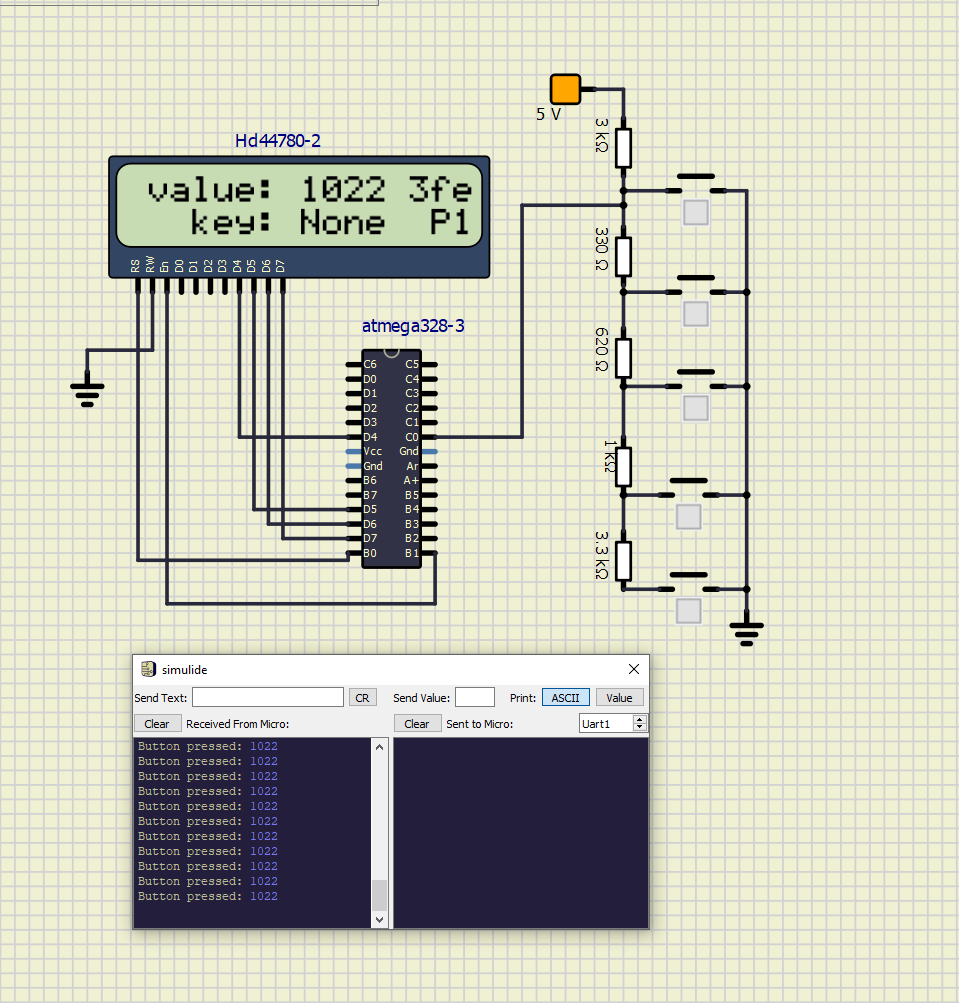
|  |  |  |  |
| --- | --- | --- | --- |
| **Push button** | **PC0[A0] voltage** | **ADC value (calculated)** | **ADC value (measured)** |
| Right | 0 V | 0 | 0 |
| Up | 0.495 V | 101 | 101 |
| Down | 1.202 V | 246 | 245 |
| Left | 1.970 V | 403 | 402 |
| Select | 3.1818 | 651 | 650 |
| none | 5 | 1023 | 1022 |

**DE2 UART** Jiří Vitouš

|  |
| --- |
|  |
| **Function name** | **Function parameters** | **Description** | **Example** |
| uart\_init | UART\_BAUD\_SELECT(9600, F\_CPU) | Initialize UART to 8N1 and set baudrate to 9600 Bd | uart\_init(UART\_BAUD\_SELECT(9600, F\_CPU)); |
| uart\_getc | none | Get char from input buffer | uint8\_t c= uart\_getc(); |
| uart\_putc | char | Put character into output buffer to send | uart\_putc(‘c’); |
| uart\_puts | const char\* | Put character array one by one into output buffer | uart\_puts(“Ahoj”); |
|  |  |  |  |

|  |
| --- |
|  |
| **Operation** | **Register(s)** | **Bit(s)** | **Description** |
| Voltage reference | ADMUX | REFS1:0 | 01: AVcc voltage reference, 5V |
| Input channel | ADMUX | MUX3:0 | 0000: ADC0, 0001: ADC1, ... |
| ADC enable | ADCSRA | ADEN | Zapnout ADC |
| Start conversion | ADCSRA | ADSC | Začít s konverzí vstupní úrovně napětí na dignitální reprezentaci |
| ADC interrupt enable | ADCSRA | ADIE | Umožnit přerušení |
| ADC clock prescaler | ADCSRA | ADPS2:0 | 000: Division factor 2, 001: 2, 010: 4, ... |
| ADC result | ADLAR | ADC9:0 | Hodnota po konverzi |
|  |  |  |  |



ISR(ADC\_vect)

{

*uint16\_t* value = 0;

char lcd\_string[4] = "0000";

value = ADC; // Copy ADC result to 16-bit variable

*itoa*(value, lcd\_string, 10); // Convert to string in decimal

lcd\_gotoxy(8, 0); // set cursor to position 'a'

lcd\_puts(" "); // clear space for new number

lcd\_gotoxy(8, 0); // set cursor to position 'a'

lcd\_puts(lcd\_string); // send string

uart\_puts("Button pressed: "); //send text preceding value

uart\_puts(lcd\_string); // send character string over UART (value of adc)

uart\_puts("\n"); // end line

lcd\_gotoxy(13, 0); // set cursor to position 'b'

lcd\_puts(" "); // clear space for new number

lcd\_gotoxy(13, 0); // set cursor to position 'b'

*itoa*(value, lcd\_string, 16); // Convert to string in hex

lcd\_puts(lcd\_string); // send string

// code for printing button name pressed

if(value<50)

{

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

lcd\_gotoxy(8, 1);

lcd\_puts("Right");

}

else if((value>=50) & (value < 170))

{

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

lcd\_gotoxy(8, 1);

lcd\_puts("Up");

}

else if((value>=170) & (value < 350))

{

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

lcd\_gotoxy(8, 1);

lcd\_puts("Down");

}

else if((value>=350) & (value < 500))

{

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

lcd\_gotoxy(8, 1);

lcd\_puts("Left");

}

else if((value>=500) & (value < 800))

{

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

lcd\_gotoxy(8, 1);

lcd\_puts("Select");

}

else

{

lcd\_gotoxy(8, 1);

lcd\_puts(" ");

lcd\_gotoxy(8, 1);

lcd\_puts("None");

}

// parity bit computation for even parity of whole “value”, if odd then add parity^=1;

*uint8\_t* parity=0;

for(*uint8\_t* i=0;i<16;i++) // go through the whole length of “value”

{

parity^= (value & 0x01); // parity XOR (if last bit is one)

value>>=1; // bit shift value right

}

lcd\_gotoxy(15, 1); // move to some free location

if(parity==1) // decide if parity bit is zero or one

lcd\_putc('1'); // put parity bit onto display

else

lcd\_putc('0'); // put parity bit onto display

}

