Nerd Ralph

science and technology stuff

Friday, December 27, 2013

Writing AVR assembler code with the Arduino IDE

Although I have written a lot of code in high-level languages like C++, I enjoy writing assember the most. For inserting assembler code into Arduino sketches, you can read a gcc inline assembly guide. If you have some assembly code and want to use it, there is an easier way than converting it to inline assembly; you can make it a library.

The Arduino Serial class consumes a lot of resources, and even the tiny cores serial class (TinyDebugSerial) adds overhead to the half duplex software UART code it seems to be based on. I decided to integrate my implementation of AVR305 with an Arduino sketch.

I started by making a directory called BasicSerial in the libraries directory. Inside I created a BasicSerial.S file for my assember code. In order for assembler code to be callable from C++, it is necessary to follow the avr-gcc register layout and calling convention, and mark the function name global. The TxByte function takes a single char as an argument, which gcc will put in r24. The Arduino core uses interrupts which would interfere with the software UART timing, so interrupts are disabled at the start of TxByte and re-enabled at the end. Here's the code:

```
#include <avr/io.h>
; correct for avr/io.h 0x20 port offset for io instructions
#define UART_Port (PORTB-0x20)
#define UART_Tx 0

#define bitcnt r18
#define delayArg 19

#if F_CPU == 8000000L
  #warning Using 8Mhz CPU timing
  #define TXDELAY 21
#elif F_CPU == 16000000L
  #warning Using 16Mhz CPU timing
  #define TXDELAY 44
#else
  #error unrecognized F_CPU value
#endif

.global TxByte
; transmit byte in r24 - 15 instructions
; calling code must set Tx line to idle state (high) or 1st byte may be lost
; i.e. PORTB |= (1<<UART_Tx)
TxByte:
 cli
        sbi UART_Port-1, UART_Tx              ; set Tx line to output
        ldi bitcnt, 10                             ; 1 start + 8 bit + 1 stop
        com r24                                    ; invert and set carry
TxLoop:
        ; 10 cycle loop + delay
        brcc tx1
        cbi UART_Port, UART_Tx                ; transmit a 0
tx1:
        brcs TxDone
        sbi UART_Port, UART_Tx                ; transmit a 1
TxDone:
        ldi delayArg, TXDELAY
TxDelay:
; delay (3 cycle * delayArg) -1
        dec delayArg
        brne TxDelay
        lsr r24
        dec bitcnt
        brne TxLoop
 reti    ; return and enable interrupts
```

The last thing to do is to create a header file called BasicSerial.h:

```
extern "C" {
void TxByte(char);
}
```

If the extern "C" is left out, C++ name mangling will cause a mismatch. To use the code in the sketch, just include BasicSerial.h, and call TxByte as if it were a C function. Here's a sample sketch:

```
#include <BasicSerial.h>

// sketch to test Serial

// change LEDPIN based on your schematic
#define LEDPIN  PINB1

void setup(){
  DDRB |= (1<<LEDPIN);    // set LED pin to output mode
}

void serOut(const char* str)
{
   while (*str) TxByte (*str++);
}
```

Blog Archive

- ► 2017 (2)
- ► 2016 (17)
- ► 2015 (38)
- ► 2014 (37)
- ▼ 2013 (9)
  - ▼ December (2)
    - Writing AVR assembler code with the Arduino IDE
    - Trimming the fat from avr-gcc code
  - ► November (1)
  - ► September (2)
  - ► August (4)

About Me

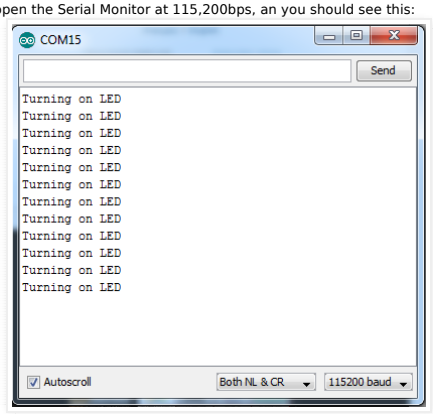
Ralph Doncaster

I have four children, two step-children, and a grand-daughter. Wife 1.0 was a failure, but as with many nerd projects, version 2.0 is a big improvement. :-) As an adult I was diagnosed with ADHD and Asperger's. gmail: ralphdoncaster

View my complete profile

```
void loop(){
  serOut("Turning on LED\n");
  PORTB |= (1<<LEDPIN);  // turn on LED
  delay(500);            // 0.5 second delay
  PORTB &= ~(1<<LEDPIN); // turn off LED
  delay(1000);           // 1 second delay
}
```

Download and run the sketch, open the Serial Monitor at 115,200bps, an you should see this:



I've posted BasicSerial.zip containing BasicSerial.S and BasicSerial.h. Have fun!

New year's update:
I've modified the code so the delay timing is calculated by a macro in BasicSerial.h. Just modify the line:
`#define BAUD_RATE 115200`
BasicSerialv2.zip

Posted by Ralph Doncaster at 10:49 AM +4 Recommend this on Google

19 comments:

Porotito January 20, 2014 at 5:54 PM

Hi Ralph!
"calling code must set Tx line to idle state (high) or 1st byte may be lost"

When must be set tx, before or after the call? I´m trying to interrupt drive your rx code without success so far.

great work! thanks!

Reply

Ralph Doncaster January 22, 2014 at 3:08 PM

Set Tx high (or enable pullup) before the call to TxByte. You probably won't have to do it though - the USB-TTL adapter I've tested with doesn't require Tx to be driven high - it only requires it to be pulled low.

Reply

Porotito January 23, 2014 at 10:04 AM

Hello!
Got it working interrupt driven by pin change int, a fifo buffer and a single pin on tiny85. Just adjusted a bit the receive delays due interrupt latency. Here is my code in case of someone interest:
init_uart:
sbi UART_Port, UART_Rx ;rx pullup high
sbi UART_Ddr, UART_Rx ;tx/rx = out
in r24, GIMSK ;enable pin change int
ori r24, (1 << PCIE)
out GIMSK, r24
sbi PCMSK, PCINT3
ret

PCINT0_vect:
push r24
in r24, SREG
push r24
in r24, PINB
sbrc r24, UART_Rx ;act on high to low transitions only
rjmp PCINT0_vect_exit

ldi r24, ASM_RXDELAY_15
ldi r22, ASM_RXDELAY_1

rcall RxTimedByte

sts rx_byte, r24 ;new byte

rcall rx_fifo_store ;store new byte in fifo

PCINT0_vect_exit:
pop r24
out SREG, r24
pop r24
reti

Many thanks Ralph!

Reply

Ralph Doncaster February 19, 2014 at 9:01 PM

Just noticed your comments in the "awaiting moderation" list. I adjusted the moderation settings so that shouldn't happen again. Good work getting it working with interrupts. With the interrupt trigger, RxTimedByte won't need to check for the start bit - maybe you already removed that.
I didn't pull up the datasheet, but I think you can set the interrupt to be active low only, that way you won't need the check for high to low transition (which should only ever happen inside the interrupt, and therefore you should never see it anyway).

Reply

Colin McInnes May 23, 2014 at 3:12 PM

I get an error with avr-gcc saying that "-assembler-with-cpp" is an unrecognized option. Which version of Arduino/avr-gcc are you using?

Reply

Replies

Ralph Doncaster May 24, 2014 at 1:04 PM

I've tested it with 1.0.4, 1.0.5-r2, and the 1.5 beta (nightly build).

Colin McInnes May 26, 2014 at 10:38 AM

Turns out it was a Ubuntu issue, their version of 1.0.5 is not the same as the released version.

Reply

Darkdoom August 6, 2015 at 9:16 AM

This comment has been removed by the author.

Reply

Darkdoom August 6, 2015 at 9:52 AM

I hacked together a little Arduino Library, working on Arduino 1.6.4 with the Streaming library:

http://www.file-upload.net/download-10823054/attiny85ser.rar.html

attiny85ser s = attiny85ser();

s << "[INFO] ledblink_wheel pos: " << pos << " pause: " << pause << endl;

Working like a charm, might be not the most efficient way but .. it is working.

Reply

Replies

Ralph Doncaster August 6, 2015 at 1:13 PM

I generally avoid file upload sites; they usually have annoying pop-ups, and some have been known to wrap files in an installer that contains malware.
Since writing this post, I've updated my soft uart, and pushed it to my github account:
https://github.com/nerdralph/nerdralph/tree/master/avr/libs/bbuart

You might consider publishing your code on github or bitbucket so it is more accessible.

Reply

Rajeev Jha January 26, 2016 at 3:51 AM

Hello Ralph,
Two questions,
(1) in the code above, how do you calculate the TXDELAY? At F_CPU=8MHZ and baud 115200, it would need (8x 10^6 / 115200) clock cycles to shift one symbol, i.e. 69 CLK cycles. How do you calculate TXDELAY=44?

(2) in the new code, version 2 updated here, How does the assembly routine get the TXDELAY calculated via macro in header file?
I see this in comment " transmit byte in r24 with bit delay in r22 - 15 instructions"
It would help people not familiar with assembly and assembly-c interface.

Many Thanks

Reply

Replies

Ralph Doncaster January 28, 2016 at 8:00 PM

The txdelay is 21 for 8Mhz and 44 for 16Mhz. See the #if F_CPU ... lines. The delay loop is 3 cycles, so the delay is 3x tx delay (see code comments)

In V2, the txdelay is passed to the txbyte function along with the character to output.

Reply

Unknown April 11, 2016 at 1:49 PM

Hi, I use ATTiny85 with your soft. I try to change le Tx pin but it's alway pin 5 (on IC) == pin PB0 on tiny85. I change #define UART_Tx 0 with #define UART_Tx 3 , but I have no change in test.
How change it ?
Think

Reply

Replies

Ralph Doncaster April 11, 2016 at 4:05 PM

Changing UART_Tx should do the trick. The way the Arduino IDE does dependencies is a bit quirky so maybe it's not recompiling the .S file when you make the change. I do most of my programming from the command line using makefiles, so I don't know how to force a compile of all files in the IDE. Another option would be to find the temporary .o files and delete them.

Reply

Florin Samoila May 31, 2016 at 6:50 AM

hi, i'm using your method but i get "aaa aaa aaa" on the serial monitor I'm using a linux machine, could this be the problem?

Reply

Florin Samoila May 31, 2016 at 7:31 AM

i made it to work in the end :))
the problem was that i didn't uploaded a blank sketch on the arduino board

Reply

Workshop August 17, 2016 at 6:13 AM

Hi I am new assembly language,i am working with arduino. I wanna fetch the data from co2 sensor. To do so, i need to write the following command which is written in assembly, could anyone help me to write the same code in arduino.
send command 80h 02h 00h 7Eh to co2 sensor system until uplink response with type being 80h

Reply

Dave Boechler September 26, 2016 at 6:02 PM

Great work on this! I am a bit weak on assembly, but attempted to incorporate your script into an avr-gcc test program. It spits out properly to the serial port, but the c routine does not blink the led or delay. Condensed main.c:

```
void serOut(const char* str)
{
while (*str) TxTimedByte (*str++,TXDELAY);
}

main...
DDRB |= (1<<PINB5); // set LED pin to output mode
serOut("Toggle LED\r\n");
PORTB ^= (1<<PINB5); // toggle LED
_delay_ms(100); // 0.1 second delay
```

Also I had to remove extern "C" from the header:
What am I doing wrong?

Thanks

Reply

Replies

Ralph Doncaster September 27, 2016 at 9:05 AM

If you're getting the serial port output, the problem is with your C code.
If there is no loop in your main(), that would explain why you do not see your LED blink. If you do have a loop, I'd suggest posting the full code to avrfreaks.net and maybe someone there will help you.

Reply

Enter your comment...

Comment as: Select profile...

Publish Preview

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)

Simple template. Powered by Blogger.