Hoofdstuk 6 – Seriële communicatie, het OLED display en de SD kaart

Thuisopdracht 1

a) In de P1 companion standard van de DSMR vind je:

5.1 Physical connector

The P1 port connector type is RJ12. The Metering System holds a female connector; the OSM (Other Service Module) connects via standard RJ12 male plug.

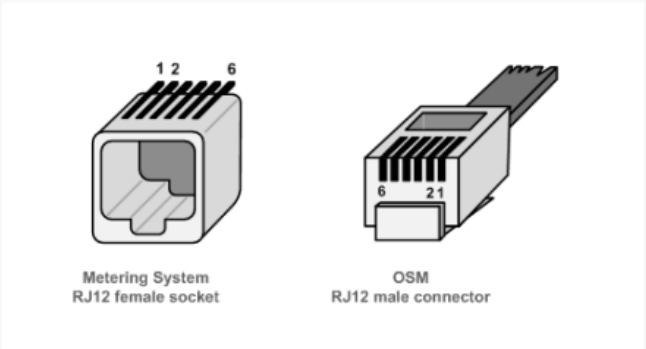


Figure 5-1: Physical connectors.

The P1 connector in the Metering System must be accessible at all times and should not be sealed or protected by a sealed cover. The P1 pin assignment is detailed in the table below:

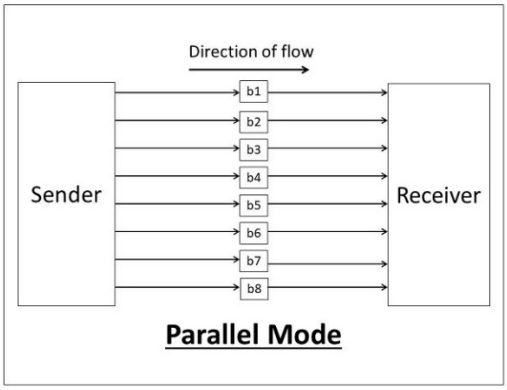
Pin #	Signal name	Description	Remark
1	+5V	+5V power supply	Power supply line
2	Data Request	Data Request	Input
3	Data GND	Data ground	
4	n.c.	Not connected	
5	Data	Data line	Output. Open collector
6	Power GND	Power ground	Power supply line

Table 5-1: Physical connector pin assignment

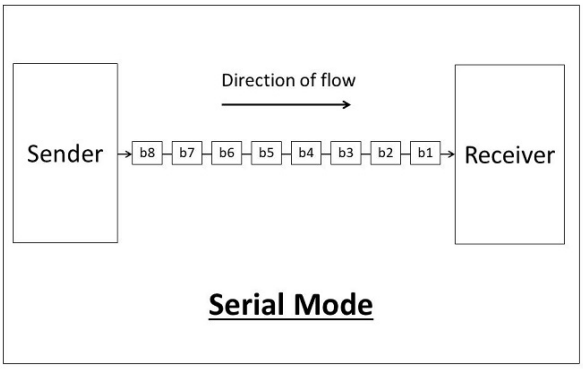
c) De dataoverdracht gaat via pin 5. Op pin 2 wordt door de microcontroller een signaal gezet zodra deze om data vraagt via de P1-poort.

Thuisopdracht 2:

a) Bij parallelle data-overdracht wordt er via meerdere kabels tegelijk data van A naar B verzonden:



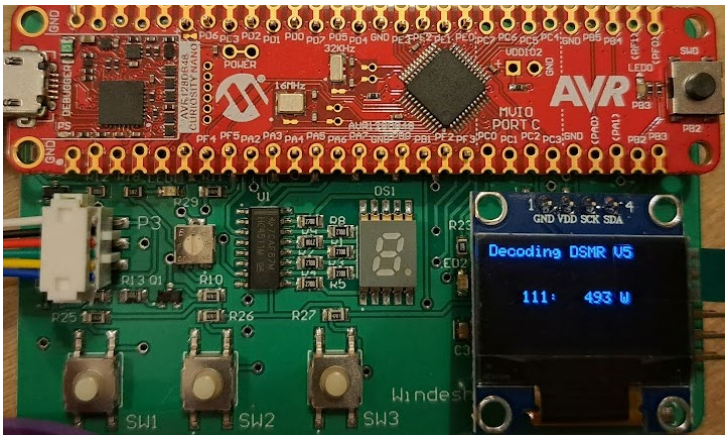
b) Bij seriële data-overdracht wordt er via één kabel, achter elkaar data van A naar B gestuurd:



c) Voordelen serieel: minder materiaal nodig.
Nadelen serieel: verzender en ontvanger moeten gesynchroniseerd worden. Het kost meer tijd om de data te versturen, want het gebeurt één voor één.

Voordelen parallel: het is makkelijk te implementeren of programmeren. Het is snel, want alle data wordt tegelijk verzoenden.
Nadelen parallel: veel materiaal nodig, dus duur. Tussen de parallelle kanalen kan interferentie ontstaan.

Thuisopdracht 3:



Thuisopdracht 4:

Op de SD-kaart staat een log-file, waarin het huidige verbruik staat weggeschreven:

```
0;  530
1;  530
2;  530
3;  530
```

Deze kan je inlezen in excel en bijvoorbeeld afbeelden in een grafiek.

Sluipverbruik zou je ’s nachts kunnen aflezen, als er geen apparaten actief gebruikt worden.

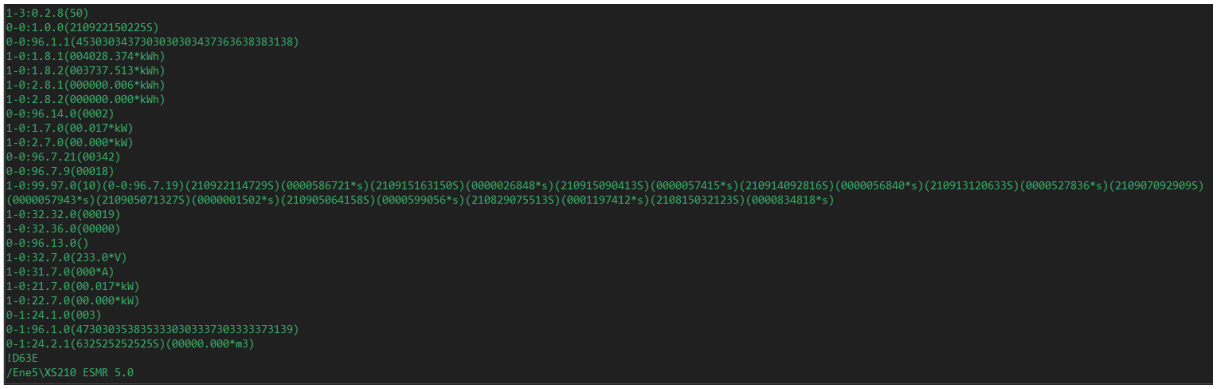
Thuisopdracht 5:

a) Baud is een eenheid die het aantal signaalwisselingen ofwel symbolen per seconde op een datatransmissiekanaal, of meer algemeen, op een informatieverbinding aangeeft.
Voorbeeld: een modem van 2400 bit/s seint, door gebruik te maken van symbolen die vier bits informatie overdragen, met een snelheid van 600 baud.

b) Versie 5.0.2: Hoofdstuk 6.1: 115200 baud.

c) Hoofdstuk 6.13 geeft een voorbeeld van een P1 telegram.

d)



Labopdracht 1:

a) Maak gebruik van UARTtemplate.c en zet in de main:

```
int main(void)
{
    USART3_init();

    while (1)
    {
        USART3_sendString("Studentnummer \r\n");
        _delay_ms(1000);
    }
}
```

b) Verander de baudrate naar 9600:

```
void USART3_init(void)
{
    // AVR128DB48
    PORTB.DIRSET = PIN0_bm;
    PORTB.DIRCLR = PIN1_bm;

    USART3.BAUD = (uint16_t)(USART3_BAUD_RATE(9600));

    USART3.CTRLC = USART_CHSIZE0_bm | USART_CHSIZE1_bm;

    USART3.CTRLB |= USART_TXEN_bm;
}
```

Meet op pin PB0 de output.

Labopdracht 2:

a) Er wordt een variabele i aangemaakt, die elke seconde op het display wordt geprint en daarna met 1 wordt verhoogd.

c) Stel via de regel GLDC_GoTo(10,25) andere coördinaten in.

```
int main(void)
{
	GLCD_Setup();
	GLCD_SetFont(Font5x8, 5, 8, GLCD_Overwrite);

	int i = 0;
	char buffer[10];

	GLCD_GotoXY(10,25);
	GLCD_PrintString("Hello World!");

	while(1)
	{
		GLCD_GotoXY(90,50);
		sprintf(buffer, "%3d", i);
		GLCD_PrintString(buffer);
		GLCD_Render();
		_delay_ms(1000);
		i++;
	}
}
```

Labopdracht 3:

Deelopdracht a)

```
int main(void)
{
	GLCD_Setup();
	GLCD_SetFont(Font5x8, 5, 8, GLCD_Overwrite);

	int i = 0;
	char buffer[10];

	while(1)
	{
		GLCD_GotoXY(90,50);
		sprintf(buffer, "%3d", i);
		GLCD_PrintString(buffer);
		GLCD_Render();
		_delay_ms(1000);
		i++;

		if(i==10)
		{
			i = 0;
		}
	}
}
```

Deelopdracht b)

```
int main(void)
{
	GLCD_Setup();
	GLCD_SetFont(Font5x8, 5, 8, GLCD_Overwrite);

	int i = 0;
	char buffer[10];

	while(1)
	{
		int knop = !(PORTE.IN & PIN1_bm);

		GLCD_GotoXY(90,50);
		sprintf(buffer, "%3d", i);
		GLCD_PrintString(buffer);
		GLCD_Render();

		if(knop)
		{
		_delay_ms(1000);
		i++;
		}


		if(i==10)
		{
			i = 0;
		}
	}
}
```

Deelopdracht c)

Hier zul je ervoor moeten zorgen dat je microcontroller kijkt wat de huidige stand van de knop is, maar ook onthoud wat de vorige stand is. Alleen als de knop nu is ingedrukt, en daarvoor niet was ingedrukt moet er opgeteld worden.

```
int main(void)
{
	GLCD_Setup();
	GLCD_SetFont(Font5x8, 5, 8, GLCD_Overwrite);

	int i = 0;
	char buffer[10];
	int vorigeknop = 0;


	while(1)
	{
		int knop = !(PORTE.IN & PIN1_bm);

		if(knop&&!vorigeknop)
		{
		i++;
		}

		if(i==10)
		{
			i = 0;
		}

		vorigeknop = knop;

		GLCD_GotoXY(90,50);
		sprintf(buffer, "%3d", i);
		GLCD_PrintString(buffer);
		GLCD_Render();
	}
}
```

Labopdracht 4:

Deelopdracht a)

```
int main(void)
{
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
	TCA0.SINGLE.PER = 156; // tick every 10 msec = 100 hz: clock = 4000000 / 256 =
15625 hz
	TCA0.SINGLE.CTRLA = (TCA_SINGLE_CLKSEL1_bm | TCA_SINGLE_CLKSEL2_bm);
	TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm;

	sei();

	FATFS FatFs;

	// init sdcard
	UINT bw;
	f_mount(&FatFs, "", 1);		// Give a work area to the FatFs module

	// open file
	FIL *fp = (FIL *)malloc(sizeof (FIL));
	if (f_open(fp, "file.txt", FA_WRITE | FA_CREATE_ALWAYS) == FR_OK) // Create a
file
	{
		char *text = "Naam en studentnummer\r\n";
		f_write(fp, text, strlen(text), &bw);	// Write data to the file
		f_close(fp);// Close the file
	}

	while(1)
	{
	}
}
```

Deelopdracht b)

Hier wil je dat de waarde van de schakelaars elke seconde op een nieuwe regel op de SD-kaart wordt opgeslagen. Je moet dus zorgen dat de SD-kaart niet elke keer op dezelfde regel schrijft en je moet in elke while loop schrijven:

```
int main(void)
{
	TCA0.SINGLE.INTCTRL = TCA_SINGLE_OVF_bm;
	TCA0.SINGLE.PER = 156; // tick every 10 msec = 100 hz: clock = 4000000 / 256 =
15625 hz
	TCA0.SINGLE.CTRLA = (TCA_SINGLE_CLKSEL1_bm | TCA_SINGLE_CLKSEL2_bm);
	TCA0.SINGLE.CTRLA |= TCA_SINGLE_ENABLE_bm;

	sei();

	FATFS FatFs;

	// init sdcard
	UINT bw;
	f_mount(&FatFs, "", 1);		// Give a work area to the FatFs module

	char buffer[100];

	while(1)
	{
		int SW3 = !(PORTE.IN & PIN1_bm);
		int SW2 = !(PORTE.IN & PIN2_bm);
		int SW1 = !(PORTE.IN & PIN3_bm);

		sprintf(buffer, "[%d %d %d]\r\n", SW1,SW2,SW3);

		// open file
		FIL *fp = (FIL *)malloc(sizeof (FIL));
		if (f_open(fp, "file.txt", FA_WRITE | FA_OPEN_APPEND) == FR_OK) // Let
op: open_append om elke keer op een nieuwe regel te schrijven
		{
			f_write(fp, buffer, strlen(buffer), &bw);	// Write data to
the file
			f_close(fp);// Close the file
		}

		_delay_ms(1000);
	}
}
```

Labopdracht 5:

Gebruik de template-file P1ShowCurrentUseOLED en probeer ter oefening iets anders te printen dan “current use”.