention of much of the software community. The purpose of this paper is to present an approach for developing embedded applications for these micro-controllers in Ada, using GNAT AVR as Microsoft Windows cross-compiler and GNAT Programming Studio as IDE.

Keywords: 8-bit AVR micro-controller, GNAT AVR, Ada, ZFP.

1 Introduction

The Atmel AVRs[1] are presented as a family of 8- and 32-bit micro-controllers designed to provide flexibility, based a priori in C and assembly programming. More specifically, Arduino is a family of boards with a central 8-bit AVR micro-controller and peripheral I/Os which implement a bunch of low-level hardware support such as analog and digital data write, read and storage, USART protocols, Ethernet, wifi modules and others, for a reasonable low cost. They became very popular in the development of memory mapped robotics with open-source hardware for hobbyists and low-to-medium complexity embedded products and introductory robotics programming courses [1, 2, 3, 4, 5].

For open-source development in C/C++, there are four main branches: Atmel Studio, WinAVR, Arduino IDE and Eclipse. Atmel Studio[2] provides the IDE with C and assembly compiler and debugger and several in-built libraries with pre-implemented features and specific configurations for the whole 8- and 32-bit families. WinAVR[3] is a suite of executable, open source software development tools for the Atmel AVR series of RISC microprocessors hosted on the Windows platform, which includes the GNU GCC compiler for C and C++. Arduino IDE[4] is a platform written in Java and based on Processing and avr-gcc and provides C++ development based on application examples. On Eclipse, the AVR Eclipse Plugin[5] provides tools for developing C applications on AVR micro-controllers. In all cases, Atmel's Application Notes serve as development guidelines, as well as the micro-controllers datasheets, which contain the complete hardware description and some code examples[6]. For RTOS'es support, there are plenty of choices for AVR development, we could cite AvrX[7], FreeRTOS[8] and RTEMS[9], most of them written in C/C++.

For the development in Ada, there are some academic references for development on AVRs, most focused on the 32-bit micro-controllers. Gregertsen and Skavhaug [6] describe a deterministic multitasking run-time environment supporting the Ravenscar tasking model of Ada 2005 implemented on the Atmel AVR32 UC3A micro-controller. In [7], they describe an object-oriented real-time framework for Ada 2005 and Ravenscar profile implemented on AVR32 UC3 micro-controller; in [8], they propose the addition of execution-time control features for interrupt handling as an addition to the Ada standard library and describes it using an implementation of GNAT bare-board Ravenscar run-time environment on the Atmel AVR32 architecture; and finally in [9, 10], Gregertsen and Skavgaug describe an implementation of timing event and execution time control features with support for interruption on the same architecture.

For the 8-bit micro-controllers, however, there are fewer academic references for the development in Ada. Ras and Cheng [11] describe a deterministic run-time environment for Ada-05 on the 8-bit ATmega16 micro-controller; and Andersen [12] presented an approach for developing Arduinos with the AVR-Ada cross-compiler.

There are some open-source cross-compiler options for programming AVR micro-controllers in Ada, we can cite AVR-Ada[10] and GNAT AVR GPL [13]. In this paper we will show an approach to generate a .hex target burning file for the micro-controller using the open-source GNAT AVR cross compiler and the GPS IDE. The GNAT AVR provides a Microsoft Windows cross-compiler and builder for AVR 8-bit micro-controllers [14] and implements *Zero Footprint Profile (ZFP)*, which does not require any run-time routines, and is intended for high-critically applications related to DEF Stan 00-55 certification [15]. Thus it does not have support for tasking or exception propagation. So, the purpose of this paper is to serve as an user guide for development of monotask applications in 8-bit AVR micro-controllers in Ada, following the ZFP.

[1] `http://www.atmel.com/products/microcontrollers/avr/default.aspx`
[2] `http://www.atmel.com/microsite/atmel_studio6/`
[3] `http://winavr.sourceforge.net/index.html`
[4] `http://arduino.cc/en/Main/Software`
[5] `http://avr-eclipse.sourceforge.net/wiki/index.php/The_AVR_Eclipse_Plugin`
[6] `http://www.atmel.com/products/microcontrollers/avr/default.aspx`
[7] `http://www.barello.net/avrx/index.htm`
[8] `http://www.freertos.org/`
[9] `http://www.rtems.com/`
[10] `http://sourceforge.net/apps/mediawiki/avr-ada/index.php?title=Main_Page`

2 Materials and Methods

2.1 Arduino Duemilanove (ATmega328P)

The Arduino Duemilanove is a development board based on Atmel ATmega328P/ATmega168 8-bit micro-controllers. This paper concerns to ATmega328P, but can be extended to other AVR 8-bit chips with minor changes. Duemilanove provides 14 digital I/O pins, 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header and a reset button[11]. Table 1 resumes some characteristics of this board.

Table 1: Characteristics of Arduino Duemilanove with ATmega328p

micro-controller	ATmega328P
Operating Voltage	5 V
Input Voltage (recommended)	7-12 V
Input Voltage (limits)	6-20 V
Digital I/O Pins	14
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB[12]
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

2.2 Development Environment and Target Generation

The required applications are the GNAT AVR GPL edition[13], which comes with the IDE and the Windows cross-compiler; and WinAVR, which provides CRT libraries for the micro-controllers.

For generating the output .hex file on Windows, the simple project consists of the following elements:

- Specification package which will map the AVR registers;
- Main file which will include the chip specification package;
- GPS .gpr configuration file;
- An object library provided by WinAVR;
- A batch script which will provide .elf/.hex conversion and a pre-configured burning tool.

[11]General description of Arduino Duemilanove can be found at `http://arduino.cc/en/Main/arduinoBoardDuemilanove`

[12]In which 2 KB is used by bootloader.

[13]`http://libre.adacore.com`

2.2.1 *The Specification Package*

Each AVR device has a number of I/Os associated with its internal registers. These descriptions are present in micro-controller datasheet and the specification package shall map the memory address entries defined in the datasheet.

For the ATmega328P micro-controller, datasheet [16] section *Register Summary* on page 423 describes all the memory addresses available to develop the applications. Using the Memory Mapped I/O approach as described by McCormick et al. [17], each word shall be mapped, and sometimes it is usefull to map also the meaning of each one of the bits. The words so should be pointed exactly to the memory address it describes.

For example, the registers *Port B Data Register (PortB)*, *Port B Data Direction Register (DDRB)* and *Port B Input Pins Address (PINB)* could be mapped in a specification package as

Listing 1: Simplified Specification Package for ATmega328P

```
-- atmega328p.ads
with Interfaces; use Interfaces;
with System;

package ATmega328P is

   -- PORTB: Port B Data Register
   PORTB : Unsigned_8;
   for PORTB'Address use System'To_Address (16#25#);

   -- DDRB: Port B Data Direction Register
   DDRB : Unsigned_8;
   for DDRB'Address use System'To_Address (16#24#);

   -- PINB: Port B Input Pins
   PINB : Unsigned_8;
   for PINB'Address use System'To_Address (16#23#);

end ATmega328P;
```

2.2.2 *GPS Project File*

The GPS project file has to include the AVR configurations for the packages *Ide*, *Compiler*, *Builder* and *Linker*. Below are presented some suggestions on how to configure these project file packages in order to build the .elf output which will be used to generate the .hex output.

Listing 2: GPS Project File for ATmega328P

```
-- arduino.gpr
project Arduino is

   for Source_Dirs use (".", "src");
   for Object_Dir use "obj";
   for Exec_Dir use "bin";
   for Main use ("main.adb");

   package Ide is
      for Gnat use "avr-gnat";
      for Gnatlist use "avr-gnatls";
      for Debugger_Command use "avr-gdb";
   end Ide;

   package Compiler is
```

```
      for Default_Switches ("ada") use ("−mmcu=avr5");
   end Compiler;

   package Builder is
      for Executable_Suffix use ".elf";
      for Default_Switches ("ada") use ("−−RTS=rts−zfp");
   end Builder;

   package Linker is
      for Default_Switches ("ada") use ("obj\crtm328p._o", "
            −nostdlib", "−lgcc", "−mavr5", "−Tdata=0
            x00800200", "−−mmcu=avr5");
   end Linker;

end Arduino;
```

The *Ide* package configures the tools related to the AVR-GNAT installation. The *Builder* package sets the suffix for the builder output file and specifies the Zero Footprint Profile as the Real-Time System.

In the *Compiler* package, the option *-mmcu=avr5* enables compilation using AVR5 architecture, which is the case of the ATmega328P.

In the *Linker* package, the *crtm328p._o* is the CRT[14] got from WinAVR installation[15], since GNAT AVR GPL only provides CRT for ATmega2560. The option *-nostdlib* avoids linking with standard libraries, not provided by GNAT AVR GPL, and the option *-lgcc* tells the linker to use the compiler support library. The options *-mavr5* and *-mmcu=avr5* set the device architecture for AVR5. And the option *-Tdata* sets the start address of the data section [18].

For the ATmega2560 micro-controller, we should exchange from AVR5 to AVR6 architecture and the CRT. GNAT provides this CRT in a form of *crt1-atmega2560.S* file[16], which shall be compiled in order of obtaining the *crt2560._o*. One can compile it using

```
avr−gcc −c −mmcu=avr6 −o crt2560._o crtl−atmega2560.S
```

2.2.3 The Main File

The Main file contains the main task available by ZFP and the libraries related to the code. The following dummy code should be modified in order to code the micro-controller.

Listing 3: Dummy main file

```
with ATmega328P; use ATmega328P;

procedure Main is
begin
   null;
end Main;
```

[14]C Run-Time Library.

[15]In this case, the crtm328p.o has its extension renamed to _o to avoid been deleted when the user makes a project clean.

[16]The file *crt1-atmega2560.S* can be found on GNAT AVR GPL 2012 installation in the folder *C:\GNAT\2012\share\examples\gnat-cross\avr\atmega2560.*

2.2.4 The Hex & Burn Batch Script

An option for a batch script to generate the .hex and burn it on a micro-controller in COM7 follows.

Listing 4: Batch script for converting the .elf and burn the resulting .hex

```
@rem hexitandburn_duemilanove.bat

@echo Converts .elf to .hex
avr−objcopy −O ihex main.elf main.hex

@echo Burn the chip
c:\WinAVR\bin\avrdude.exe −p m328p −c avrisp −P com7
      −b 57600 −e −U flash:w:main.hex
```

In the case of using the ATmega2560 micro-controller, the *avrdude* line would change to

```
c:\WinAVR\bin\avrdude.exe −p m2560 −c stk500v2 −P
      com7 −b 115200 −e −U flash:w:bin\main.hex
```

And it is suggested to create a .xml file in Windows user profile folder (eg. c:\Documents and Settings\user\.gps\plug-ins) to enable a menu to execute the actions of convert and burn the .hex in the micro-controller.

Listing 5: Script embedded.xml to enable a menu in GPS to convert and burn Arduino Duemilanove

```
<?xml version="1.0" ?>
<embedded>
      <action name="Hexit␣and␣Burn␣Duemilanove">
            <external>cmd /c
                  hexitandburn_duemilanove.bat</
                  external>
      </action>

      <submenu after="Build">
            <title>Embedded</title>
            <menu action="Hexit␣and␣Burn␣
                  Duemilanove">
                     <title>[Hexit and Burn]
                          Duemilanove</title>
            </menu>
      </submenu>
</embedded>
```

3 Limitations of the Zero Footprint Profile

The Zero Footprint Profile defines an Ada run-time certifiable run-time with a memory footprint reduced to null. It excludes in particular the use of dynamic Ada semantic. The ZFP still allows the use of the major Ada features such as generics, child units, library-level tagged types, interfaces and local exception handling [19]. The ZFP is suitable for software certification using DEF Stan 00-55 standard [15].

The ZFP limitations are [20]:

- Constructs defined in the ARM Section 9 (Tasking and Synchronization)
- Exception propagation
- Packed arrays with component size other than 1, 2, 4, 8 or 16 bits

- Some exponentiation operation
- 64-bit integer and fixed-point types
- Boolean operation on packed arrays
- Some comparison operations on arrays of discrete components
- The attributes *Image*, *Value*, *Body_Version*, *Version*, *Width*, and *Mantissa*
- Controlled types
- Some applications of the attributes *Address*, *Access*, *Unchecked_Access*, *Unrestricted_Access*
- Non library-level tagged types
- Annex E (Distributed Systems)

4 An Example Implementation

In this section we will show a very simple example application. For this implementation, it is enough to use the *arduino.gpr* project file described in section 2.2.2 and the *atmega328p.ads* described in section 2.2.1 as specification package and a main file. So, suppose we desire to move a 5-wires Stepper Motor some steps forward, blink a led once, move the Stepper Motor some steps backward and blink the same led twice. The Stepper Motor used is a Mototech S35S5 extracted from an old optical scanner.

Figure 1 shows the connections among the components. The Arduino Duemilanove is connected to the Stepper Motor through the current driver ULN2803AP using pins B0, B1, B2 and B3, and the led is connected using pin B5.

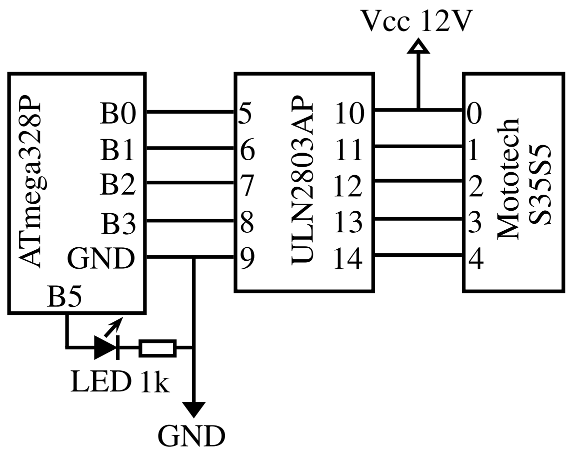

Figure 1: Connections between Arduino Duemilanove, Led, ULN2803AP and Stepper Motor.

```
-- main.adb
with ATmega328P; use ATmega328P;

procedure Main is

  -- Stepper Motor coils definitions
  COIL0 : constant := 2 ** 0;
  COIL1 : constant := 2 ** 2;
  COIL2 : constant := 2 ** 1;
  COIL3 : constant := 2 ** 3;

  -- Default delay and number of cycles
  DEFAULT_DELAY : constant := 100;
  DEFAULT_CYCLES : constant := 50;

  -- Null instruction delay
  procedure Custom_Delay (Wait_Cycle : Integer) is
  begin
    for i in 1 .. Wait_Cycle loop
      for j in 1 .. Wait_Cycle loop
        null;
      end loop;
    end loop;
  end Custom_Delay;

  procedure Blink_Times (Times : Integer) is
  begin
    for Counter in 1 .. Times loop
      PortB := 2#00100000#; Custom_Delay (5 *
           DEFAULT_DELAY);
      PortB := 2#00000000#; Custom_Delay (5 *
           DEFAULT_DELAY);
    end loop;
  end Blink_Times;

begin

  -- Initialize all PortB pins as outputs
  DDRB := 2#11111111#;

  -- Forward over the coils
  for Counter in 1 .. DEFAULT_CYCLES loop
    PortB := COIL0; Custom_Delay (DEFAULT_DELAY);
    PortB := COIL1; Custom_Delay (DEFAULT_DELAY);
    PortB := COIL2; Custom_Delay (DEFAULT_DELAY);
    PortB := COIL3; Custom_Delay (DEFAULT_DELAY);
  end loop;

  -- Blink led once
  Blink_Times (1);

  -- Backward over the coils
  for Counter in 1 .. DEFAULT_CYCLES loop
    PortB := COIL3; Custom_Delay (DEFAULT_DELAY);
    PortB := COIL2; Custom_Delay (DEFAULT_DELAY);
    PortB := COIL1; Custom_Delay (DEFAULT_DELAY);
    PortB := COIL0; Custom_Delay (DEFAULT_DELAY);
  end loop;

  -- Blink led twice
  Blink_Times (2);

  -- Disable all output pins
  PortB := 0;

end Main;
```

5 Conclusion

This paper shows an approach for developing Ada applications for 8-bit AVR micro-controllers, using GNAT AVR as the Microsoft Windows cross-compiler, and GNAT Programming Studio as IDE. GNAT AVR follows the Zero Footprint Profile for Real-Time Library, thus not all Ada Real-Time features are present in this development such as multi-tasks.

We believe that Ada can improve the quality of the applications for AVRs. The AVR community is invited to try this approach. Also the Ada community is encouraged to propagate its knowledge about real-time applications to the AVRs field in a mean that it certainly will gain more popularity over more one embedded software branch.

References

[1] J. D. Brock, R. F. Bruce, and S. L. Reiser, "Using Arduino for introductory programming courses," *J. Comput. Sci. Coll.*, vol. 25, no. 2, pp. 129–130, Dec. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1629036.1629057

[2] P. Jamieson, "Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?" International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS'11), 2011, accessed: 23/06/2012. [Online]. Available: http://www.users.muohio.edu/jamiespa/html_papers/fecs_11.pdf

[3] R. Balogh. Robotics course with the Acrob robot. Österreichische Gesellschaft für innovative Computerwissenschaften. Accessed: 23/06/2012. [Online]. Available: http://www.innoc.at/fileadmin/user_upload/_temp_/RiE/Proceedings/15.pdf

[4] ——, "Acrob - an Educational Robotic Platform," *AT&P Journal Plus*, vol. 10, no. 2, pp. 6–9, 2010, ISSN 1336-5010.

[5] I. B. Gartseev, L.-F. Lee, and V. N. Krovi, "A Low-Cost Real-Time Mobile Robot Platform (ArEduBot) to support Project-Based Learning in Robotics & Mechatronics," R. Stelzer and K. Jafarmadar, Eds. INNOC - Austrian Society for Innovative Computer Sciences, 2011, pp. 117–124, accessed: 23/06/2012. [Online]. Available: http://www.innoc.at/fileadmin/user_upload/_temp_/RiE/Proceedings/25.pdf

[6] K. N. Gregertsen and A. Skavhaug, "An efficient and deterministic multi-tasking run-time environment for Ada and the Ravenscar profile on the Atmel AVR32 UC3 microcontroller," in *Design, Automation Test in Europe Conference Exhibition, 2009.*, April 2009, pp. 1572 –1575.

[7] ——, "A Real-Time Framework for Ada 2005 and the Ravenscar Profile," in *Software Engineering and Advanced Applications, 2009. SEAA '09. 35th Euromicro Conference on*, aug. 2009, pp. 515 –522.

[8] ——, "Execution-time control for interrupt handling," *Ada Lett.*, vol. 30, no. 1, pp. 33–44, May 2010. [Online]. Available: http://doi.acm.org/10.1145/1806546.1806550

[9] ——, "Implementing the new Ada 2005 timing event and execution time control features on the AVR32 architecture," *Journal of Systems Architecture*, vol. 56, no. 10, pp. 509 – 522, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1383762110000937

[10] K. N. Gregertsen, "Execution time control : A hardware accelerated ada implementation with novel support for interrupt handling," Ph.D. dissertation, Norwegian University of Science and Technology, Department of Engineering Cybernetics, 2012.

[11] J. Ras and A. M. Cheng, "A deterministic run-time environment for Ada-05 on the ATmega16 microcontroller," *Ada Lett.*, vol. 30, no. 3, pp. 13–22, Oct. 2010. [Online]. Available: http://doi.acm.org/10.1145/1879097.1879072

[12] J. S. Andersen, "Programming Arduinos in Ada." Free and Open Source Software Developers' European Meeting (FOSDEM'12), February 2012.

[13] AdaCore. (2009, Mar) GNATPro available for 8-bit AVR Microcontroller. NY, USA. Accessed: 23/06/2012. [Online]. Available: http://www.adacore.com/press/8-bit-avr-microcontroller/

[14] ——. Ada Development Environment for the Atmel AVR 8-bit microcontroller. NY, USA. [Online]. Available: http://www.adacore.com/gnatpro-safety-critical/platforms/avr/

[15] Ministry of Defence, *Requirements for Safety Related Software in Defense Equipment*, Directorate of Standardization - Defense Standard, Rev. INTERIM Def Stan 00-55/Issue 1, Apr 1991.

[16] Atmel Corporation. 8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash: ATmega48PA/ ATmega88PA/ ATmega168PA/ ATmega328P. San Jose, CA, USA. Accessed: 21/06/2012. [Online]. Available: http://www.atmel.com/Images/doc8161.pdf

[17] J. W. McCormick, F. Singhoff, and J. Hugues, *Building Parallel, Embedded, and Real-Time Applications with Ada*, 1st ed. New York, USA: Cambridge University Press, 2011.

[18] AVR Topics: J.2 Compiler and Linker Flags for AVR. Accessed: 24/06/2012. [Online]. Available: http://docs.adacore.com/gnat-cross-docs/html/gnat_ugx_12.html

[19] AdaCore. Runtime Profiles. [Online]. Available: http://www.adacore.com/gnatpro/toolsuite/runtimes/

[20] ——. Ada Restrictions in the Zero Footprint Profile. [Online]. Available: http://docs.adacore.com/gnat-hie-docs/html/gnathie_ug_4.html#SEC27