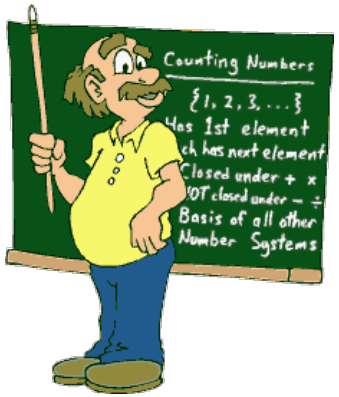
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

MSYS

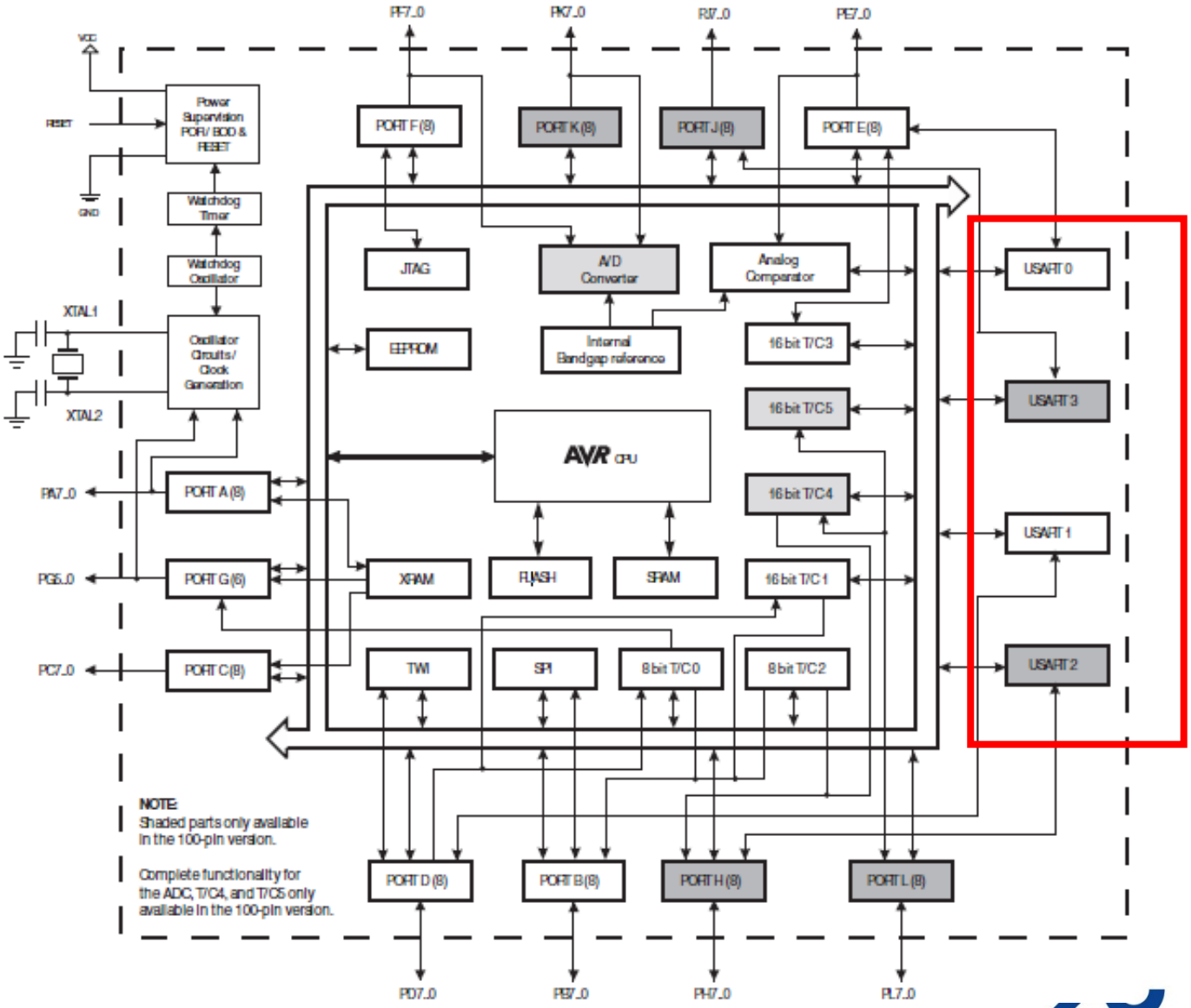
Microcontroller Systems

Lektion 18: UART interrupts m.m.



Version: 13-11-2017, Henning Hargaard

Mega2560: 4 USART'er

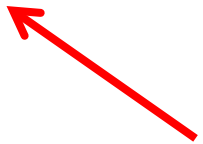


AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: USART pins

USART0 :
TXD0 = PE, ben 1
RXD0 = PE, ben 0

USART1 :
TXD1 = PD, ben 3
RXD1 = PD, ben 2

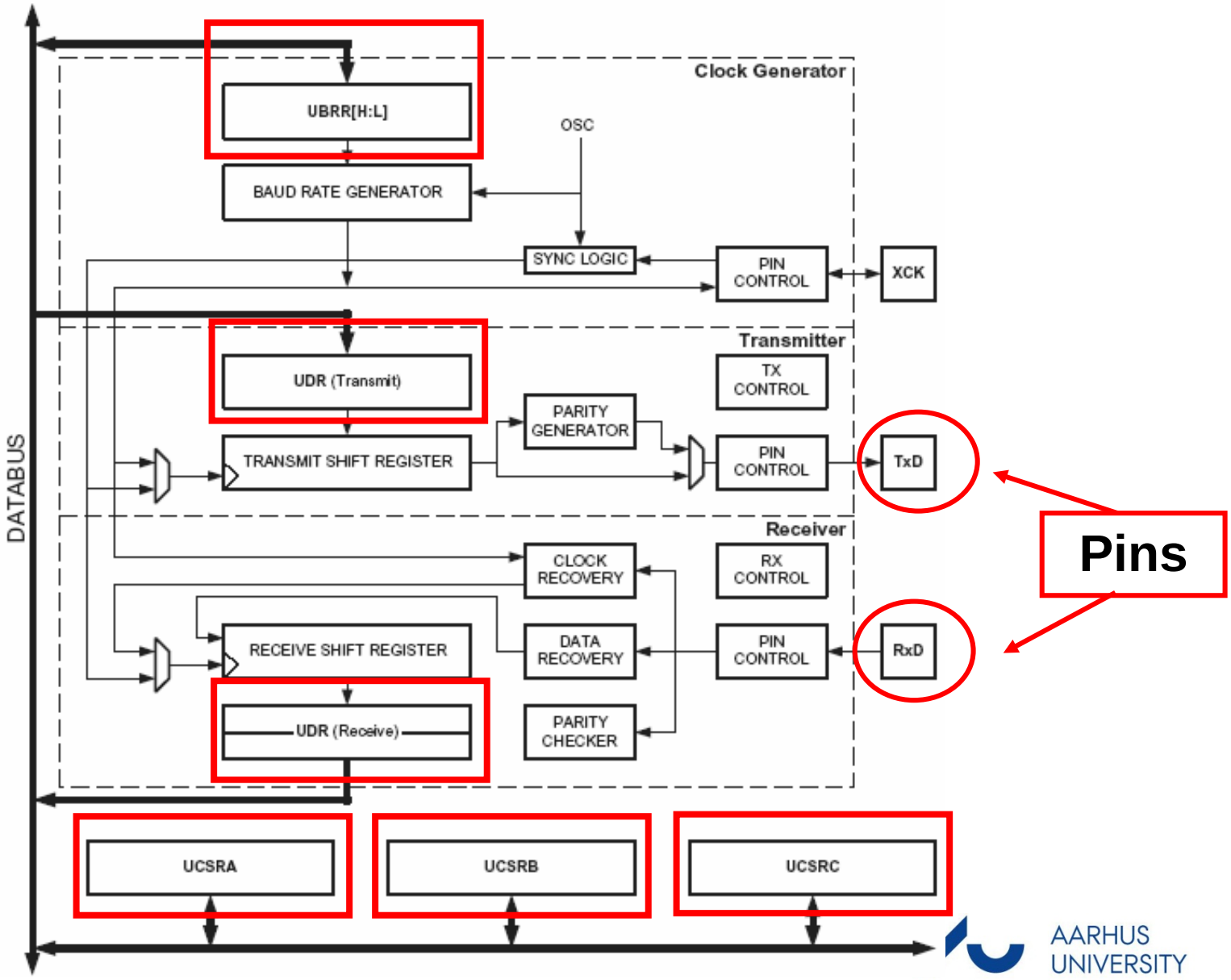


”Arduino Mega2560”
USB - stikket

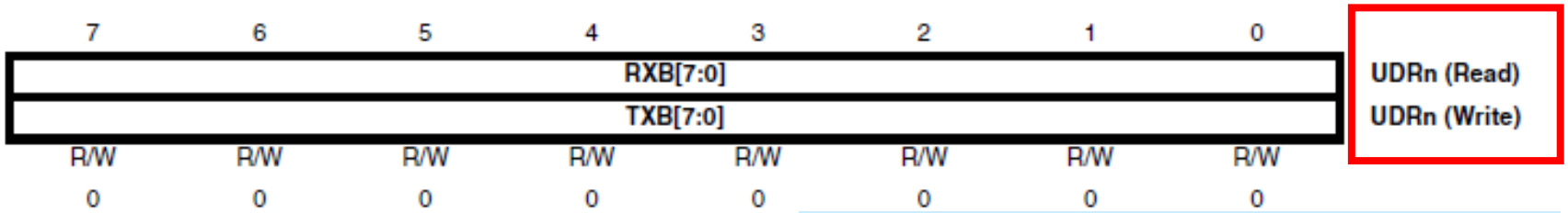
USART2 :
TXD2 = PH, ben 1
RXD2 = PH, ben 0

USART3 :
TXD3 = PJ, ben 1
RXD3 = PJ, ben 0

Mega32/Mega2560 USART



Mega2560: UDRn. Usart Data Registers



UDR0, UDR1, UDR2 eller UDR3

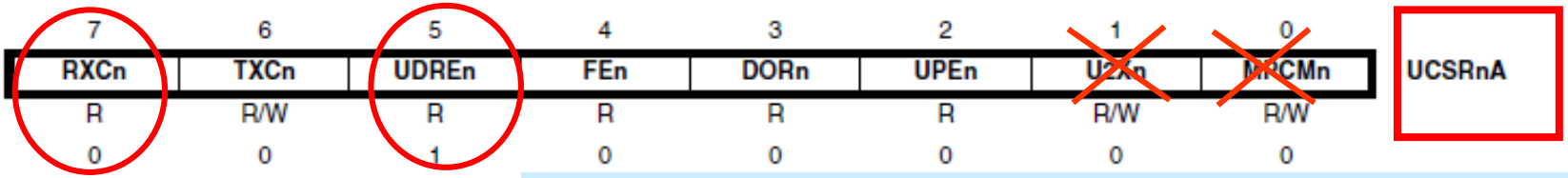
Bemærk: Fysisk to forskellige registre med samme navn (UDRn)

```
char x;
```

Læsning af modtaget tegn : **x = UDR0;**

Send tegn : **UDR0 = x;**

Mega2560: UCSRnA: Control and Status Registers A



UCSR0A, UCSR1A, UCSR2A eller UCSR3A

Når RXCn er 1: Nyt tegn modtaget (”kan hentes i UDRn”).

Når TXC er 1: ”Sender tom” (klar til at sende nyt tegn og sendeskifteregisteret tomt).

Når UDREn er 1: Klar til at sende nyt tegn (”der må skrives til UDRn”).

FEn : ”Framing Error” (modtaget tegn har fejl i stop bit).

DORn : ”Data overrun” (tegn modtaget, inden foregående er blevet læst af SW).

PEn : ”Parity Error” (modtaget tegn har paritetsfejl).

Test (”socrative.com”, Room = MSYS)

- Hvilken metode er korrekt for at vente på, at et nyt tegn modtages (når der ikke anvendes interrupt) ?
- A:

```
if (UCSR0A & 0b10000000) != 0)
return UDR0;
```

- B:

```
while (UCSR0A & 0b10000000 == 0)
{ }
return UDR0;
```

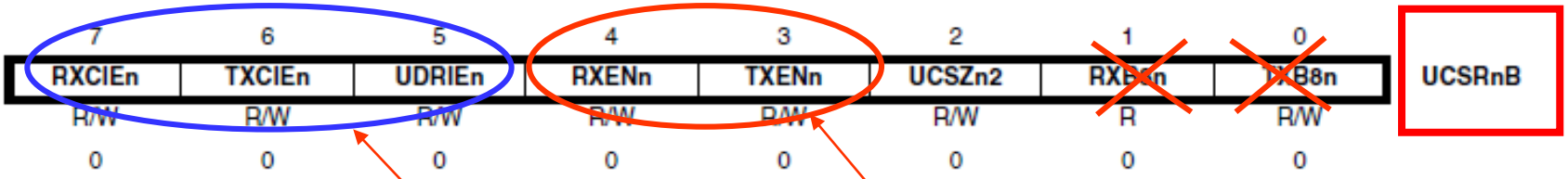
- C:

```
while (UCSR0A & 0b10000000 != 0)
{ }
return UDR0;
```

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: UCSRnB. Control and Status Register B

UCSR0B, UCSR1B, UCSR2B eller UCSR3B

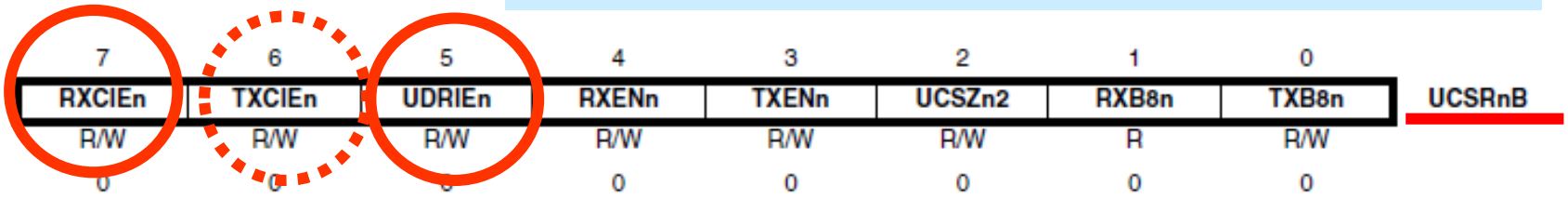


Bruges ved interrupt -styret USART

RXENn = 1: ”RX Enable” (tænder for modtageren).
TXENn = 1: ”TX Enable” (tænder for senderen).

Mega2560: USART interrupt enables

UCSR0B, UCSR1B, UCSR2B eller UCSR3B



RXCIEn = 1: USART Rx Interrupt Enable.

Interrupt hver gang et nyt tegn er modtaget.
Tegnet vil ligge klar i UDR, når vi får interruptet.

(**TXCIEn = 1:** USART Tx Interrupt Enable).

Interrupt hver gang sende-skifteregisteret bliver tomt og UDR klar til at sende et nyt tegn.
Betyder, at man må sende næste tegn (til UDR).
(Oftest benyttes det andet sender-interrupt i stedet).

UDRIEn = 1: USART Data Register Empty Interrupt Enable.

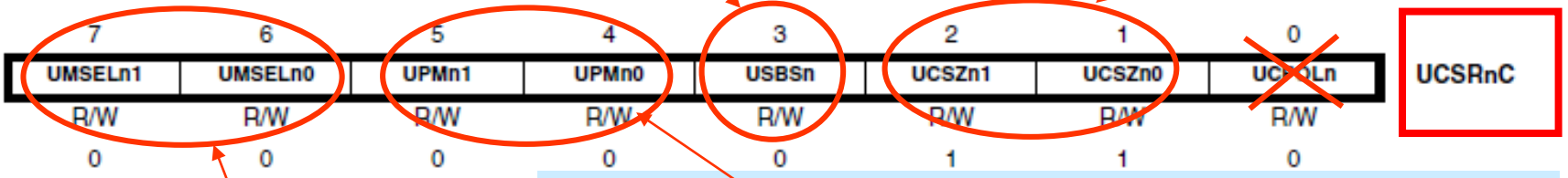
Interrupt hver gang UDR er klar til at sende et tegn.
Betyder, at man kan sende næste tegn (til UDR).

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: UCSRnC. Control and Status Register C

UCSZn1 og UCSZn0 : Antal data bits (se næste side).

USBSn : 0 = 1 stop bit. 1 = 2 stop bits.



UCSR0C, UCSR1C, UCSR2C eller UCSR3C

UMSELn1 = 0 og UMSELn0 = 0 : Asynkron mode !

UPMn1 og UPMn0: Valg af paritet:

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

Test (”socrative.com”, Room = MSYS)

- Vi antager, at der benyttes 8 databit og EVEN parity. Hvilken bitstrøm har paritetsfejl ?

A:

Data = 11101001 Paritetsbit = 1

B:

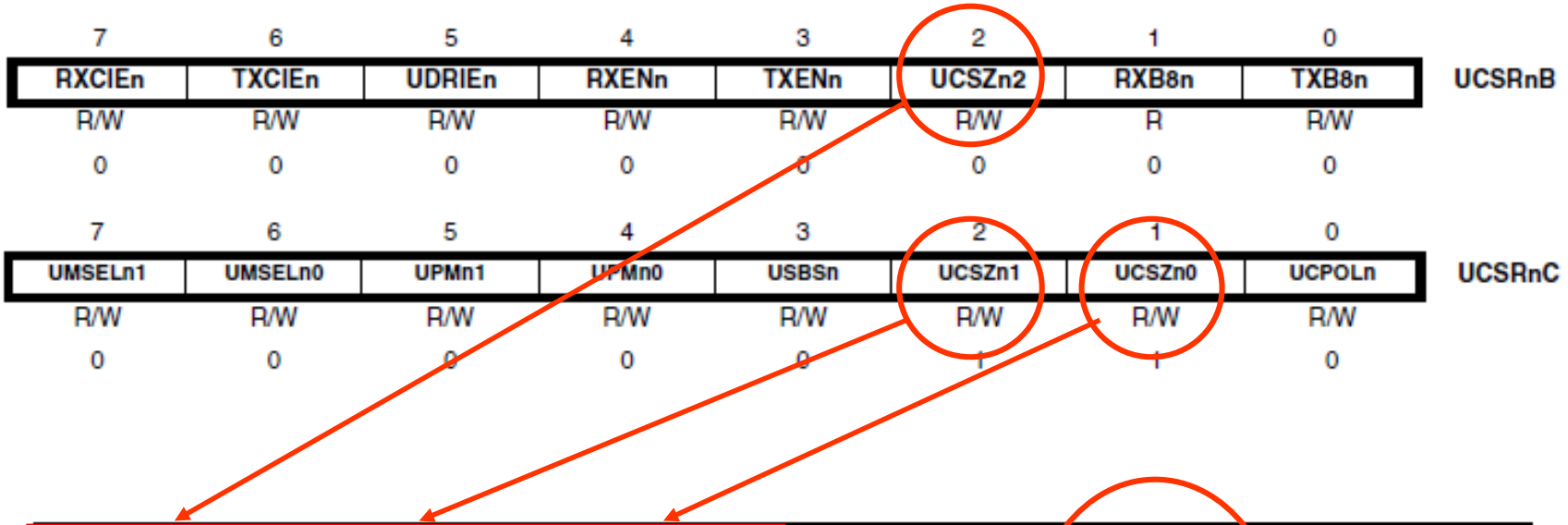
Data = 10001011 Paritetsbit = 0

C:

Data = 10001011 Paritetsbit = 1

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: Antal data bits



7	6	5	4	3	2	1	0	
RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
0	0	0	0	0	0	0	0	

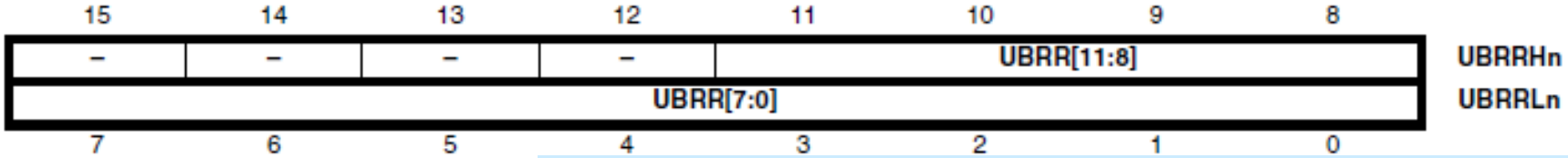
7	6	5	4	3	2	1	0	
UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	UCSRnC
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0	0	0	0	0	1	1	0	

UCSZn2	UCSZn1	UCSZn0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega2560: UBRRHn + UBRRLn. Baud Rate Registre

UBRRH0, UBRRH1, UBRRH2 eller UBRRH3



UBRRL0, UBRRL1, UBRRL2 eller UBRRL3

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2Xn = 0)	$BAUD = \frac{f_{OSC}}{16(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed mode (U2Xn = 1)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$

”UBRR” = (256 * UBRRH) + UBRRL

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Test (”socrative.com”, Room = MSYS)

- CPU clock frekvens = 3,6864 MHz
  UBRRH = 2
  UBRRL = 255
  Hvilken BAUD rate anvendes ?

  A:
  300 bit/s
  B:
  1200 bit/s
  C:
  9600 bit/s
- D:
  115200 bit/s

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Mega32: USART interrupt vektorer

Table 10-1: Interrupt Vector Table for the Mega32

Interrupt	ROM Location (Hex)
Reset	0000
External Interrupt request 0	0002
External Interrupt request 1	0004
External Interrupt request 2	0006
Time/Counter2 Compare Match	0008
Time/Counter2 Overflow	000A
Time/Counter1 Capture Event	000C
Time/Counter1 Compare Match A	000E
Time/Counter1 Compare Match B	0010
Time/Counter1 Overflow	0012
Time/Counter0 Compare Match	0014
Time/Counter0 Overflow	0016
SPI Transfer complete	0018
USART, Receive complete	001A
USART, Data Register Empty	001C
USART, Transmit Complete	001E
ADC Conversion complete	0020
EEPROM ready	0022
Analog Comparator	0024
Two-wire Serial Interface	0026
Store Program Memory Ready	0028

Mega2560: USART Interrupt vektorer (1 af 2)

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016(3)	PCINT2	Pin Change Interrupt Request 2
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	$001E	TIMER2 OVF	Timer/Counter2 Overflow
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI, STC	SPI Serial Transfer Complete
26	$0032	USART0 RX	USART0 Rx Complete
27	$0034	USART0 UDRE	USART0 Data Register Empty
28	$0036	USART0 TX	USART0 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator
30	$003A	ADC	ADC Conversion Complete

Mega2560: USART Interrupt vektorer (2 af 2)

Vector No.	Program Address[2]	Source	Interrupt Definition
31	$003C	EE READY	EEPROM Ready
32	$003E	TIMER3 CAPT	Timer/Counter3 Capture Event
33	$0040	TIMER3 COMPA	Timer/Counter3 Compare Match A
34	$0042	TIMER3 COMPB	Timer/Counter3 Compare Match B
35	$0044	TIMER3 COMPC	Timer/Counter3 Compare Match C
36	$0046	TIMER3 OVF	Timer/Counter3 Overflow
37	$0048	USART1 RX	USART1 Rx Complete
38	$004A	USART1 UDRE	USART1 Data Register Empty
39	$004C	USART1 TX	USART1 Tx Complete
40	$004E	TWI	2-wire Serial Interface
41	$0050	SPM READY	Store Program Memory Ready
42	$0052[3]	TIMER4 CAPT	Timer/Counter4 Capture Event
43	$0054	TIMER4 COMPA	Timer/Counter4 Compare Match A
44	$0056	TIMER4 COMPB	Timer/Counter4 Compare Match B
45	$0058	TIMER4 COMPC	Timer/Counter4 Compare Match C
46	$005A	TIMER4 OVF	Timer/Counter4 Overflow
47	$005C[3]	TIMER5 CAPT	Timer/Counter5 Capture Event
48	$005E	TIMER5 COMPA	Timer/Counter5 Compare Match A
49	$0060	TIMER5 COMPB	Timer/Counter5 Compare Match B
50	$0062	TIMER5 COMPC	Timer/Counter5 Compare Match C
51	$0064	TIMER5 OVF	Timer/Counter5 Overflow
52	$0066[3]	USART2 RX	USART2 Rx Complete
53	$0068[3]	USART2 UDRE	USART2 Data Register Empty
54	$006A[3]	USART2 TX	USART2 Tx Complete
55	$006C[3]	USART3 RX	USART3 Rx Complete
56	$006E[3]	USART3 UDRE	USART3 Data Register Empty
57	$0070[3]	USART3 TX	USART3 Tx Complete

Mega2560: USART 0. ISR navne

```
ISR (USART0_RX_vect)
{
}
```

```
ISR (USART0_TX_vect)
{
}
```

```
ISR (USART0_UDRE_vect)
{
}
```

Mega2560 ISR navne (USART 0 - 3)

USART 0 interrupts	ISR name
RX	USART0_RX_vect
TX	USART0_TX_vect
UDR Empty	USART0_UDRE_vect

USART 2 interrupts	ISR name
RX	USART2_RX_vect
TX	USART2_TX_vect
UDR Empty	USART2_UDRE_vect

USART 1 interrupts	ISR name
RX	USART1_RX_vect
TX	USART1_TX_vect
UDR Empty	USART1_UDRE_vect

USART 3 interrupts	ISR name
RX	USART3_RX_vect
TX	USART3_TX_vect
UDR Empty	USART3_UDRE_vect

Metode for interruptstyret UART modtager

```
ISR (USART0_RX_vect)
{
char x;
   // Hent modtaget tegn
   x = UDR0;
   // - og brug så x til "noget"
   .........
}
```

Bemærk, at vi ikke behøver at ”vente på, at der modtages et nyt tegn”.

Vi får interruptet, fordi et tegn allerede er modtaget !

LAB11: UART driver

```
/*****************************************
* "uart.h":                              *
* Header file for Mega2560 UART driver.  *
* Using UART 0.                          *
* Henning Hargaard, 11/11 2015           *
*****************************************/
void InitUART(unsigned long BaudRate, unsigned char DataBit, char Parity);
unsigned char CharReady();
char ReadChar();
void SendChar(char Tegn);
void SendString(char* Streng);
void SendInteger(int Tal);
/*****************************************/
```

Driverens header fil.

InitUART()

void InitUART(unsigned long BaudRate, unsigned char DataBit, char Parity)
Skal initiere UART 0 til den ønskede BAUD-rate (300 - 115200), det ønskede antal databits (5 - 8) og ønsket paritet ('E' = Even Parity, 'O' = Odd Parity – ellers No Parity).

Hvis Parameteren BaudRate er mindre end 300 eller større end 115200, må der ikke ske nogen initiering af UART'en.
Hvis Parameteren DataBit er mindre end 5 eller større end 8, må der ikke ske nogen initiering af UART'en.

Vi antager, at Mega2560's clockfrekvens er 16 MHz.
Den værdi, der skal skrives til UBRRH og UBRRL, skal i funktionen beregnes på basis af "BaudRate"- parameteren og Mega2560's CPU clockfrekvens (bemærk, at afrunding kan forekomme).

Desuden skal UART'en initieres til:

- Asynkron mode.
- Både RX og TX enabled.
- 1 stop bit.
- Alle interrupts disabled.

Løsning: InitUART()

```
// Constants
#define XTAL 16000000

void InitUART(unsigned long BaudRate, unsigned char DataBit, char Parity)
{
  if ((BaudRate >= 300) && (BaudRate <= 115200) && (DataBit >=5) && (DataBit <= 8))
  {
    // "Normal" clock, no multiprocessor mode (= default)
    UCSR0A = 0b00100000;
    // No interrupts enabled
    // Receiver enabled
    // Transmitter enabled
    // No 9 bit operation
    UCSR0B = 0b00011000;
    // Asynchronous operation, 1 stop bit
    // Bit 2 and bit 1 controls the number of data bits
    UCSR0C = (DataBit-5)<<1;
    // Set parity bits (if parity used)
    if (Parity == 'E')
      UCSR0C |= 0b00100000;
    else if (Parity == 'O')
      UCSR0C |= 0b00110000;
    // Set Baud Rate according to the parameter BaudRate:
    UBRR0 = XTAL/(16*BaudRate) - 1;
  }
}
```

Karakter funktioner

unsigned char CharReady( )
Meddeler, om UART 0 har modtaget et tegn.
Hvis et tegn er modtaget, returneres en værdi forskellig fra 0 (= TRUE).
Hvis der ikke er modtaget et tegn, returneres værdien 0 (= FALSE).
Funktionen skal ***ikke*** afvente modtagelse af et tegn, men blot returnere oplysningen om, hvorvidt et tegn er modtaget.

char ReadChar( )
Returnerer et modtaget tegn fra UART 0's modtageregister (UDR).
Funktionen ***skal først afvente, at et tegn modtages*** (bit RXC0 i registeret UCSRA0).
Derefter skal tegnet i UDR0 returneres.

void SendChar(char Tegn)
Sender et tegn via UART 0. Tegnets overføres som parameter.
Inden tegnet skrives til data registeret (UDR0), skal funktionen afvente "UART data register empty" (bit UDRE0 i registeret UCSRA0).

CharReady(), ReadChar() og SendChar()

```
unsigned char CharReady()
{
   return UCSR0A & (1<<7);
}

char ReadChar()
{
  // Wait for new character received
  while ( (UCSR0A & (1<<7)) == 0 )
  {}
  // Then return it
  return UDR0;
}

void SendChar(char Tegn)
{
  // Wait for transmitter register empty (ready for new character)
  while ( (UCSR0A & (1<<5)) == 0 )
  {}
  // Then send the character
  UDR0 = Tegn;
}
```

SendString()

void SendString(char* Streng)
Udskriver en 0-termineret tekststreng ved hjælp af UART 0.
Funktion modtager som parameter en pointer til den streng, som vi ønsker udskrevet.
Pointeren peger altid på det første tegn i strengen, som altså er 0-termineret.
Brugeren har på forhånd (altså inden denne funktion kaldes) oprettet og lagret strengen.

Nedenstående viser i pseudo-kode, hvordan funktionen kan implementeres:

```
while (”Det som pointeren peger på” ikke er 0)
{
   SendChar(”Det som pointeren peger på”);
   Flyt pointeren en plads frem;
}
```

SendString()

```
void SendString(char* Streng)
{
  // Repeat until zero-termination
  while (*Streng != 0)
  {
    // Send the character pointed to by "Streng"
    SendChar(*Streng);
    // Advance the pointer one step
    Streng++;
  }
}
```

SendInteger()

void SendInteger(int Tal)
Denne funktion skal udskrive værdien af integer "Tal", der modtages som parameter.

Hvis man f.eks. kalder funktionen på følgende måde:

SendInteger(147);

skal følgende tegn sendes via UART 0: '1', '4' og '7'.

Hint:
Opret først i funktionen et lokalt array af "passende" størrelse.
Brug dernæst standard-funktionen **itoa( )** til at konvertere "Tal" til en streng, der gemmes i dette array. Husk **#include <stdlib.h>.**
itoa(tal, array, 10) gemmer strengen svarende til "tal" i "array" (og 0-terminerer denne).
Brug derefter funktionen **SendString()** til at sende strengen.

Tekst-strenge

	'V'	'i'	'g'	'g'	'o'	0
Index:	**0**	**1**	**2**	**3**	**4**	**5**

- En tekststreng (string) er et char array, der indeholder tekst. Hvert element indeholder altså et tegn.
- Det sidste element skal være en NUL-terminering (værdien 0). Det fortæller os, hvor strengen slutter.
- I compileren kan vi skrive en streng direkte ved brug af anførselstegn: "…".
  Compileren putter automatisk NUL-termineringen på !

```
char str[6] = "Viggo";
```

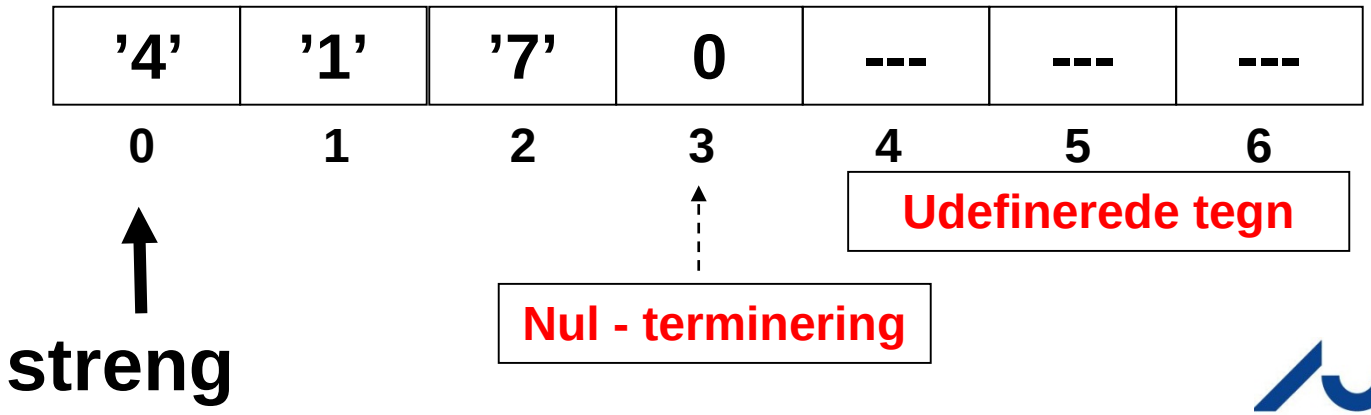
itoa(int n, char* str, int radix)

Hvis man inkluderer <stdlib.h>, har man adgang til en lang række funktioner, hvoraf itoa() er en af de vigtigste.

itoa(int n, char* str, int radix) vil skrive n's værdi som en 0-termineret tekst i et array, som pointeren str peger på. Radix sættes normalt til 10.

Eksempel:

```
char streng[7];
itoa( 417, streng, 10 );   // *str er en pointer til at char array
```



SendInteger()

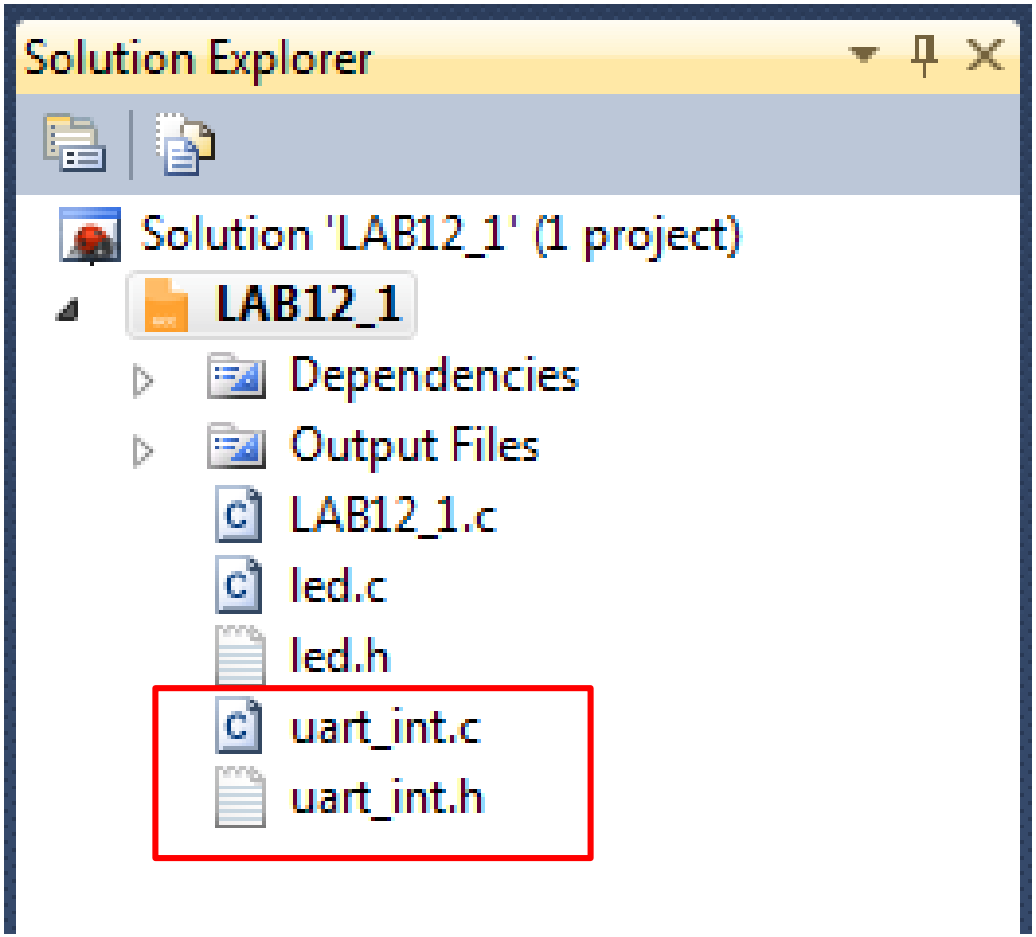
```
void SendInteger(int Tal)
{
char array[7];
  // Convert the integer to an ASCII string (array), radix = 10
  itoa(Tal, array, 10);
  // - then send the string
  SendString(array);
}
```

LAB12: Mulighed for UART interrupt

```
/*****************************************
* "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);
/*****************************************/
```

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.

LAB12, del 1a


Solution Explorer
Solution 'LAB12_1' (1 project)
LAB12_1
Dependencies
Output Files
LAB12_1.c
led.c
led.h
uart_int.c
uart_int.h

LAB12, del 1a

Skriv i ”LAB12.c”’s main()-funktion kode, der:

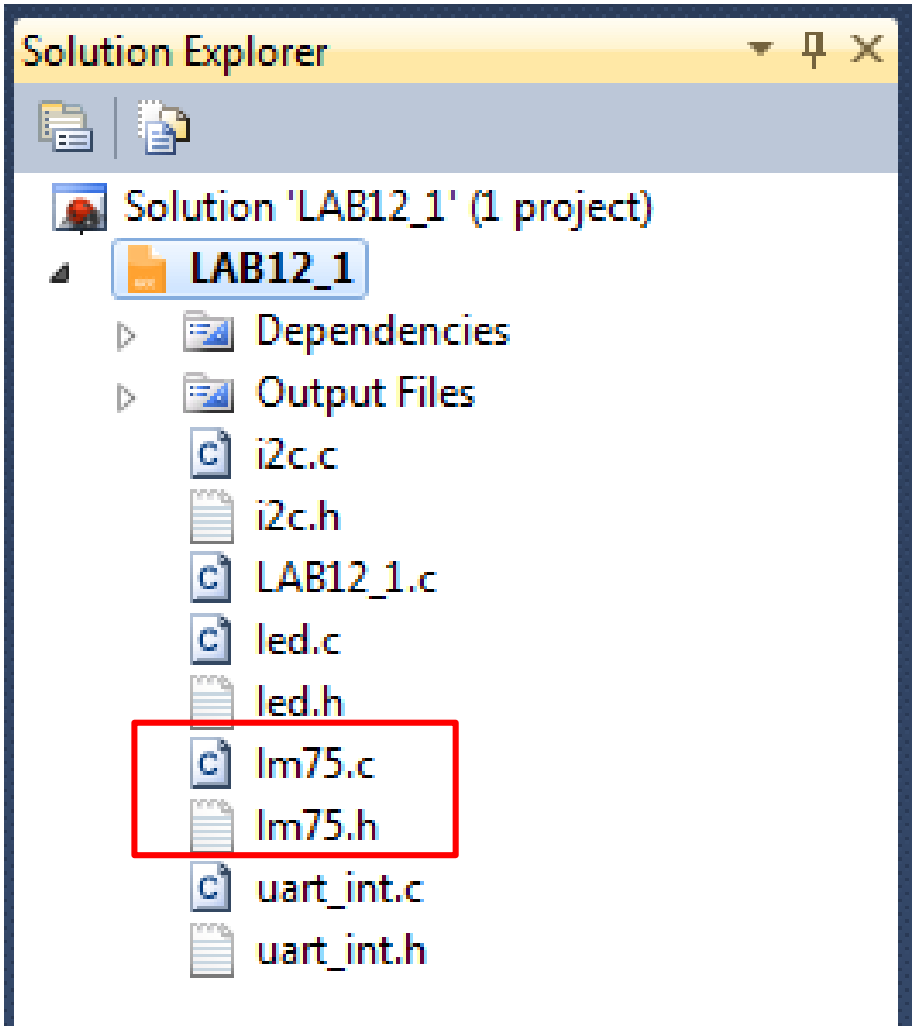
- Initierer UART’en, således at UART modtage-interruptet bliver enabled.
- Foretager global interrupt enable.
- Initierer LED driveren.
- Herefter går i en uendelig sløjfe (laver ”ingenting”).

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.

LAB12, del 1b


Solution Explorer
Solution 'LAB12_1' (1 project)
LAB12_1
Dependencies
Output Files
i2c.c
i2c.h
LAB12_1.c
led.c
led.h
lm75.c
lm75.h
uart_int.c
uart_int.h

LAB12, del 1b


PR5824 www.prelectronics.com
SW7
SW6
SW5
SW4
SW3
SW2
SW1
SW0
PR
electronics

LAB12, del 1b

```
/*************************************
* "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)
/*************************************/
```

LAB12, del 1b

Foretag disse tilføjelser til ”LAB12_1.c” :

1. #include ”LM75.h”

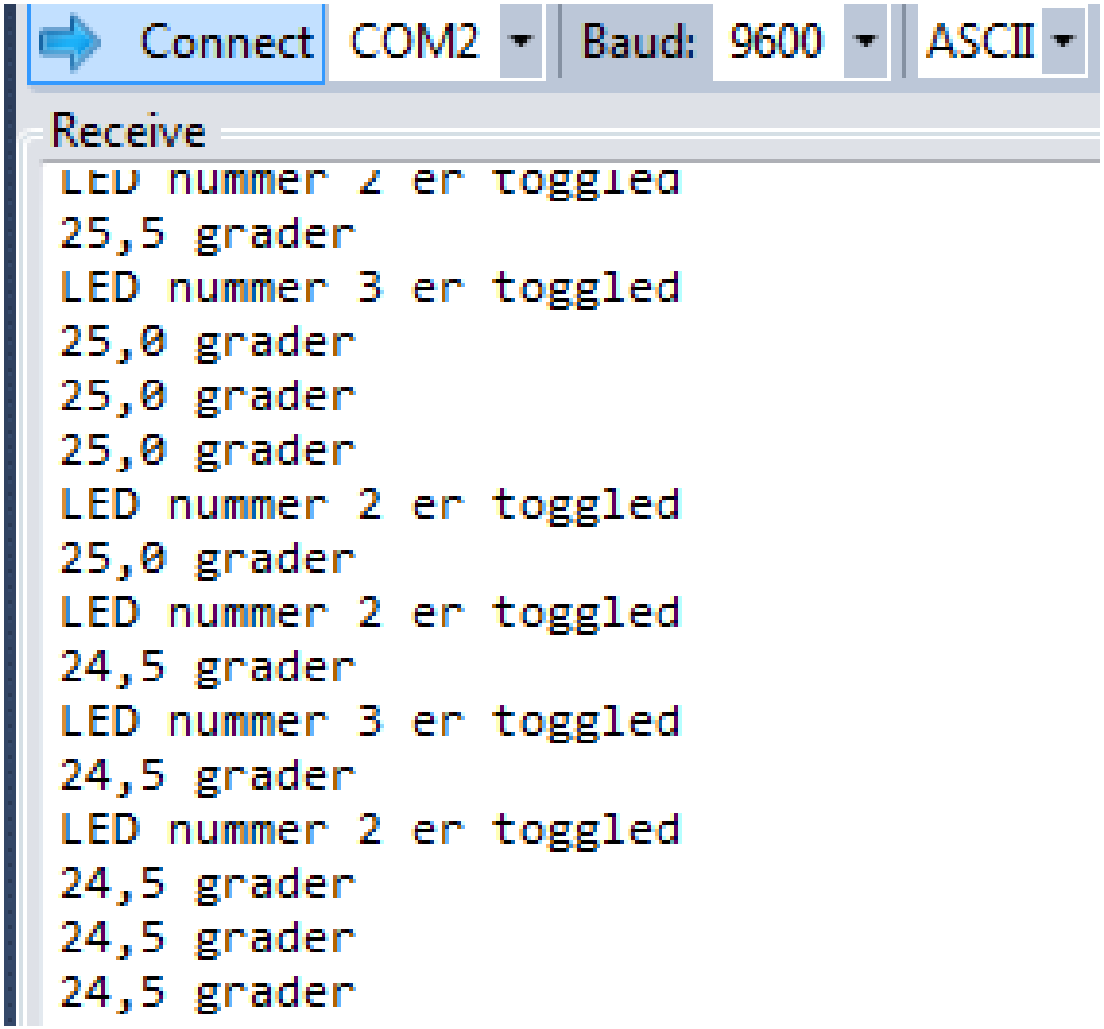
2. Inden while(1) – sløjfen: Kald LM75_init( ).

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.

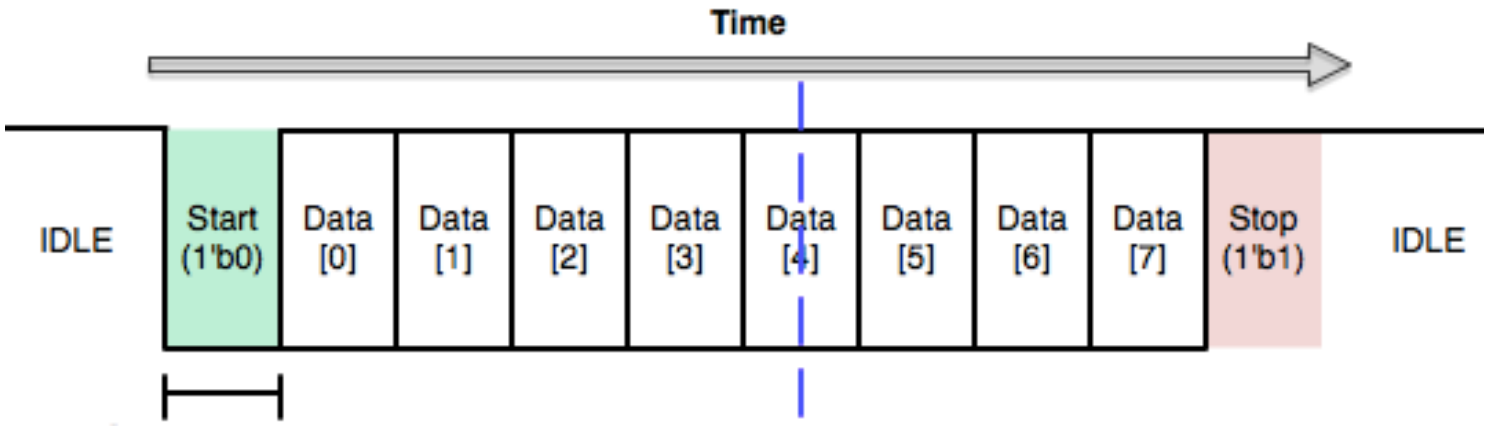
   Udskriv temperaturen til terminalen i formatet ”24,5 grader”.
   Brug hertil relevante funktioner fra UART-driveren.

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

LAB12, del 1b


Connect
COM2
Baud:
9600
ASCII
Receive
LED nummer 2 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

LAB12, del 2 (SW sender)

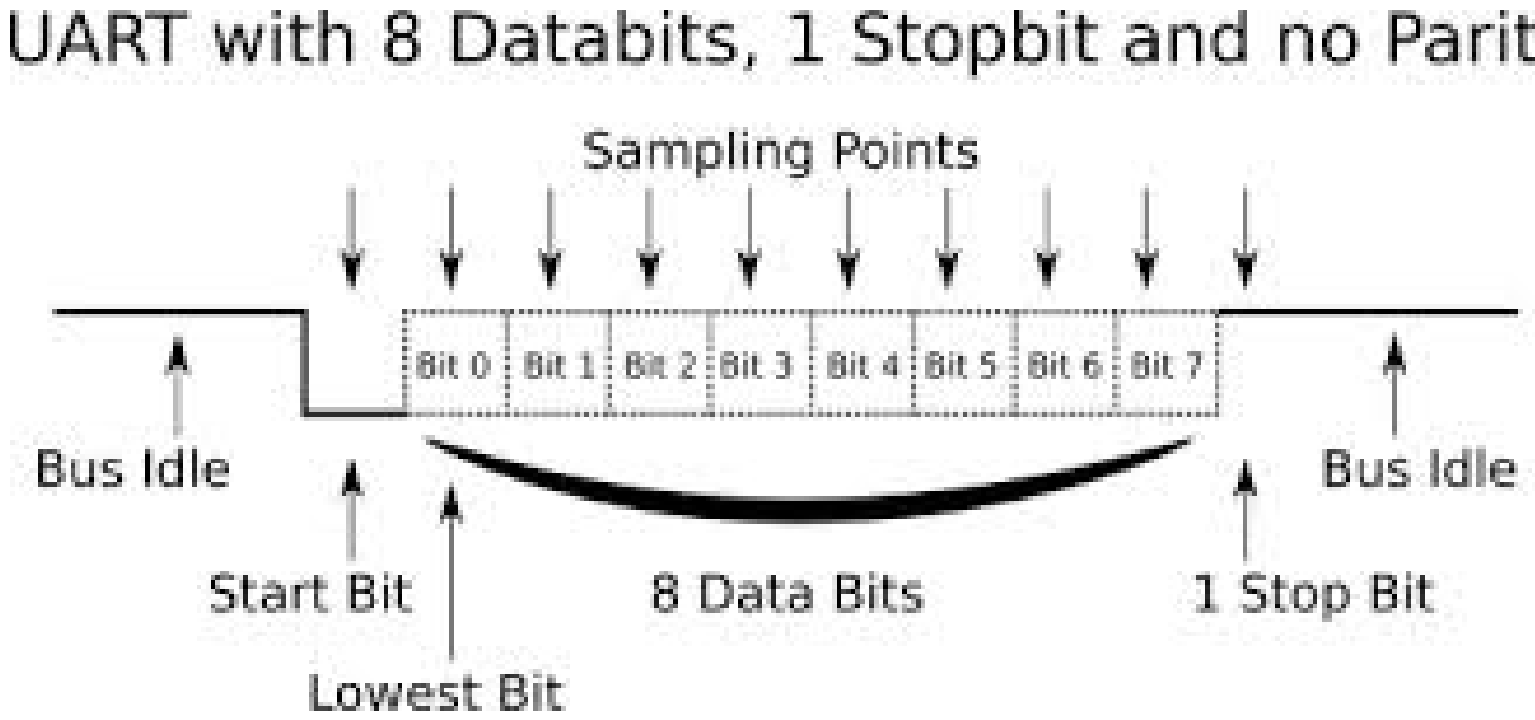


Skriv og test en funktion **void SendCharSW(char Tegn)** , der implementer afsendelse af tegnet "Tegn" som beskrevet ovenfor. Anvend 8 databit,"ingen paritet" og 1 stopbit. Baud rate = 9600.

LAB12, del 2: Skabelon

```
/*********************************************
* MSYS, LAB 12, Del 2                        *
* Software UART sender.                      *
*                                            *
* Henning Hargaard 20/11 2015                *
*********************************************/
#include <avr/io.h>
#define F_CPU 16000000
#include <util/delay.h>

#define DDR   DDRE
#define PORT PORTE
#define PINNR 1
#define BAUD  9600

#define NO_us 1000000/BAUD

// 8 data bit, no parity, 1 stop bit
void SendCharSW(char Tegn)
{
  //  <------------- Skriv kode her
}

int main(void)
{
    UCSR0B = 0;
    DDR |= (1<<PINNR);
    while(1)
    {
        SendCharSW('A');
        SendCharSW('B');
        SendCharSW('C');
    }
}
```

Ekstra i LAB12: SW UART receiver ?



Slut på lektion 18