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)

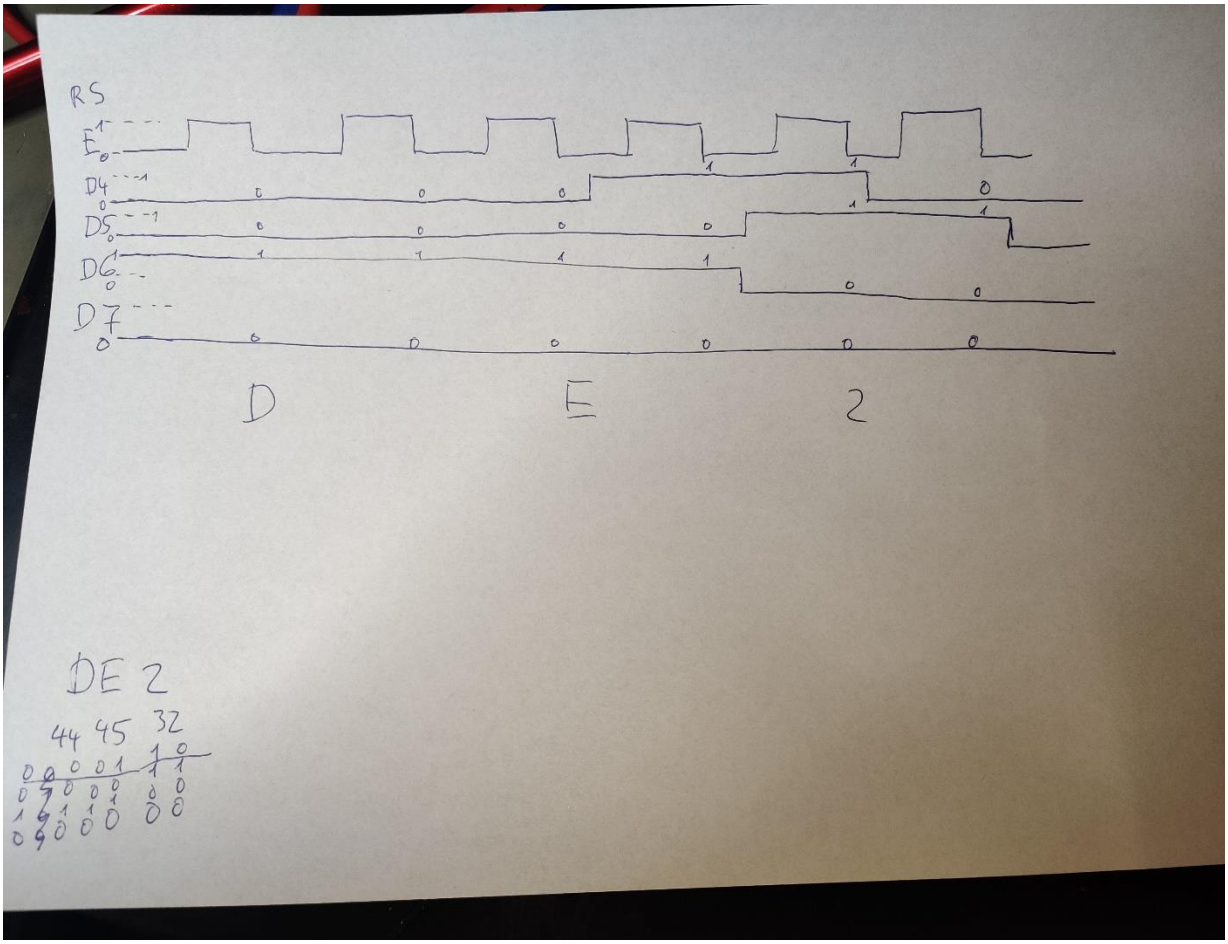
What is the ASCII table? What are the values for uppercase letters `A` to `Z`, lowercase letters `a` to `z`, and numbers `0` to `9` in this table?

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



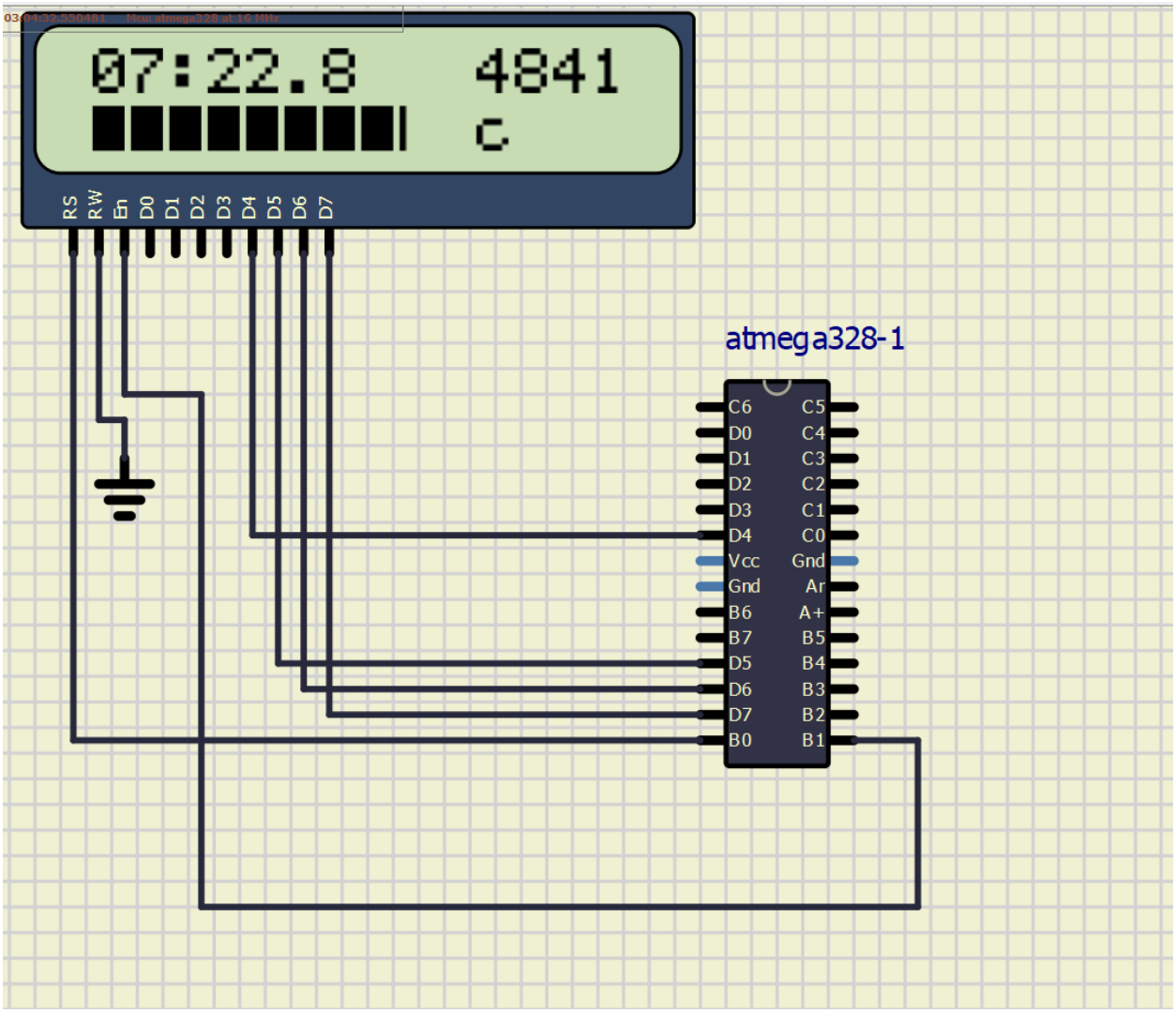
Function name	Function parameters	Description	Example

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 řetězec 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