Introduction to AVR Microcontrollers

Revision: June 20, 2008

DIGILENT®
www.digilentinc.com

215 E Main Suite D | Pullman, WA 99163
(509) 334 6306 Voice and Fax

Overview

This document describes how to begin embedded development using microcontrollers. Microcontrollers are available from many different manufacturers and are designed with differing architectures, but this document focuses on the Atmel AVR microcontroller architecture.

This document is accompanied by the *Cerebot II Getting Started* project source code, written in the C language. Refer to the project source code while reading this document to better understand techniques used to control the AVR microcontroller located on the Cerebot II board.

This document is based on the Digilent Educational Solutions material, available free at www.digilentinc.com.

Introduction to Microcontrollers

Many electronic devices are controlled by an embedded microcontroller (MCU). Microcontrollers consist of four main subsystems:

- memory
- input/output
- control unit
- arithmetic logic unit (ALU)

The central processing unit (CPU) is composed of a control unit and input/output subsystem interfaces to serial and/or parallel ports. The memory subsystem consists of RAM for data storage and ROM, EEPROM, and Flash for program storage. Other key features include peripheral devices such as timers/counters, pulse-width modulation (PWM) channels, comparators, watchdog timers, and universal synchronous asynchronous receiver/transmitters (USARTs).

Microcontrollers are special-purpose computers contained on a single chip. These chips can be embedded in many different devices such as digital thermostats, appliances, personal electronic devices, and the Cerebot II board.

Atmel's AVR microcontrollers use a RISC (reduced instruction set computing) core that generally processes one instruction per clock cycle. The core reduces the need for a large complex instruction set. AVR Flash microcontrollers have the ability to operate at clock frequencies up to 20MHz and can therefore execute a maximum of 20 million instructions per second (MIPS).

AVR microcontrollers are designed around the Harvard computer architecture and support 32 general-purpose registers. The microcontrollers also support 16- and 32-bit arithmetic calculations. The instruction set contains 130 instructions that can be used to create high-density programs.

Atmel's AVR RISC family of microcontrollers supports a variety of on-chip peripherals such as:

- EEPROM
- analog to digital converters (ADCs)
- multiple timers
- various serial communication methods (USART, TWI, SPI)
- multiple pulse-width modulation channels (PWMs)
- digital I/O

The Atmel AVR 8-bit RISC architecture allows for low power consumption and can be operated between 1.8V and 5.5V. If you are using a Digilent AVR-based board, refer to the appropriate reference manual for usable input voltage values, as many have on-board voltage regulators. The fact that the AVR is an efficient device that can be operated at low voltages allows it to be embedded in battery powered applications such as robotics. The AVR architecture also provides various sleep modes, which powers-down unused peripherals and may even turn off the CPU to conserve power. The AVR uses interrupts to wake from sleep modes. The AVR architecture also provides for embedded development by supporting in-system programming, on-chip debuggers, and in-circuit emulators.

Embedded Development using the Cerebot II

The Cerebot II has an embedded Atmel ATmega64 AVR microcontroller on the board. To get the board to blink the on-board LEDs based on the position of an input jumper, the controlling architecture of the AVR must be understood.

The AVR's characteristics are controlled using various registers. These registers all have unique names and each consists of one byte (eight bits.) In this byte, each bit may control a different property of the peripheral it is related to.

For general input/output (I/O) operation of pins on the AVR, the following registers control their properties:

- Data direction register (DDRx), where x is the port letter designation. When a bit in the data direction register is written to a '1', the corresponding pin is set as an output. When a bit in the data direction register is written to a '0', the corresponding pin is set as an input.
- Port register (PORTx), where x is the port letter designation. This register is paired with the DDRx register to control the behavior of corresponding pins. If the pin is set as an output, then the PORTx register controls drive on the pin. When a '0' is written to a bit in PORTx, the corresponding pin will be tied to ground. When a '1' is written to a bit in PORTx, the corresponding pin will be tied to VDD, which in the case of the CerebotII is 3.3V. Alternatively, if a bit in the DDRx register is set as an input, then the corresponding bit in the PORTx register controls the state of the internal pull-up resistor connected to the corresponding pin. When the bit in the PORTx register is written to a '0', the internal pull-up is disabled, allowing the pin's input to float. When the bit in the PORTx register is written to a '1', the pull-up resistor is activated, tying the input to VDD. This is often a more desirable situation. If an input pin is not electrically tied to either VDD or ground, the state of the pin cannot be predicted, leading to erratic behavior of the input.
- Pin register (PINx), where x is the port letter designation. This register is also paired with the DDRx register. When a bit in the DDRx register is set as an input, the corresponding pin may be read using the PINx register. This register is read, rather than written to, and contains the

state of the I/O pins. To read the state of a certain input pin, such as the pin connected to the user-input JP5 jumper, a mask should be used.

Installing the Necessary Tools

A few tools are needed to get started with the Cerebot II. These tools include:

- a Cerebot II board with 5V power supply
- a Windows computer with available USB or serial port
- Atmel's AVR Studio4 with WinAVR GCC plug-in software
- Digilent's Adept software suite and AVRP (AVR programmer)
- A Digilent JTAG-USB cable (this cable programs the board using SPI)

To begin, install the software listed above. AVR Studio 4 is available free from www.atmel.com. Digilent's software is available free from www.digilentinc.com.

After installing the software, make sure that no Digilent products are attached to your computer and follow these steps:

1. Power the Cerebot II using the 5V power supply.
2. Attach the JTAG-USB cable to the Cerebot II board by connecting the cable to the J1 header on the board. Make sure that the VDD pin on the cable matches the VCC pin on the board's header and that the TMS pin on the cable matches the RST pin on the board's header. Note: This will leave all of the pins on the neighboring JP7 header unconnected.
3. Plug the other side of the cable into an available USB port on your computer.
4. When the Windows Found New Hardware Wizard appears, select "No, not this time" and click Next.
5. Select "Install from a list or specific location (Advanced)" and click Next.
6. Select "Don't search. I will choose the driver to install" and click Next.
7. Make sure that "Show compatible hardware" is checked. Under the Model list box, "Digilent USB JTAG cable" should have appeared. Select "Digilent USB JTAG cable" and click Next.
8. If a warning displays, click Continue Anyway.
9. The wizard should indicate that the cable's driver has been successfully installed. Click Finish. If you encounter any problems installing the driver, re-install the Digilent software suite with the current version available on Digilent's website and try again.

Programming the Board

Before modifying the source code, program the Cerebot II with the .hex file that is in the project directory. To do this, make sure that you have already installed the USB driver as indicated above and that the Cerebot II is still powered and connected to the computer via the programming cable.

Then follow these directions:

1. Open the USB Administrator program that is part of the Adept software suite. If your cable is properly configured, the USB Administrator program should display an item in the list box with a serial number listed next to it. Take note of the ID and serial number listed.
2. Start the AVRP programmer. Click the Programmer Settings tab and click Configure. A Communication Modules box will open. Click Add Module. Another box will open; click the

USB tab. In the Connected Modules list box, select the module with the matching serial number you found earlier and click Add and Done.

3. In the Communication Modules screen, select the module in the list box with the matching serial number, click Save and then Close. Finally, in the Digilent AVR Programmer window, select Digilent USB JTAG/SPI Cable in the Programmer drop-down box, then select the new device in the Port drop-down box.
4. Click on the Program tab. In the AVR Device drop-down box, select Cerebot. Now navigate to the .hex program file by clicking the "…" button. The .hex file will be located in the default folder located in the project directory. Open the .hex file and click Program. Messages should be displayed in the bottom box of the AVR Programmer window indicating whether the Cerebot II was successfully programmed. If the device was programmed correctly, the LEDs on the Cerebot II should be toggling on and off. If you move the jumper connected to JP5 between the '0' and '1' position, the LEDs should either be sweeping or demonstrating a binary counter.

Studying the Source Code

To develop software for the AVR microcontroller, open the AVR Studio 4 integrated development environment (IDE). To open the included project, go to the Project menu and select Open Project. Find the folder that contains the source code, select Cerebot II_getting_started and click Open.

The program should start opening the different source files on the screen. The project consists of header files that include various useful definitions and the C source code files that contain the functions used in the program.

Header files make writing the source code much easier. Header files not only clean up the source code by making it appear less cluttered, they can also contain useful and meaningful definitions that make writing code more intuitive. Instead of referencing the schematic to find out which port a certain Cerebot II pin is connected to, writing an intuitive set of definitions can make it much easier to write software. A full set of meaningful definitions has been included in the project and is located in the cerebotII.h header file. Please read the header file to become familiar with the naming conventions used in the project.

The use of comment blocks in the source code can greatly increase code readability and neatness. This will make it easier to maintain the code when writing revisions. Included in the main.c file are many comment blocks, many of which are provided for later code expansion but are not yet being used. Read through the source code, paying attention to the comment blocks.

The first thing that any program should execute is the initialization of internal peripherals and global variables. In the main.c file, there are two functions provided that accomplish these tasks. The first function called is DeviceInit(). This function initializes the registers that are going to be used in the program and sets all unused pins as inputs with pull-ups enabled. By setting each unused pin this way, the AVR has increased protection from the outside environment. If there were global variables present in the source code, they would have been initialized in the AppInit() function.

Notice the use of the OR (|), AND (&), and NOT (~) operators in the source code. These are used to modify specific bits in registers and provide bit-masking during read instructions.

After the AVR executes the DeviceInit() and AppInit() functions, the program enters the main loop. A microcontroller should never return from the main function. In order to prevent this, a while loop is traditionally used to prevent the program from ever completing. The program would then continually

execute the instructions contained in the loop or be interrupt-driven, waiting for specific hardware interrupts to drive the program.

In the main loop, the microcontroller reads the status of the JP5 jumper, then executes instructions that toggle the on-board LEDs in unique ways. Before looping again, the program executes the Wait_ms() function, which just wastes clock cycles for a specified time.

To program the board with a new program, select the Build option from the Build menu bar. The debugger will open, indicating whether the build was successful. This action creates a new .hex file that will be located in the default folder in the project directory. To load the new program onto the board, follow the instructions detailed above.

Using Peripherals

Atmel's 8-bit AVR microcontrollers have many on-chip peripherals than can be used to accomplish different tasks and interact with the outside environment in many different ways. These include analog-to-digital converters (ADCs), Timer/Counters, PWM channels and many more. Make sure to refer to the data sheet for the relevant microcontroller. All of the available peripherals and registers are clearly explained in the data sheet, and in many cases sample source code is provided.

Digilent offers a wide variety of peripheral modules (Pmods) that can be plugged into the Cerebot II. Among these Pmods are speakers, LCD displays, keypads, ADCs, DACs and many more. Using these peripherals allows for a potentially unlimited number of projects.