[펌] Lecture 5 - AVR GCC Compiling

분류없음
2008/09/15 22:47
수정 : 수정(창으로) | (비공개)→ 공개로 변경합니다 | 관련글(트랙백) | 삭제
http://www.sparkfun.com/commerce/tutorial_info.php?tutorials_id=105
Lecture 5 - AVR GCC Compiling

Sorry for the confusion. When these tutorials were written and photographed, we used the ATmega8. We now carry the newer ATmega168. You will find all ATmega168 information in the following pages, but the pictures will show an ATmega8.

I know very little about the ins and outs of the AVR-GCC compiler. I've learned a few basics that helped me along the way, but when you run up against a jam, google and AVRfreaks.net are your friend.

First, we did the blinky. Open this code in PN2 and make sure you can compile it. Click on Tools->Make All. The window in the bottom screen should say 'Process Exit Code: 0' meaning the compilation was successful. If not, there should be a line number listing of the problem line of code. Be sure to check above and below the indicated line for problems.

In the second example C file called basic-out-atmega168.c (basic-out.c for the ATmega8), I've inserted a handful of functions and lines of code. First of the black magic:

```
#define FOSC 16000000
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
```

What is all this noise at the top of the file? This is a series of defines that calculates the MYUBRR variable with the correct number. Since serial communication depends on the fact that we will be transmitting and receiving at 9600 bits per second, it's crucial to tell the ATmega168 what bit rate to set. Because the ATmega168 is dictated by the oscillator that it is using, we must correctly calculate what value we need to load into the ATmega168 hardware so that the ATmega168 sends the serial pulses at the correct rate with a given oscillator type. In our case, we are using a 16MHz oscillator so we can setup the define statement as shown. MYUBRR is calculated at compile time and is loaded successfully into the hardware UART during run time.

A reader's untested submission:

> The UBRR value calculation in Lecture 5 could be more accurate with the following macro:
>
> ```
> #define MYUBRR (((((FOSC * 10) / (16L * BAUD)) + 5) / 10) - 1)
> ```
>
> There is also pretty useful web form for UBRR calculation:
>
> http://www.wormfood.net/avrbaudcalc.php

```
static FILE mystdout = FDEV_SETUP_STREAM(uart_putchar, NULL, _FDEV_SETUP_WRITE);
```

This line creates a buffer for the printf statement to post to. I'd rather not explain it, simply because I don't understand it. When I am working on a new coding project I never start from a blank page, I *always* start from a known working program and slowly bring in bits of other projects to get the code I need, writing bits along the way. Please start from this printf example and build away. The purpose here is to get your printing string to the terminal window.

Checkout the ioinit() function. You'll notice some new commands.

```
UCSR0B = (1<<
```

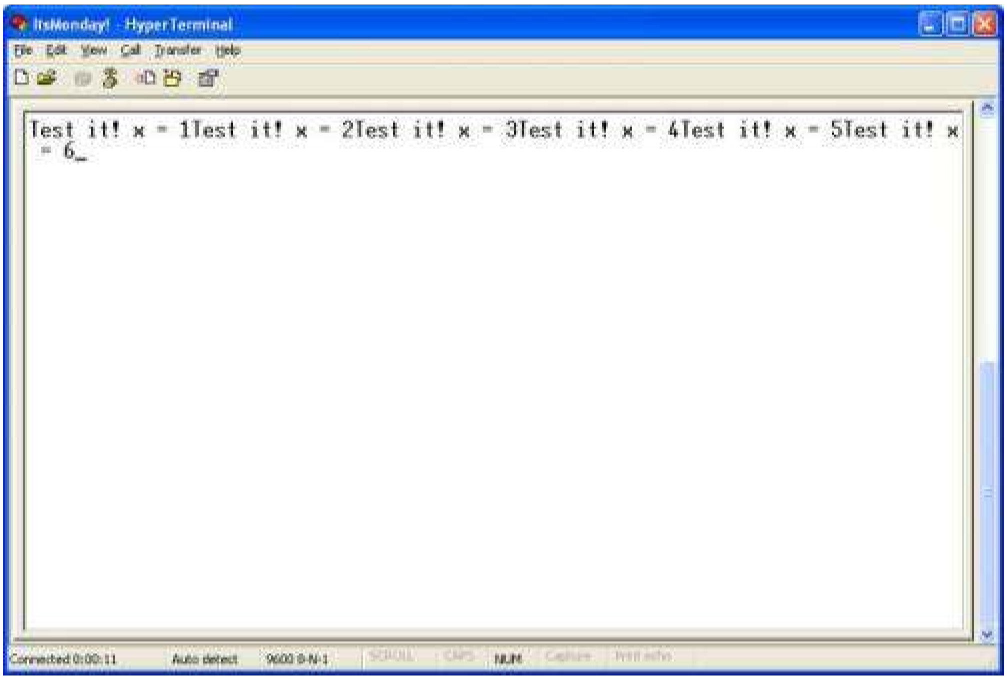
This is the really funky, but very practical, method of setting bits on the AVR series. RXEN0 is defined in some file as '5'. 1<

Finally, we see the very comfortable line of code:

```
printf("Test it! x = %d", x);
```

What did you say? This is not a comfortable line of C code for you? Ok - printf is somewhat of a universal function to pass serial strings and variables to the outside world. The line of code above will pass the string "Test it! x =" to the serial port and it should display on the terminal window. After that, the %d gets converted to an actual decimal number so that whatever digital number is currently stored in the variable x gets printed to the terminal screen. So what? This simple printf statement allows you to print variable contents and see what's going on within your C program.

Time to load the basic-out-atmega168.c file onto your breadboard. Open up PN2, compile basic_out-atmega168.c. Power up your board, click on Tools->[WinAVR] Program from within Programmer's Notepad. The code should now be loaded onto your ATmega168. If WinAVR throws a verification error, try again. Open up the terminal window at 9600bps if you don't already have it open.



```
Test it! x = 1Test it! x = 2Test it! x = 3Test it! x = 4Test it! x = 5Test it! x
 = 6_
```

Text output from the ATmega168 and MAX232 circuit

All right! We've got output from the ATmega168! Now let's talk about some more of the code:

```
sbi(PORTC, STATUS_LED);
```

Another funky one if you're not used to the AVR series. To toggle a GPIO pin (general purpose input/output pin), you need to read the state of the port, mask the bit change into the state-word, and then write the 8-bits back onto the port effectively modifying just the one bit. Instead of doing all that by every time you want to toggle a port pin, there's this handy macro:

```
#define sbi(var, mask) ((var) |= (uint8_t)(1 << mask))
```

SBI sets a bit. CBI clears a bit. You have to specify which port you're working with and which pin you want to alter.

Throw another define at the top of your code:

```
#define STATUS_LED 0
```

Now you can control your STATUS_LED on PORT C using these two simple commands:

```
sbi(PORTC, STATUS_LED);
```

To turn on the LED and

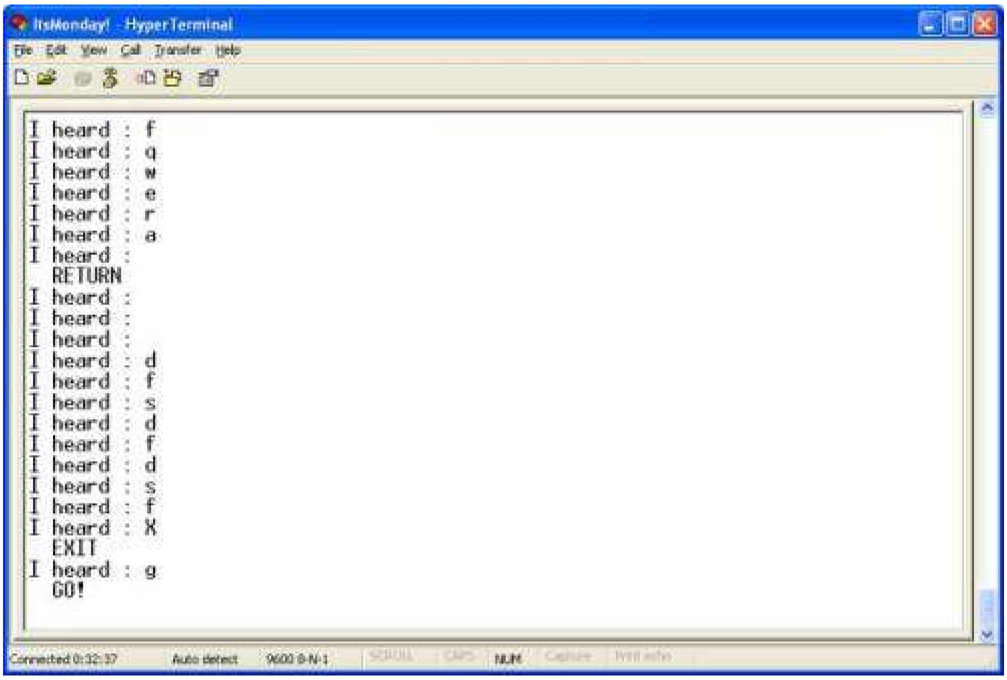
```
cbi(PORTC, STATUS_LED);
```

To turn it off.

You should have an LED tied to pin 23 on the ATmega168. When in doubt, toggle your status LED to figure out where the code is hanging or use a printf statement.

There are also some tweaks to the delay_ms() routine. Because we increased the oscillator from 1MHz to 16MHz, I increased the loop iterations to tie up the processor for longer. I didn't do any real calculations so don't depend on my delay_ms routine. delay_ms(1000) looks to be roughly a 1 second delay.

Open basic-in-atmega168.c (basic-in.c for the ATmega8) and load up your breadboard:



ItsMonday! - HyperTerminal
File Edit View Call Transfer Help
Connected 0:32:37 Auto detect 9600 8-N-1 SCROLL CAPS NUM Capture Print echo

```
I heard : f
I heard : q
I heard : w
I heard : e
I heard : r
I heard : a
I heard :
 RETURN
I heard :
I heard :
I heard :
I heard : d
I heard : f
I heard : s
I heard : d
I heard : f
I heard : d
I heard : s
I heard : f
I heard : X
 EXIT
I heard : g
 GO!
```

Key presses and various responses

Here we see that whatever character we hit, the ATmega168 responds with 'I heard : ' and the character. Also, if you hit return, X, or g, you will see various special output.

```
key_press = uart_getchar();

printf("I heard : %c\n", key_press);

if(key_press == 'g') printf(" GO!\n");
if(key_press == 'X') printf(" EXIT\n");
if(key_press == 13) printf(" RETURN\n");
```

uart_getchar sits waiting for a character to appear in the UART. Once received, the ATmega168 outputs the character (%c) and goes to a new line (\n). It then checks to see if the key press was one of three special cases. If so, it prints an extra string accordingly. I hope you are starting to see the power of the input/act-upon/output that a microcontroller is capable of. With a little bit of work, you could program your own text-based adventure game. Go to town.

Remember back when you were struggling to get your power supply wired up? Nice job! Time to heat up your irons.

We love feedback! Please report typos, comments, or recommendations to

미아를 찾습니다

이름: 김재수 (남, 당 만5세)
일자: 1978년 9월 13일
장소: 서울특별시 영등포구, 대림동
특징: 엄지손톱 빠진 적 있음, 쌍거풀 있음, 얼굴이 둥근편이었음

이름: 이선희 (여, 당 만9세)
일자: 1983년 7월 29일
장소: 강원도 양양군, 남애해수욕장
특징: 오른쪽 손목에 V 자의 흉이 있음

Supported by Needlworks, 어린이재단

Posted by 털보네i

트랙백 0 : 댓글 0

◀ PREV : [1] : [2] : [3] : [4] : [5] : ... [132] : NEXT ▶

by **털보네i**

공지사항

카테고리

분류 전체보기 (132)
- 마이컴_원칩 (39)
- 여행정보 (6)
- 재태크_경제 (0)
- 윈도우관련 (20)
- 서버 (5)
- 게임 (2)
- 낙서장 (5)
- 영화 (2)
- PDA등 (9)
- 대체에너지 (3)
- 전기전자 (32)
- MAC OSX86 (2)

최근에 올라온 글

- [펌] Lecture 5 - AVR GCC..
- DIY GPS Hacking Projects
- Eagel CAD에서 표제란 / Fr..
- Eagle CAD tutorial
- SpokePOV
- mp3 다운로드 프로그램 -..
- 서프2 데이터 스펙 Sirf 2..
- [펌] PIC18LF2550 2.4GHz S..
- [펌] AVR Tutor - C langua..

- [펌] MAKE: AVR oscilloscope

최근에 달린 댓글

최근에 받은 트랙백

Total : 17,685
Today : 63 Yesterday : 39

tistory
티스토리 가입하기!

태그목록

글 보관함

- 2008/09 (19)
- 2008/08 (26)
- 2008/07 (25)
- 2008/06 (19)
- 2008/04 (3)

달력

« 2008/09 »

일	월	화	수	목	금	토
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

링크

홈 | 위치로그 | 태그 | 방명록 | 관리자 | 글쓰기
털보네i's Blog is powered by Daum & Tattertools / Designed by Tistory