nicht finale abgabe

Protokoll 2

- Hannes Burmeister 20413
- Nils Herzig 19473

Table of contents

- nicht finale abgabe
- Protokoll 2
  - Table of contents
  - Setup / Info
  - 4 Experiment
    - Analysis of UART Data Frames - Send
      - 4.1.1
    - Analysis of UART Data Frames - Send & Receive
      - 4.2.1
    - Serial Calculator
      - 4.3.1
    - Serial Controller

Setup / Info

The source code files are located in the src directory. The code was developed and tested in a Fedora:38 Docker container using the `nielsenb/msp430-development-tools` repo. I've uploaded a prebuild docker image: `docker run -v $PWD:workdir --privileged nilsherzig/msp430env`.

To execute each task, you can use the respective make step. For instance:

```
cd src
make task411
```

This command cross compiles task `4.1.1`, flashes the resulting .elf, and establishes a connection via minicom with the appropriate baud rate. A gdbserver can be started using `sudo mspdebug tilib "gdb"` connect a gdb / r2 client like this:

```
gdb ./build/[task].elf # load debug symbols
(gdb) target remote localhost:2000
```

4 Experiment

Analysis of UART Data Frames - Send

4.1.1 Run via `make task411`

a) Write a C program that continuously emits the character "a" on the UART0. Configure the UART0 using SMCLK, 4800 baud, 8 data bits, no parity, and 1 stop bit. Use ACLK as the source for the baud rate generator.

c program:

```
#include <msp430g2553.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;

  if (CALBC1_1MHZ == 0xFF)
    while (1)
      ;
  DCOCTL = 0;
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;

  P1SEL = BIT2;
  P1SEL2 = BIT2;

  UCA0CTL1 |= UCSSEL_2;
  UCA0BR0 = 13;
  UCA0BR1 = 0;
  UCA0MCTL = UCOS16;
  UCA0CTL1 &= ~UCSWRST;

  while (1)
  {
    UCA0TXBUF = 'a';
    while (!(IFG2 & UCA0TXIFG))
      ;

    __delay_cycles(15000);
  }
}
```

b) Use a terminal program on the host to verify the correct operation of your program.

Using screen or minicom:

```
TERM=xterm sudo screen /dev/ttyACM1 4800
TERM=xterm sudo minicom -D /dev/ttyACM1 -b 4800
```

Analysis of UART Data Frames - Send & Receive

4.2.1

a) Write a C program that continuously waits for a lower case character received by UART RX and emits the corresponding upper case letter on UART TX. Configure the UART for 9600 bauds, 8 data bits, no parity, 1 stop bit. Configure SMCLK as the source for the baud-rate generator (SMCLK=1MHz). Use the Hyperterminal as remote station.

```
#include <msp430g2553.h>

int main(void)
{
  WDTCTL = WDTPW + WDTHOLD;

  if (CALBC1_1MHZ == 0xFF)
    while (1)
      ;
  DCOCTL = 0;
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;

  P1SEL = BIT1 + BIT2;
  P1SEL2 = BIT1 + BIT2;

  UCA0CTL1 |= UCSSEL_2;
  UCA0BR0 = 104;
  UCA0BR1 = 0;
  UCA0MCTL = UCBRS0;
  UCA0CTL1 &= ~UCSWRST;

  for (;;)
  {
    while (!(IFG2 & UCA0TXIFG))
      ;
```

```
    while (!(IFG2 & UCA0RXIFG))
      ;
    ;
    UCA0TXBUF = UCA0RXBUF - 32;
  }
}
```

b) Record the traffic on the input/output pins of the UART by using the LogicPort. Discuss the trace/data.

Serial Calculator

4.3.1

a) Write a C program that implements the Serial Calculator using a simple state machine (reuse UART settings from previous task). Hints: Use a strategy like "divide & conquer" for the structure of your program. Define the basic functions and concentrate on them, neglect anything else.

```
#include <msp430g2553.h>

#include <stdio.h>  // sprintf
#include <string.h> // strtok
#include <stdlib.h> // atoi

char inputCharArray[20];
int result;

void setup(void)
{
  WDTCTL = WDTPW + WDTHOLD;

  while (CALBC1_1MHZ == 0xFF)
    ;
  DCOCTL = 0;
  BCSCTL1 = CALBC1_1MHZ;
  DCOCTL = CALDCO_1MHZ;

  P1SEL = BIT1 + BIT2;
  P1SEL2 = BIT1 + BIT2;

  UCA0CTL1 = UCSWRST + UCSSEL_2;
  UCA0BR0 = 104;
  UCA0BR1 = 0;
```

```
  UCA0MCTL = UCBRS0;
  UCA0CTL1 &= ~UCSWRST;
}

void printNewLine()
{
  while (!(IFG2 & UCA0TXIFG))
    ;
  UCA0TXBUF = '\r';
  while (!(IFG2 & UCA0TXIFG))
    ;
  UCA0TXBUF = '\n';
}

void read()
{
  int ucContinue = 1;
  int ucCount = 0;

  while (ucContinue)
  {
    while (!(IFG2 & UCA0RXIFG))
      ;
    inputCharArray[ucCount] = UCA0RXBUF;

    while (!(IFG2 & UCA0TXIFG))
      ;
    UCA0TXBUF = inputCharArray[ucCount];

    ucContinue = (inputCharArray[ucCount] != '=');
    ucCount++;
    inputCharArray[ucCount] = 0;
  }
  printNewLine();
}

void compute()
{
  char *split;
  int num1, num2;

  split = strtok(inputCharArray, "+");
  if (split != NULL)
  {
    num1 = atoi(split);
    split = strtok(NULL, "=");
```

```
    if (split != NULL)
    {
      num2 = atoi(split);
    }
  }

  result = (num1 + num2);
}

void write()
{
  int ucContinue;
  int ucCount;

  char s[10];
  sprintf(s, "%d", result);

  ucCount = 0;
  ucContinue = (s[ucCount] != 0);
  while (ucContinue)
  {
    while (!(IFG2 & UCA0TXIFG))
      ;
    UCA0TXBUF = s[ucCount];
    ucCount++;
    ucContinue = (s[ucCount] != 0);
  }
  printNewLine();
}

int main()
{
  setup();
  for (;;)
  {
    read();
    compute();
    write();
  }
  return 0;
}
```

Serial Controller

a) Write a C program that implements the Serial Controller for the three independent channels of the RGB LED.