// hello.rxtx.44.c

//

// step response transmit-receive hello-world

// 115200 baud FTDI interface

//

// Neil Gershenfeld

// 11/6/11

//

// (c) Massachusetts Institute of Technology 2011

// This work may be reproduced, modified, distributed,

// performed, and displayed for any purpose. Copyright is

// retained and must be preserved. The work is provided

// as is; no warranty is provided, and users accept all

// liability.

//

#include <avr/io.h>

#include <util/delay.h>

#define output(directions,pin) (directions |= pin) // set port direction for output

#define set(port,pin) (port |= pin) // set port pin

#define clear(port,pin) (port &= (~pin)) // clear port pin

#define pin\_test(pins,pin) (pins & pin) // test for port pin

#define bit\_test(byte,bit) (byte & (1 << bit)) // test for bit set

#define bit\_delay\_time 8.5 // bit delay for 115200 with overhead

#define bit\_delay() \_delay\_us(bit\_delay\_time) // RS232 bit delay

#define half\_bit\_delay() \_delay\_us(bit\_delay\_time/2) // RS232 half bit delay

#define char\_delay() \_delay\_ms(10) // char delay

#define PWM\_delay() \_delay\_us(25) // PWM delay

#define nloop 100 // loops to accumulate

#define serial\_port PORTA

#define serial\_direction DDRA

#define serial\_pin\_out (1 << PA4)

#define transmit\_port PORTA

#define transmit\_direction DDRA

#define transmit\_pin (1 << PA2)

#define led\_port PORTA

#define led\_direction DDRA

#define red (1 << PA7)

#define green (1 << PA6)

#define led\_port PORTB

#define led\_direction DDRB

#define blue (1 << PB2)

void put\_char(volatile unsigned char \*port, unsigned char pin, char txchar) {

//

// send character in txchar on port pin

// assumes line driver (inverts bits)

//

// start bit

//

clear(\*port,pin);

bit\_delay();

//

// unrolled loop to write data bits

//

if bit\_test(txchar,0)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,1)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,2)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,3)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,4)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,5)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,6)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

if bit\_test(txchar,7)

set(\*port,pin);

else

clear(\*port,pin);

bit\_delay();

//

// stop bit

//

set(\*port,pin);

bit\_delay();

//

// char delay

//

bit\_delay();

}

int main(void) {

//

// main

//

static unsigned char count;

static uint16\_t up,down;

//unsigned char count, pwm;

//

// set clock divider to /1

//

CLKPR = (1 << CLKPCE);

CLKPR = (0 << CLKPS3) | (0 << CLKPS2) | (0 << CLKPS1) | (0 << CLKPS0);

//

// initialize output pins

//

set(serial\_port, serial\_pin\_out);

output(serial\_direction, serial\_pin\_out);

clear(transmit\_port, transmit\_pin);

output(transmit\_direction, transmit\_pin);

//

// initialize LED pins

//

set(led\_port, red);

output(led\_direction, red);

set(led\_port, green);

output(led\_direction, green);

set(led\_port, blue);

output(led\_direction, blue);

// init A/D

//

ADMUX = (0 << REFS2) | (0 << REFS1) | (0 << REFS0) // Vcc ref

| (0 << ADLAR) // right adjust

| (0 << MUX3) | (0 << MUX2) | (1 << MUX1) | (1 << MUX0); // PB3

ADCSRA = (1 << ADEN) // enable

| (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // prescaler /128

//

// main loop

//

while (1) {

//

// accumulate

//

up = 0;

down = 0;

for (count = 0; count < nloop; ++count) {

//

// settle, charge

//

settle\_delay();

set(transmit\_port, transmit\_pin);

//

// initiate conversion

//

ADCSRA |= (1 << ADSC);

//

// wait for completion

//

while (ADCSRA & (1 << ADSC))

;

//

// save result

//

up += ADC;

//

// settle, discharge

//

settle\_delay();

clear(transmit\_port, transmit\_pin);

//

// initiate conversion

//

ADCSRA |= (1 << ADSC);

//

// wait for completion

//

while (ADCSRA & (1 << ADSC))

;

//

// save result

//

down += ADC;

}

//

// send framing

//

put\_char(&serial\_port, serial\_pin\_out, 1);

char\_delay();

put\_char(&serial\_port, serial\_pin\_out, 2);

char\_delay();

put\_char(&serial\_port, serial\_pin\_out, 3);

char\_delay();

put\_char(&serial\_port, serial\_pin\_out, 4);

//

// send result

//

put\_char(&serial\_port, serial\_pin\_out, (up & 255));

char\_delay();

put\_char(&serial\_port, serial\_pin\_out, ((up >> 8) & 255));

char\_delay();

put\_char(&serial\_port, serial\_pin\_out, (down & 255));

char\_delay();

put\_char(&serial\_port, serial\_pin\_out, ((down >> 8) & 255));

char\_delay();

//

// off -> red

//

for (count = 0; count < 255; ++count) {

clear(led\_port,red);

for (pwm = count; pwm < 255; ++pwm)

PWM\_delay();

set(led\_port,red);

for (pwm = 0; pwm < count; ++pwm)

PWM\_delay();

}

//

// red -> green

//

for (count = 0; count < 255; ++count) {

set(led\_port,red);

clear(led\_port,green);

for (pwm = count; pwm < 255; ++pwm)

PWM\_delay();

clear(led\_port,red);

set(led\_port,green);

for (pwm = 0; pwm < count; ++pwm)

PWM\_delay();

}

//

// green -> blue

//

for (count = 0; count < 255; ++count) {

set(led\_port,green);

clear(led\_port,blue);

for (pwm = count; pwm < 255; ++pwm)

PWM\_delay();

clear(led\_port,green);

set(led\_port,blue);

for (pwm = 0; pwm < count; ++pwm)

PWM\_delay();

}

//

// blue -> on

//

for (count = 0; count < 255; ++count) {

set(led\_port,blue);

clear(led\_port,green);

clear(led\_port,red);

for (pwm = count; pwm < 255; ++pwm)

PWM\_delay();

set(led\_port,blue);

set(led\_port,green);

set(led\_port,red);

for (pwm = 0; pwm < count; ++pwm)

PWM\_delay();

}

//

// on -> off

//

for (count = 0; count < 255; ++count) {

set(led\_port,blue);

set(led\_port,green);

set(led\_port,red);

for (pwm = count; pwm < 255; ++pwm)

PWM\_delay();

clear(led\_port,blue);

clear(led\_port,green);

clear(led\_port,red);

for (pwm = 0; pwm < count; ++pwm)

PWM\_delay();

}

}

}