|  |  |
| --- | --- |
|  | |
| **LCD signal(s)** | | **AVR pin(s)** | **Description** | | |
| RS | | PB0 | Register selection signal. Selection between Instruction register (RS=0) and Data register (RS=1) | | |
| R/W | | GND | Nastavuje mód komunikace čtení/zápis 0-zápis 1- čtení | | |
| E | | PB1 | Latch dat do paměti displaye po přeskoku 0-1 si display zapíše hodnoty do paměti | | |
| D[3:0] | | - | Datové piny | | |
| D[7:4] | | PD7-PD4 | Datové piny (Je možné použít pro 4 pinovou komunikaci) | | |
| What is the ASCII table? What are the values for uppercase letters A to Z, lowercase letters a to z, and numbers 0 to 9 in this table?  ASCII tabulka je převodní tabulka která přiřazuje osmibitovému slovu (číslu) konkrétní znak.  Čísla 48-57 odpovídají číslicím 0-9  Čísla 65-90 odpovídají velkým písmenům A-Z  Čísla 97-122 odpovídají malým písmenům a-z | | | | | |
|  |
| **Function name** | **Function parameters** | | | **Description** | **Example** |
| lcd\_init | LCD\_DISP\_OFF LCD\_DISP\_ON LCD\_DISP\_ON\_CURSOR LCD\_DISP\_ON\_CURSOR\_BLINK | | | Display off  Display on   Zaptnout kurzor Zaptnout blikající kurzor | lcd\_init(LCD\_DISP\_OFF); |
| lcd\_clrscr | Void | | | Vymazat disaply a nastavit kurzor na první pozici | lcd\_clrscr(); |
| lcd\_gotoxy | x,y | | | Posun kurzoru na pozici x,y | lcd\_gotoxy(2,2); |
| lcd\_putc | c | | | Dej znak c na momentální pozici | lcd putc(‘d’); |
| lcd\_puts | Char \*s | | | Zobraz řetěžec znaků | lcd\_puts(“Hello”); |
| lcd\_command | cmd | | | Poslat instrukci | lcd\_command(0xC4); |
| lcd\_data | data | | | Poslat data | lcd\_data(0xFF); |

ISR(TIMER2\_OVF\_vect)

{

static *uint8\_t* number\_of\_overflows = 0;

static *uint8\_t* tens = 0; // Tenths of a second

static *uint8\_t* secs = 0; // Seconds

static *uint8\_t* minutes = 0; // Minutes

char lcd\_string[2] = " "; // String for converting numbers by itoa()

int sq = 0;

number\_of\_overflows++;

if (number\_of\_overflows >= 6)

{

// Do this every 6 x 16 ms = 100 ms

number\_of\_overflows = 0;

if (tens < 9)

tens++;

else

{

secs++;

tens = 0;

}

if (secs > 59)

{

secs = 0;

minutes++;

}

if (minutes > 59)

{

minutes = 0;

}

lcd\_gotoxy(1, 0);

if (minutes < 10) // if less than 10 minutes write to second position in seconds

{

lcd\_putc('0');

lcd\_gotoxy(2, 0);

}

*itoa*(minutes, lcd\_string, 10); // convert and output minutes

lcd\_puts(lcd\_string);

lcd\_gotoxy(4, 0);

if (secs < 10) // if less than 10 seconds write to second position in seconds

{

lcd\_putc('0');

lcd\_gotoxy(5, 0);

}

*itoa*(secs, lcd\_string, 10); // convert and ouput seconds

lcd\_puts(lcd\_string);

lcd\_gotoxy(7, 0);

*itoa*(tens, lcd\_string, 10); // convert and output tenths of seconds

lcd\_puts(lcd\_string);

sq = (int)secs \* (int)secs; // create square of secs and output onto display

lcd\_gotoxy(11, 0);

*itoa*(sq, lcd\_string, 10);

lcd\_puts(lcd\_string);

}

}

ISR(TIMER0\_OVF\_vect)

{

static *uint8\_t* ovf = 0;

static *uint8\_t* symbol = 0;

static *uint8\_t* position = 0;

static *uint8\_t* ovf2 = 1;

ovf++;

if ((20 \* (int)ovf2 - 16 \* (int)ovf) < 0) // basically a rounding algorithm to change symbol / position every approx 20 ms

{

ovf2++;

if (symbol < 4)

symbol++;

else

{

symbol = 0;

position++;

}

}

if (position > 9 || ovf >= 60) // if last position or overflow (to compensate for rounding)

{

position = 0;

symbol = 0;

for (*uint8\_t* i = 0; i <= 9; i++) // clear bar

{

lcd\_gotoxy(1 + i, 1);

lcd\_putc(10);

}

ovf = 0;

ovf2 = 1;

}

else

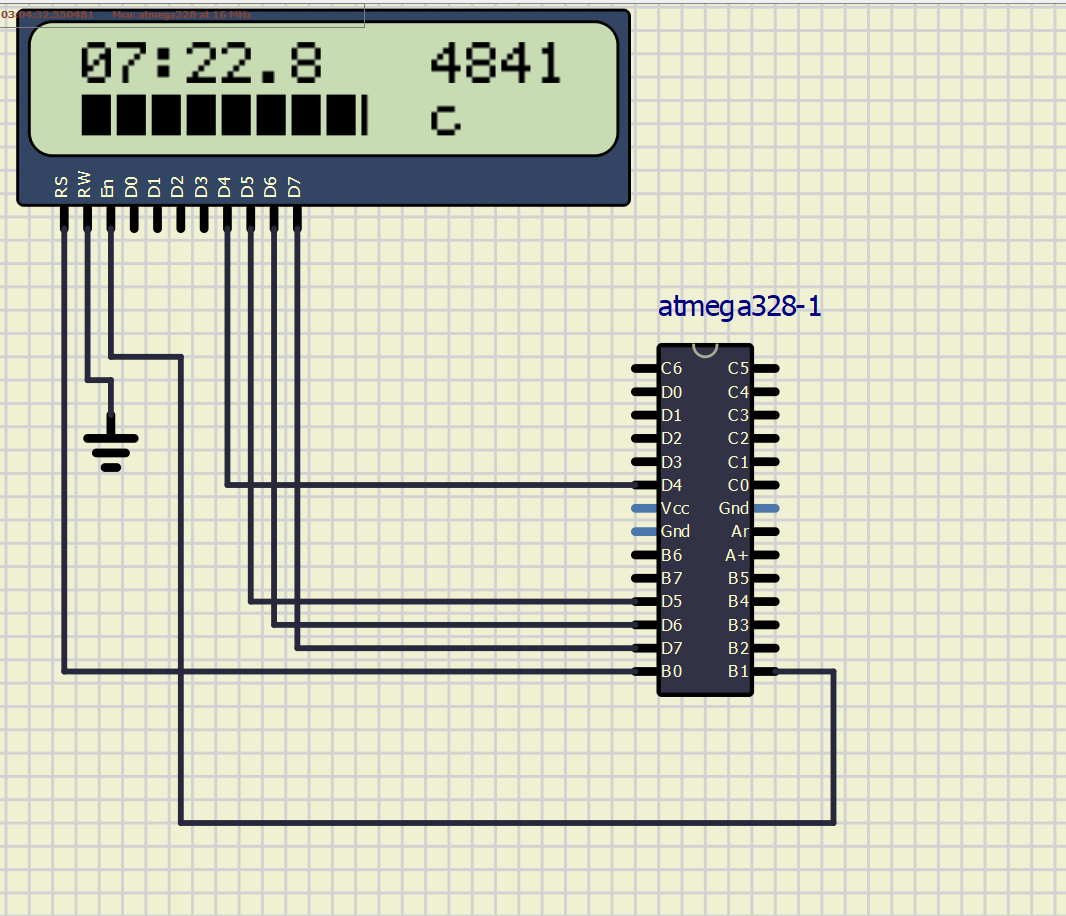
{

lcd\_gotoxy(1 + position, 1);

lcd\_putc(symbol);

}

}



Řešení se zobrazením druhé mocniny sekund a bar grafem