

Formål

Formålet med denne øvelse er at anvende UART modtage interrupts.

UART driveren fra LAB11 udvides, således at UART modtage interrupts kan enables under initieringen. Desuden er det formålet at vise, hvordan en UART kan implementeres i ren software.

Materiale

Lektion 18: "UART interrupts mm."

Lærebogen (Mazidi): Kapitel 11 "AVR serial port programming in assembly and C".

"Det blå hæfte" = "Mega2660 I/O registers": Siderne 51 til 57.

Øvelsen

I denne øvelse vil vi:

- Udvide UART driveren fra LAB11, sådan at man under initieringen vil kunne vælge, om der skal genereres et interrupt, hver gang et nyt tegn modtages.
- Skrive et testprogram, der afprøver den udvidede UART driver.
Testprogrammet skal reagere på modtagne tegn fra PC'ens terminalprogram og toggle ("skifte tilstand på") bestemte lysdioder, hvis bestemte tegn modtages.
Hertil anvendes UART modtage-interrupts.
I programmets main-sløjfe vil vi løbende aflæse temperaturføleren på "Mega25 I/O Shield" og vise denne temperatur i terminalprogrammet (øvelsens del 1b).
- Skrive og teste en UART sender implementeret i software (Software UART).

Den tidligere udviklede LED driver (fra LAB7) vil blive genanvendt.

Desuden anvendes drivere for aflæsning af "Mega2560 I/O Shield"s temperaturføler.

Disse skal ikke skrives i øvelsen, men "udleveres".

Del 1a: UART driver med modtager-interrupt

I øvelsen vil vi udvide UART-driveren fra LAB11 ved at tilføje yderligere en parameter til initierings-funktionen **InitUART()**.

Inden vi gør det, vil vi tage kopier af vores LAB11 UART driver ("uart.c" og "uart.h") og omdøbe dem til "uart_int.c" og "uart_int.h".

Herved bevarer vi vores oprindelige driver uforandret.

Den nye drivers header-fil "uart_int.h" skal nu se således ud:

```
/******  
 * "uart_int.h":  
 * Header file for Mega2560 UART driver. *  
 * Using UART 0.  
 * If parameter Rx_Int <> 0 :  
 * Receiver interrupt will be enabled  
 *  
 * Henning Hargaard, 11/11 2015  
 *****/  
void InitUART(unsigned long BaudRate, unsigned char DataBit, char Parity, unsigned char Rx_Int);  
unsigned char CharReady();  
char ReadChar();  
void SendChar(char Tegn);  
void SendString(char* Streng);  
void SendInteger(int Tal);  
/******
```

Eneste forskel fra den "gamle driver" er, at **InitUART()** har fået en ekstra parameter (unsigned char RX_int). Ved hjælp af denne parameter fortæller vi **InitUART()** om vi ønsker UART modtage-interruptet enabled eller disabled.

Hvis RX_int er 0, skal UART modtage-interruptet være disabled (som i "den gamle driver").

Hvis RX_int er forskellig fra 0, skal UART modtage-interruptet være enabled.

"uart_int.h" kan hentes fra på Blackboard.

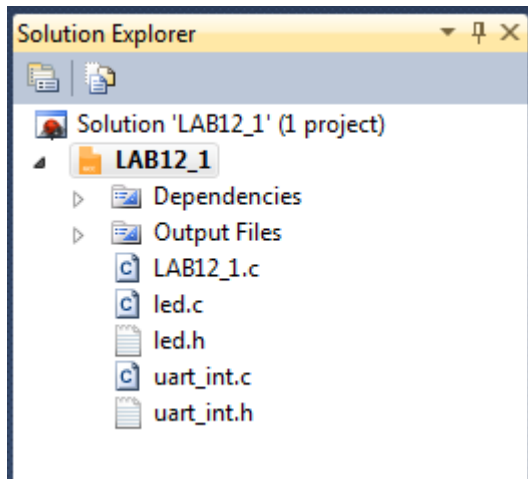
Lav nu ændringer i funktionen **InitUART()** i implementeringsfilen "UART_int.c", således at modtage-interruptet disables/enables i henhold til parameteren.

Hvis man vælger at enable UART modtage-interruptet, skal vi naturligvis sørge for, at der eksisterer en interrupt service-rutine (ISR), men den skal ikke skrives i selve driveren.

Opret nu et nyt C projekt (kald det LAB12_1) med en main-fil, der hedder "LAB12_1.c".

Tilføj til projektet source-filerne "uart_int.c" og (fra LAB7) "led.c".

Tilføj også header-filerne "uart_int.h" og (fra LAB7) "led.h".



Skriv i "LAB12.c"'s main()-funktion kode, der:

- Initierer UART'en, således at UART modtage-interruptet bliver enabled.
- Foretager global interrupt enable.
- Initierer LED driveren.
- Herefter går i en uendelig sløjfe (laver "ingenting").

Skriv også i "LAB12.c" en interrupt service rutine (ISR) for UART modtage interrupts.

I denne ISR skal følgende ske:

- Aflæs UART'ens modtageregister UDR (= modtaget tegn) til en lokal variabel.
- Hvis det modtagne tegn er '0', '1', '2', '3', '4', '5', '6' eller '7' skal den tilsvarende lysdiode toggles (hvis f.eks. tegnet er '3' skal lysdiode nummer 3 toggles). Derefter sendes der en besked retur til terminalen: "LED nummer x er toggled". "x" er lysdiodens nummer.

Til at modtage tegn må du i denne øvelse ikke anvende **ReadChar()** funktionen, men skal kun anvende modtage-interrupts.

Til at sende tegn må man anvende alle driverens relevante funktioner (f.eks. **SendString()** eller **SendInteger()**).

Test programmet grundigt.

Anvend et terminalprogram på PC'en til at kommunikere med UART'en.

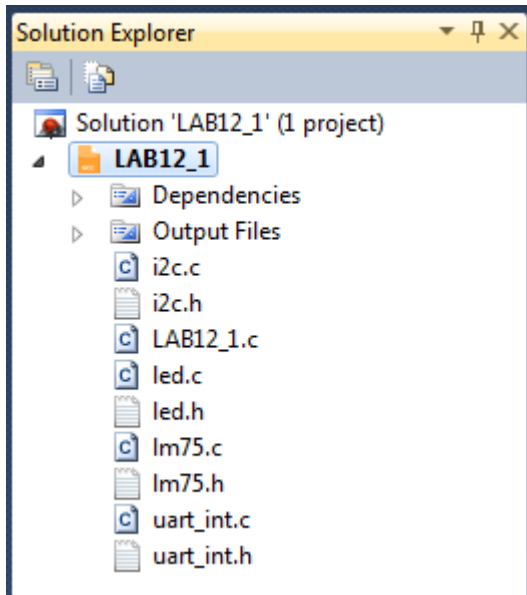
Hvis du bruger Atmel Studios "Terminal vindue", så husk at klikke på LF og CR, så de ikke længere er blå (ellers sendes der også '\r' og '\n', når du trykker på "Send"):



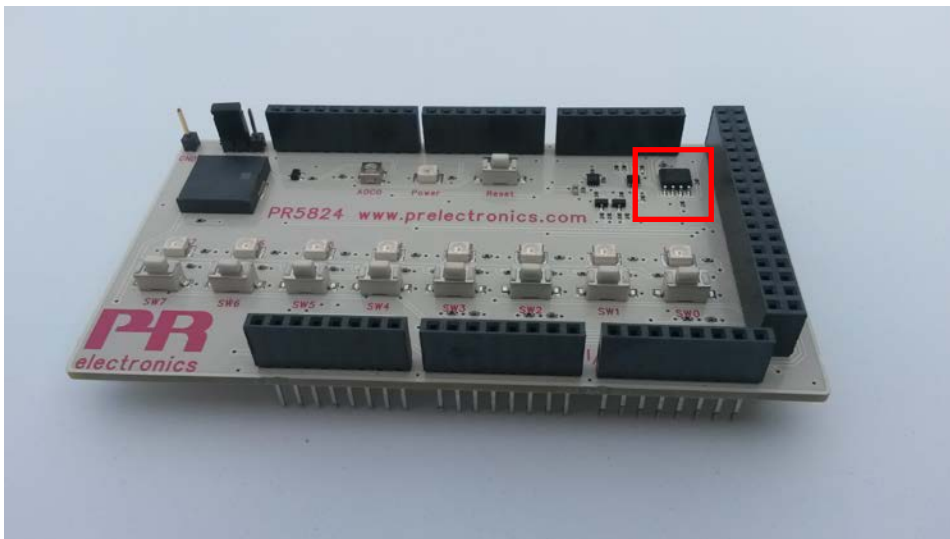
Del 1b: Udvid med temperaturlæsning

Programmet skal nu udvides, sådan at der skrives kode i programmets while(1) – sløjfe.

Da vi også vil anvende en (eksisterende) driver for temperaturføleren LM75, skal vi desuden tilføje "LM72.c", "LM75.h", "I2C.c" og "I2C.h" (alle kan hentes fra MSYS Blackboard):



På "Mega2560 I/O Shield" findes en temperaturføler, som hedder LM75 :



Der kommunikeres med LM75 via en standard grænseflade, som hedder I2C.

Desværre er der i MSYS ikke tid til at behandle I2C, og derfor udleveres driver for både I2C og for LM75.

Her ser vi altså et eksempel på, at vi vil kunne anvende "ukendt" hardware, hvis blot vi får udleveret en driver til denne hardware ☺.

Her ses LM75 – driverens header fil:

```
/*
 * "LM75.h":
 * Header file for LM75 driver.
 * LM75 is an I2C temperature sensor.
 * Temperature is returned in HALFs of
 * centigrades.
 *
 * Henning Hargaard, 13/11 2015
 */
void LM75_init();
int LM75_temperature(unsigned char SensorAddress)
/*
```

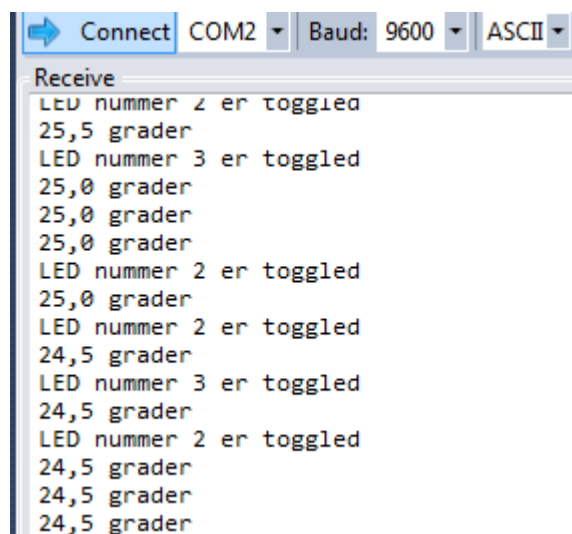
Foretag disse tilføjelser til "LAB12_1.c" :

1. #include "LM75.h"
2. Inden while(1) – sløjfen: Kald LM75_init().
3. I while(1) – sløjfen:
Skriv kode, der kalder LM75_temperature(0) og gemmer det returnerede i en variabel.
Parameteren "SensorAddress" skal altså være 0.
LM75_temperature(0) vil returnere LM75's temperatur i enheden "halve grader celcius".
Hvis temperaturen f.eks. er 24,5 grader, vil heltallet 49 blive returneret.

Udskriv temperaturen til terminalen i formatet "24,5 grader".
Brug hertil relevante funktioner fra UART-driveren.

Hold en pause på 1 sekund (brug _delay_ms(1000)) efter hver aflæsning/udskrivning.

Test programmet og konstater, at lysdioderne kan toggles "samtidig" med at temperaturen udskrives en gang i sekundet:



```
Connect COM2 Baud: 9600 ASCII
Receive
LED nummer 4 er toggled
25,5 grader
LED nummer 3 er toggled
25,0 grader
25,0 grader
25,0 grader
LED nummer 2 er toggled
25,0 grader
LED nummer 2 er toggled
24,5 grader
LED nummer 3 er toggled
24,5 grader
LED nummer 2 er toggled
24,5 grader
24,5 grader
24,5 grader
```

Temperaturføleren kan (f.eks.) opvarmes med en finger eller ved at ånde på den.

Del 2: Software UART transmitter

I nogle situationer har man ikke en hardware UART til rådighed, og man kan i den situation vælge at implementere en "software UART".

At implementere en UART receiver i ren software er mulig, men dog mere kompliceret end at implementere en UART transmitter i software.

En software UART receiver implementeres bedst ved brug af et eksternt interruptben samt en timer. Interruptbenet opsættes til at generere et interrupt på forkanten af det modtagne start bit, og timeren anvendes til at sample de modtagne data bit på de rette tidspunkter.

En mere primitiv metode anvender stadig interruptbenet til at detektere start bit, men et software delay anvendes til at styre timingen i forbindelse med data bit samplingen.

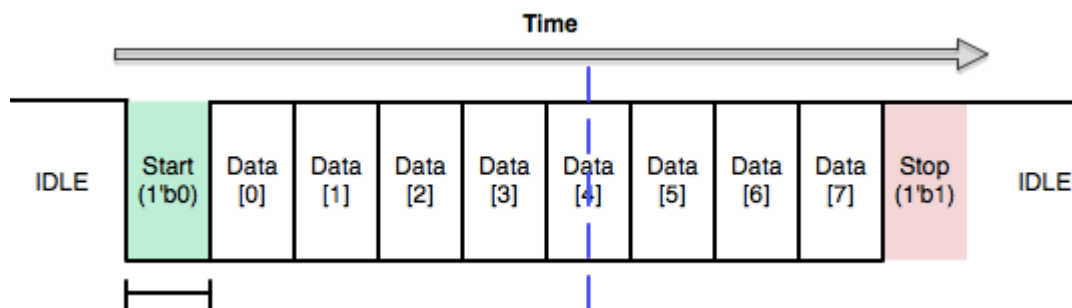
En software UART transmitter kan implementeres meget simpelt. Eneste ressource er et vilkårligt portben, der anvendes til at transmittere signalet ("TX" benet).

Den simpleste måde at styre længden af de enkelte bit er ved at anvende et software delay.

Opgaven:

Skriv og test en funktion `void SendCharSW(char Tegn)`, der implementer afsendelse af tegnet "Tegn" som beskrevet ovenfor. Anvend 8 databit, "ingen paritet" og 1 stopbit. Baud rate = 9600.

En skabelon for programmet kan hentes fra MSYS Blackboard (hedder LAB12_2).



På "Arduino Mega2560" er "PC terminalen" via USB forbundet til UART 0. Derfor bør det "tilfældige TX ben" være benet TX0 = Ben 1 på PORTE.

Vigtigt: "Arduino Mega2560" anvender UART'en, når man downloader programmet til microcontrolleren, MEN "glemmer" at disable UART'en igen. Derfor er det vigtigt at starte med at sætte bit RXEN og TXEN lave. Det kan gøres via denne statement: `UCSR0B = 0;`

Test kan igen foretages via en tilkoblet PC med terminalprogram.

Tilslut også et oscilloscop (f.eks. "Analog Discovery") til TX0-benet og konstater, om timingen er i orden. Ved meget høje baud rates vil vi sandsynligvis opleve, at timingen bliver "upræcis". Forklar dette fænomen.

Ekstra opgave (for "nørden", der evt. har ekstra tid eller bare ikke kan lade være):
Implementer en software UART receiver efter retningslinierne ovenfor.