

Lab 6

UART

Keaton Clark

October 19, 2022

Code

echo2c.c

```
1 #include <avr/io.h>
2 #include <stdio.h>
3 #include "io.h"
4
5 unsigned char U0kbhit() {
6     return (unsigned char)uart_available();
7 }
8
9 unsigned char U0getchar() {
10     return (unsigned char)uart_read_char();
11 }
12
13 void U0putchar(unsigned char U0pdata) {
14     uart_put_char(U0pdata, stdout);
15 }
```

echo3c.c

```
1 #include <avr/io.h>
2 #include <stdio.h>
3 #include "io.h"
4
5 unsigned char U0kbhit() {
6     return (unsigned char)uart_available();
7 }
8
9 unsigned char U0getchar() {
10     return (unsigned char)uart_read_char();
11 }
12
13 void U0putchar(unsigned char U0pdata) {
14     uart_put_char(U0pdata, stdout);
15 }
16 int main () {
17     uart_init(9600);
18     for (;;) {
19         printf("0x%.2X\n", uart_read_char());
20     }
21 }
```

Relavant lines from io.c because I had already written this functionality earlier and made it compatable with the get and put function pointers in C FILE structs which take types "int(FILE*)" and "int(char,FILE*)" respectively.

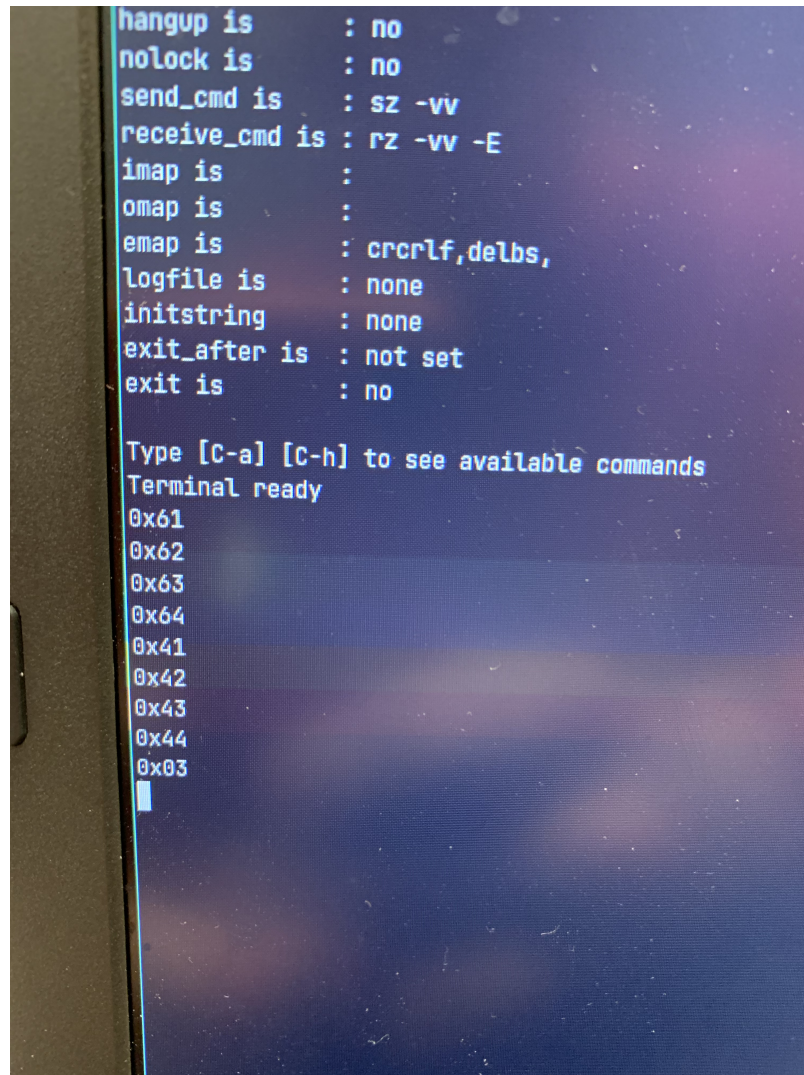
```

145 int uart_put_char(char c, FILE *stream);
146 int uart_get_char(FILE *stream);
147 FILE uart_str = {
148     .put = uart_put_char,
149     .get = uart_get_char,
150     .flags = 0b10|0b1, //Read | Write
151     .udata = 0,
152 };
153 void uart_init(uint16_t baud_rate) {
154     stdout = stdin = &uart_str; //set stdout to our FILE
    stream we made above ^
155     UBRRL = (F_CPU / (16UL * baud_rate)) - 1;
156     UCSRB = _BV(TXEN) | _BV(RXEN);
157 }
158 int uart_put_char(char c, FILE *stream) {
159     if (c == '\n') {
160         uart_put_char('\r', stream);
161     }
162     loop_until_bit_is_set(UCSR0A, UDRE0);
163     UDR0 = c;
164     return 0;
165 }
166 int uart_read_char() {
167     loop_until_bit_is_set(UCSR0A, RXC0);
168     if (UCSR0A & _BV(FE0)) return _FDEV_EOF;
169     if (UCSR0A & _BV(DOR0)) return _FDEV_ERR;
170     return UDR0;
171 }
172 int uart_available() {
173     return UCSRB & _BV(RXC0);
174 }

```

Result

From keypress sequence: *abcdABCD* < *ctrl - c* >



```
hangup is      : no
nolog is       : no
send_cmd is    : sz -vv
receive_cmd is : rz -vv -E
imap is        :
omap is        :
emap is        : crclrf,delbs,
logfile is     : none
initstring     : none
exit_after is  : not set
exit is        : no

Type [C-a] [C-h] to see available commands
Terminal ready
0x61
0x62
0x63
0x64
0x41
0x42
0x43
0x44
0x03
█
```