CIS541: Embedded and Cyber Physical Systems

Spring 2010

AVR/Butterfly Lab Manual

This lab manual helps you with implementing the pacemaker controller and the heart simulator on AVR/Butterfly boards. Upon finishing it, you will have a baseline heart simulator and also be able to program towards the AVR/Butterfly boards. Additional reference materials are also provided.

Requirement

To program with the AVR/Butterfly boards, you need to have access to a Windows machine on which the AVR Studio 4 and WinAVR software are installed. With these, you are able to write programs in C and compile them to hex files ready to transfer to the AVR/Butterfly boards. To actually dump the hex file to the board, the machine must have a RS232 port. (Update: RS232-USB adapter will be provided so having RS232 is not necessary.)

Setting up the Environment

The AVR Studio 4 and WinAVR are freely available online for download.

- AVR Studio 4.18 and service packs are at http://www.atmel.com/dyn/products/avrstudio. Free registration is required to download.
- The homepage of WinAVR is at http://winavr.sourceforge.net/. The download site is at http://sourceforge.net/projects/winavr/files/.

The installation process is straightforward. It is suggested to install WinAVR after AVR Studio 4 so that automatic configuration for programming in C and compile with GCC will be performed. With this, it is also possible to debug the C code inside AVR Studio.

AVR Studio 4 is mainly used for debugging and dumping the hex files to the AVR/Butterfly board. WinAVR provides programming environment for the AVR/Butterfly boards in C. The following sections will get you familiarize with the procedure of working with the AVR/Butterfly boards programming.

Programming with AVR/Butterfly Board

If you want to follow a thorough introduction, the book (Pardue, 2005) is a good starting point. The first two chapters are freely available online. The following tutorial is partly adapted from this book and other online sources with heavy edits, which are listed in the Resources section.

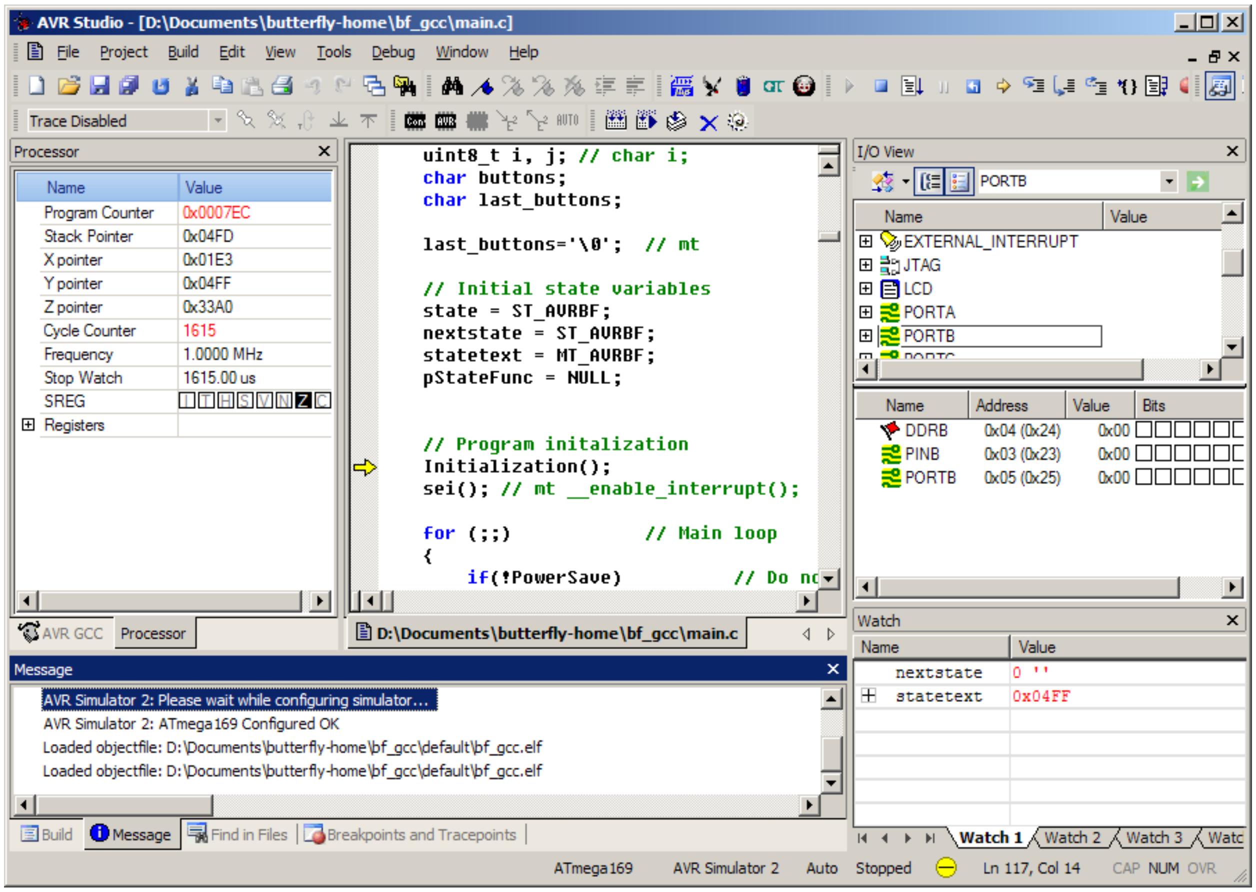
Standing on the Shoulders of Giants

One simple way to start programming with the AVR/Butterfly board is to start with other's project already. For this tutorial, we will start with the Butterfly GCC-Port program. This is the GCC port of the pre-loaded Butterfly program on the board. It demonstrates (more than) all functions that we will be using in our project. A direct link of the downloadable source is at http://www.siwawi.arubi.uni-kl.de/avr_projects/bf_gcc_20091217.zip. If you want more information about the source code, visit this website http://www.siwawi.arubi.uni-kl.de/avr_projects.

Unzip the downloaded source code, you'll find a bf_gcc folder. Double-click the bf_gcc.aps file will open AVR Studio 4. Two simple steps will let you run the program with the simulator provided with AVR Studio 4:

1. Build → Rebuild All
2. Debug → Start Debugging

If you are familiar with programming with, say, Visual Studio, then the IDE is quite similar. The difference is that more specifics with the board are provided, as detailed below.


AVR Studio - [D:\Documents\butterfly-home\bf_gcc\main.c]
File Project Build Edit View Tools Debug Window Help
Trace Disabled
AVR GCC
Processor
Build Message Find in Files Breakpoints and Tracepoints
Watch 1 Watch 2 Watch 3 Watc

Processor

Name	Value
Program Counter	0x0007EC
Stack Pointer	0x04FD
X pointer	0x01E3
Y pointer	0x04FF
Z pointer	0x33A0
Cycle Counter	1615
Frequency	1.0000 MHz
Stop Watch	1615.00 us
SREG	I T H S V N Z C
⊞ Registers	

```
    uint8_t i, j; // char i;
    char buttons;
    char last_buttons;

    last_buttons='\0';  // mt

    // Initial state variables
    state = ST_AVRBF;
    nextstate = ST_AVRBF;
    statetext = MT_AVRBF;
    pStateFunc = NULL;


    // Program initalization
    Initialization();
    sei(); // mt __enable_interrupt();

    for (;;)            // Main loop
    {
        if(!PowerSave)          // Do nc
```

D:\Documents\butterfly-home\bf_gcc\main.c

I/O View — PORTB

Name	Value
⊞ EXTERNAL_INTERRUPT	
⊞ JTAG	
⊞ LCD	
⊞ PORTA	
⊞ PORTB	

Name	Address	Value	Bits
DDRB	0x04 (0x24)	0x00	☐☐☐☐☐☐
PINB	0x03 (0x23)	0x00	☐☐☐☐☐☐
PORTB	0x05 (0x25)	0x00	☐☐☐☐☐☐

Message

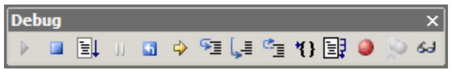
```
AVR Simulator 2: Please wait while configuring simulator...
AVR Simulator 2: ATmega169 Configured OK
Loaded objectfile: D:\Documents\butterfly-home\bf_gcc\default\bf_gcc.elf
Loaded objectfile: D:\Documents\butterfly-home\bf_gcc\default\bf_gcc.elf
```

Watch

Name	Value
nextstate	0 ''
⊞ statetext	0x04FF

ATmega169 AVR Simulator 2 Auto Stopped Ln 117, Col 14 CAP NUM OVR

1. In the status bar at the bottom, the simulator (AVR Simulator 2) and CPU type of the AVR/Butterfly board (ATmega169) are provided.
2. In the Processor panel to the left, information of the particular CPU is shown. The Stop Watch field maybe of particular interest when observation of timing is required. However, one must note that the time shown here is calculated based on the CPU frequency and number of instructions of the compiled code. It may not be real time.
3. The AVR/Butterfly board input and output port values are shown in the I/O view panel to the right.
4. The Debug toolbar is of particular importance when playing with the simulator. The meanings of the buttons are straightforward.



One thing to note when playing with the buttons is that, when it comes to real-time provided by the Real-Time Clock (RTC), the simulator may halts. This is because the program is busy waiting some value to be changed by an external RTC provided on the AVR/Butterfly board.

Structure of Sample Code

The GCC-Port of the Butterfly program is straight forward to read. This subsection provides an overview. You may safely ignore this section if you want to read the code on your own. The main logic of the program is as follows.

1. Initialization of variables (including setting current state and state function pointer pStateFunc)
2. Main loop
    a. If not in PowerSave mode
        i. Print state indication text
        ii. Get an input from the button
        iii. Execution code associated in the current state
        iv. If input from the button is not NULL, transit to next state
        v. Update state function pointer pStateFunc
    b. Code for AutoPowerSave mode
        i. If system idle time is larger than preset, set PowerSave to TRUE
    c. Code for AutoPress mode
        i. If a button is held for some time, consider it pressed even if released, until changed
    d. Code for SleepMode
        i. Turn off LCD
        ii. If UP button is pressed, return to normal mode

Programming with AVR/Butterfly Features

The detailed functions for state behaviors are implemented in separate files. Entry points of these functions are defined in menu.h. Here is a short list of where in the code you can find necessary part for implementing the Pacemaker project code.

- For reading input from the Joystick, use input=getkey() function, and then compare input with predefined key values KEY_ENTER(·), KEY_NEXT(→), KEY_PREV(←), KEY_PLUS(↑), and KEY_MINUS(↓).
- For outputting values to the LCD, use functions defined LCD_functions.h. Chapter 11 of (Pardue, 2005) also provides examples of how to use functions defined here.
- For inputting and outputting values to the AVR/Butterfly board PORTB or PORTD pins, you need two steps:
    1. Set the data direction for each pin of the port. For example, if you want to set pins 0 to 3 (lower) of PORTB as input and pins 4 to 7 (upper) as output, you can set DDRB=0xF0=0b11110000, where 0 for a pin means input, and 1 for a pin means output. The DDRB means Data Direction Register for Port B.
    2. After this, you can read values of PINB for values of Port B, and write values to PORTB for Port B.
- For using clocks, there are two approaches provided.

    1. One is to use the interrupts provided from the RTC. To do this, you need to register the functions you'll execute when a Timer interrupt has occurred. These functions can be found in the sound.c and sound.h file.
    2. Another way is just to rely on the CPU busy waiting. One such example is the Delay function defined in main.c. Note that the simulator works with this type of timer, with an estimate of the execution time (of the loop for busy waiting) according to the CPU speed parameter.
- (Optional) For using the beepers (to indicate periodic heart beats or an error), you need to set the frequency, the tempo, the duration, and then output values to OC1A. The piezo-element is connected to this port and can read and play sounds from Pulse Width Modulator (PWM). The sound.c demonstrated examples of using this feature as well as Timer0 to play tunes. The last project in Chapter 8 of (Pardue, 2005) provides a detailed analysis of how the code works.

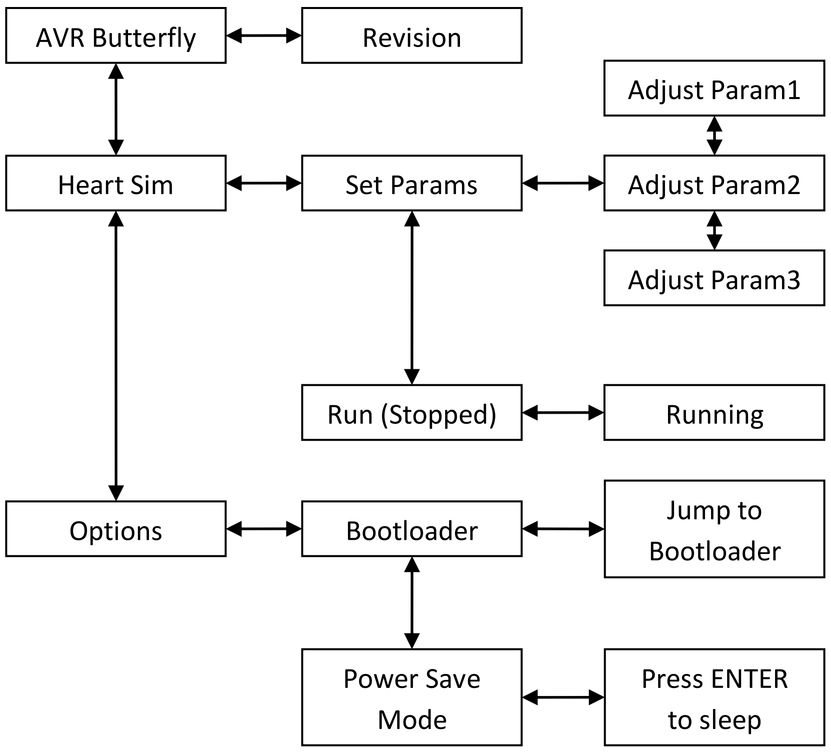
Here are some tips that might be useful.

1. When reading the code, there might be many predefined variables as OC1A, PORTB, TCCR1A, etc. It is handy to have the datasheet of the ATmega169 chip (AMTEL, 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash, 2006) for easy lookup. It is named doc2514.pdf, freely available online, and also included in the companion pack of this document.
2. It is not the case that all pins of Port B and Port D are free to use. This is because some of them are connected to other pins which you also want to use. For one example, some pins are connected to LCD pins (if you write values the screen may show weird things). For another, the OC1A pin mentioned above is actually the same as Pin 5 or Port B (PB5). See (Boehnlein, 2003) for a list of pins that you can safely use.

Creating Your Own Code

To save battery life, you are required to retain the power save module from the source code. Then, you may create your heart simulator.

Create a menu item. You can safely remove unnecessary menu items and add your own entry to the heart simulator. In addition, you may want to set parameters from the AVR/Butterfly board itself rather than changing the parameters on PC, compile, and dumping the program to the board again. A possible menu system may look like the following (and may be different). Note that only the Heart Sim module is to be implemented, other parts are already provided. Also, the "Adjust" functions are also things you can reuse (check the EnterName function in vcard.c).



Required features for the Heart Simulator are introduced in the previous section. The expected behavior is detailed in a separate document (Milestone 3 requirement).

Simulation & Debugging

As has been mentioned above, when you use a RTC, simulation & debugging is not feasible with AVR Studio 4. To see performance or correctness of your code, we need profiling and data logging with an oscillator. Lab hours will be provided for using the oscillator. Logged data can be used for analyzing correctness of your design and implementation.

Aside from timing issues, AVR Studio 4 and the GDB debugger are good enough to use. For the latter, the WinAVR website http://winavr.sourceforge.net/helpme.html contains much information for the adventurous souls.

Dumping to the Board

Step by step instructions can be found in several of the documents in the companion package. See (AMTEL, AVR Butterfly Evaluation Kit User Guide, 2005), (Boehnlein, 2003), and (Mckain, 2004), etc. The only difference in our environment is that an additional RS232 to USB adapter will be used.

Resources

Aside from the documents mentioned in the text (also listed in the References section), here are some more spots you may want to hit.

- The AVR Projects website http://www.siwawi.arubi.uni-kl.de/avr_projects/. It contains much information including the GCC-Port code you've downloaded.

- The AVR libc website http://www.nongnu.org/avr-libc/ and reference manual http://www.nongnu.org/avr-libc/user-manual/index.html.
- The WinAVR website http://winavr.sourceforge.net/ and its document page http://winavr.sourceforge.net/helpme.html.
- The AVR Freaks website http://www.avrfreaks.net/.

References

AMTEL. (2006, July). *8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash.* Retrieved from http://www.atmel.com/dyn/resources/prod_documents/doc2514.pdf

AMTEL. (2005, April). *AVR Butterfly Evaluation Kit User Guide.* Retrieved from http://www.atmel.com/dyn/resources/prod_documents/doc4271.pdf

AMTEL. (2003, May). *AVR Butterfly Quick Start User Guide.* Retrieved from http://www.atmel.com/dyn/resources/prod_documents/doc4249.pdf

Boehnlein, A. (2003, October 11). *Programming the Butterfly for Idiots Like Me.* Retrieved from AVR Projects: http://www.siwawi.arubi.uni-kl.de/avr_projects/pbfi.pdf

Mckain, W. (2004, May 14). *How to Program the AVR Butterfly.* Retrieved from http://www.coe.uncc.edu/~jmconrad/AVR/ButterflyGuide.pdf

Pardue, J. (2005). *C Programming for Microcontrollers Featuring ATMEL's AVR Butterfly and the free WinAVR Compiler.* Smiley Micros.