EMB 2010 Team3 Exercise2

1 Some setup

This exercise is about making one big loop out of UART 0, 1, 2. It is done by sending a character from the pc into the first UART, UART0 which reads the character and sends it to UART2, the UART2 reads the input and sends the character back to the pc. When the character is sent further on in the system one is added to the value. Ex. The PC sends the character 'a'. UART0 receives this value, adds one to it ('b') and sends it further on in the loop. UART2 receives the character, adds one to it and returns it to the PC, where the value 'c' appears on the screen.
Unfortunately we could not get UART1 to work on the 32-bit board we have, otherwise it would also have been part of the loop. From the 'LPC2489 Users Guide':
Full modem RS232 on UART1 (cannot be used on 32-bit data bus cpu boards, but RxD2/TxD2 can alternatively be connected to the RS232 interface)

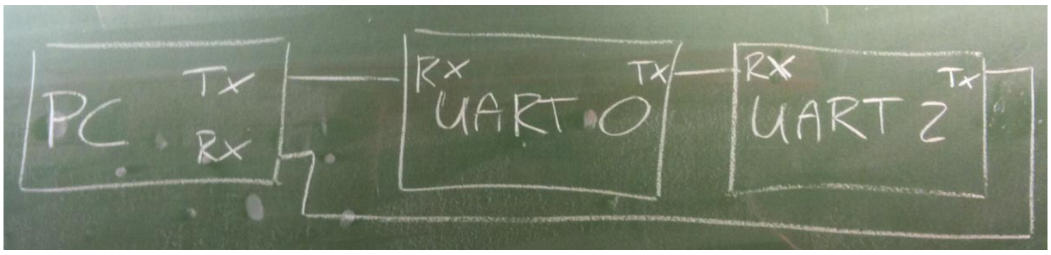


Figure 1: How it is meant that the hardware should be putted together.

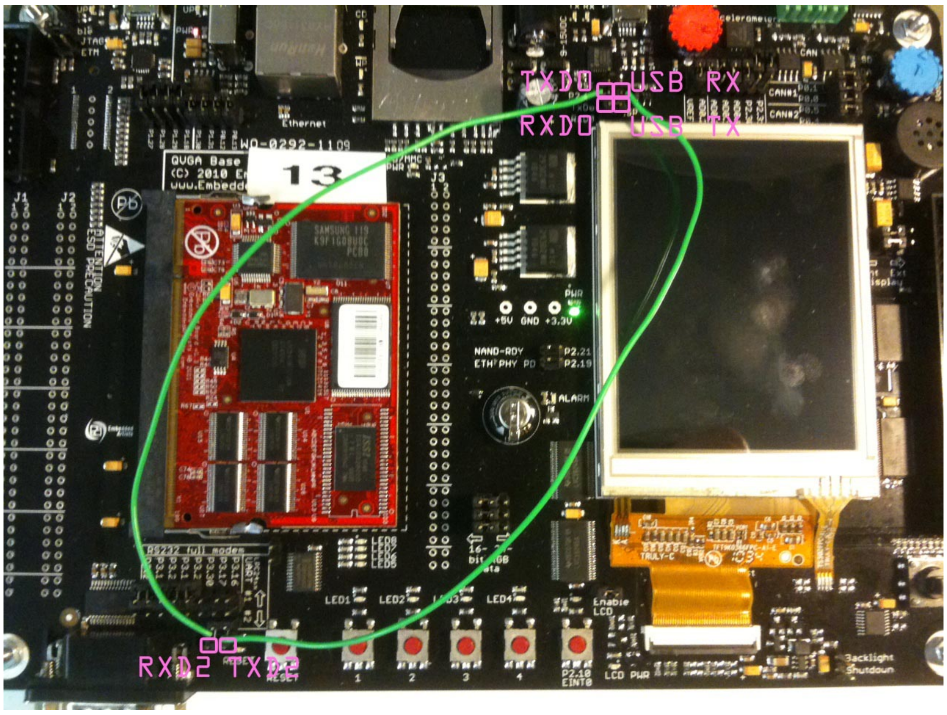
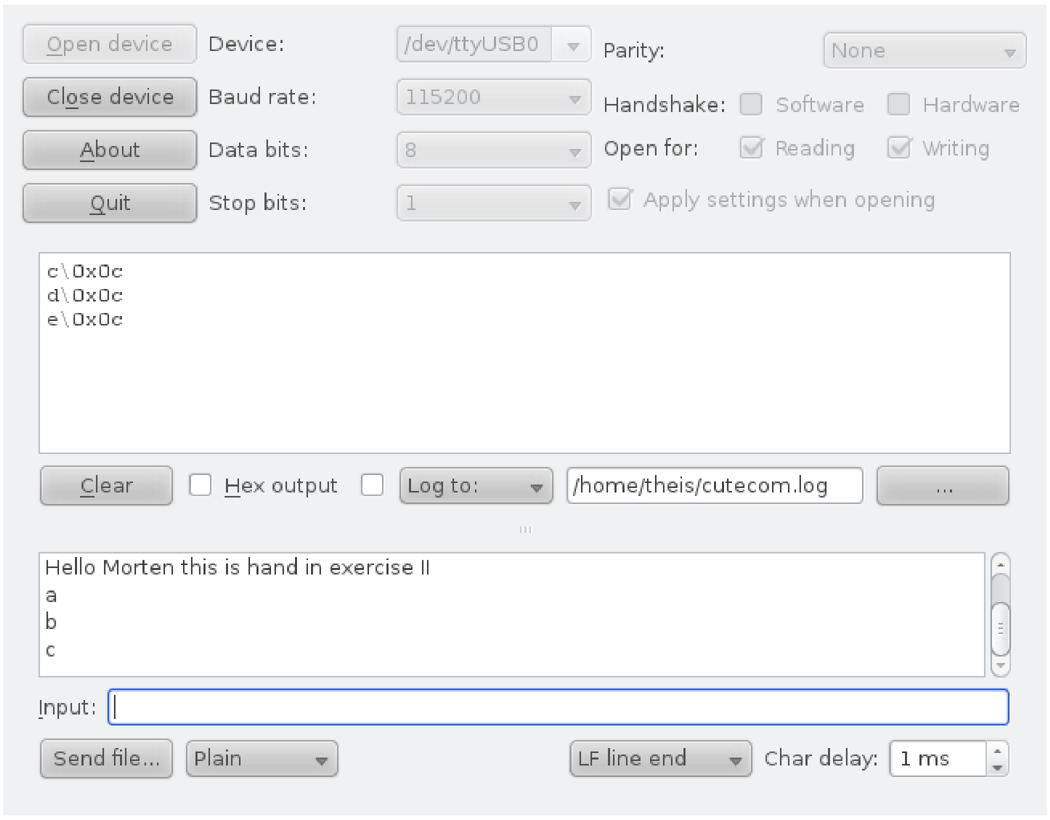


Figure 2: How the hardware implementation is realised.



Device: /dev/ttyUSB0
Baud rate: 115200
Data bits: 8
Stop bits: 1
Parity: None
Handshake: ☐ Software ☐ Hardware
Open for: ✓ Reading ✓ Writing
✓ Apply settings when opening

```
c\0x0c
d\0x0c
e\0x0c
```

☐ Hex output ☐ Log to: /home/theis/cutecom.log

```
Hello Morten this is hand in exercise II
a
b
c
```

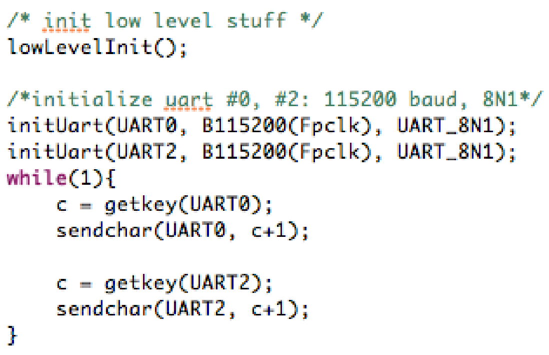
Input:

Plain LF line end Char delay: 1 ms

Figure 3: What is sent and what is received. Note: the "system" is also tried with other baud rate setups and works fine.

2 Some code

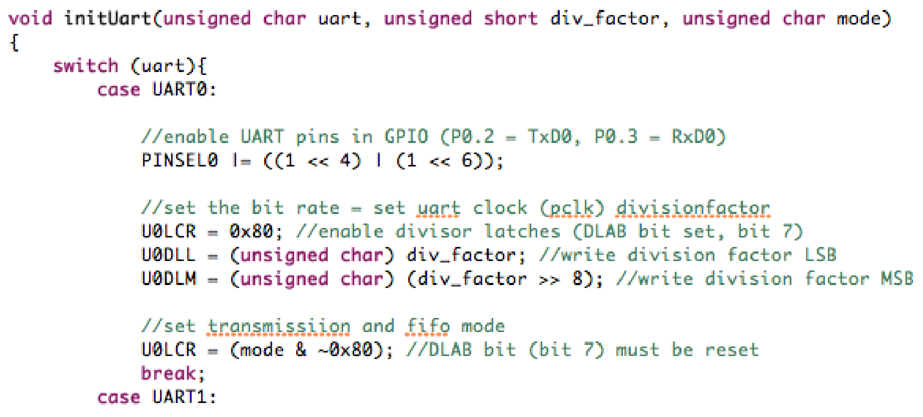
In the start of the code some peripherals are set up in the lowLevelInit function, such as power enable registers, clock speed etc. 'initUart' is called twice to set up both UART0 and UART2.
In the 'system loop' the program simply waits for an input from the PC, sends the character to the UART2 (HW linked). The UART2 reads the character and sends it back to the pc.



```
/* init low level stuff */
lowLevelInit();

/*initialize uart #0, #2: 115200 baud, 8N1*/
initUart(UART0, B115200(Fpclk), UART_8N1);
initUart(UART2, B115200(Fpclk), UART_8N1);
while(1){
    c = getkey(UART0);
    sendchar(UART0, c+1);

    c = getkey(UART2);
    sendchar(UART2, c+1);
}
```

Figure 4: The main loop and initial setup.

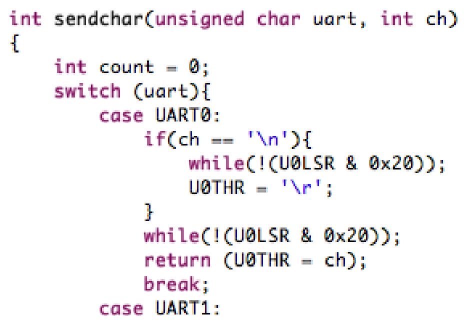
The initialization is only changed a little compared to the given example. Simply a switch-case is made, to setup only the selected UART.



```
void initUart(unsigned char uart, unsigned short div_factor, unsigned char mode)
{
    switch (uart){
        case UART0:

            //enable UART pins in GPIO (P0.2 = TxD0, P0.3 = RxD0)
            PINSEL0 |= ((1 << 4) | (1 << 6));

            //set the bit rate = set uart clock (pclk) divisionfactor
            U0LCR = 0x80; //enable divisor latches (DLAB bit set, bit 7)
            U0DLL = (unsigned char) div_factor; //write division factor LSB
            U0DLM = (unsigned char) (div_factor >> 8); //write division factor MSB

            //set transmission and fifo mode
            U0LCR = (mode & ~0x80); //DLAB bit (bit 7) must be reset
            break;
        case UART1:
```

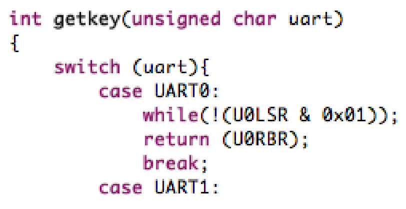
Figure 5: Initialisation of UARTx.

The send and receive functions are also just expanded to send and receive characters to/from the selected UART port (switch-cases are used).



```
int sendchar(unsigned char uart, int ch)
{
    int count = 0;
    switch (uart){
        case UART0:
            if(ch == '\n'){
                while(!(U0LSR & 0x20));
                U0THR = '\r';
            }
            while(!(U0LSR & 0x20));
            return (U0THR = ch);
            break;
        case UART1:
```

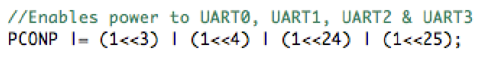
Figure 6: Send a character to UARTx.



```
int getkey(unsigned char uart)
{
    switch (uart){
        case UART0:
            while(!(U0LSR & 0x01));
            return (U0RBR);
            break;
        case UART1:
```

Figure 7: Read a character from UARTx.

The power (PCUART) is by default enabled for channel 0 and 1, but disabled for channel 2 and 3. Therefore all of these is enabled in the framework.c file (the lowLevelInit function).



```
//Enables power to UART0, UART1, UART2 & UART3
PCONP |= (1<<3) | (1<<4) | (1<<24) | (1<<25);
```

Figure 8: Power on all UART modules.