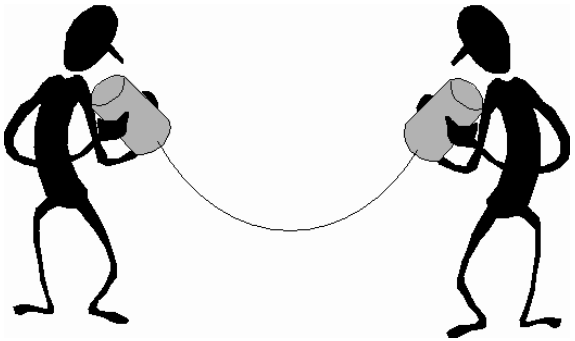
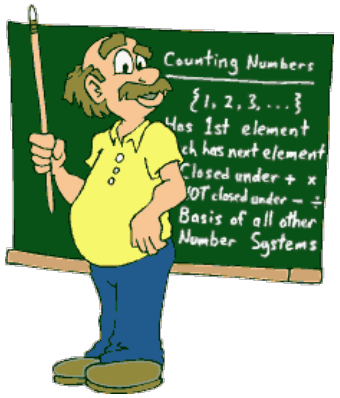
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

MSYS

Microcontroller Systems

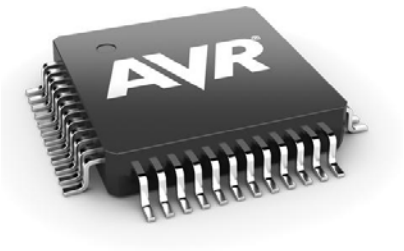
Lektion 17: Seriel kommunikation





Version: 9-4-2018, Henning Hargaard

Seriel communication



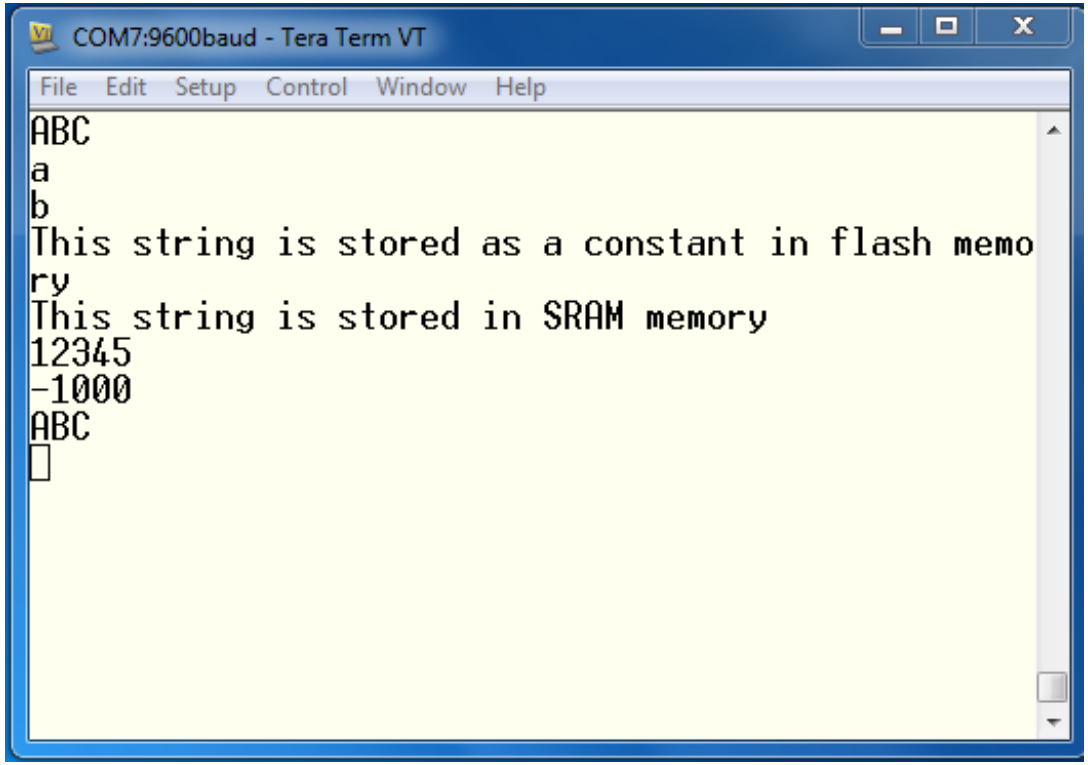
Serielt kabel

Microcontroller:
UART

PC:
UART

- Til udveksling af data mellem 2 systemer (f.eks. mellem en PC og en microcontroller) anvendes ofte seriel kommunikation.
- Hvert system må have en HW-enhed, der kaldes en UART.
- Mega32 har 1 indbygget UART.
- Mega2560 har 4 indbyggede UART'er.

PC: ”Tera Terminal”



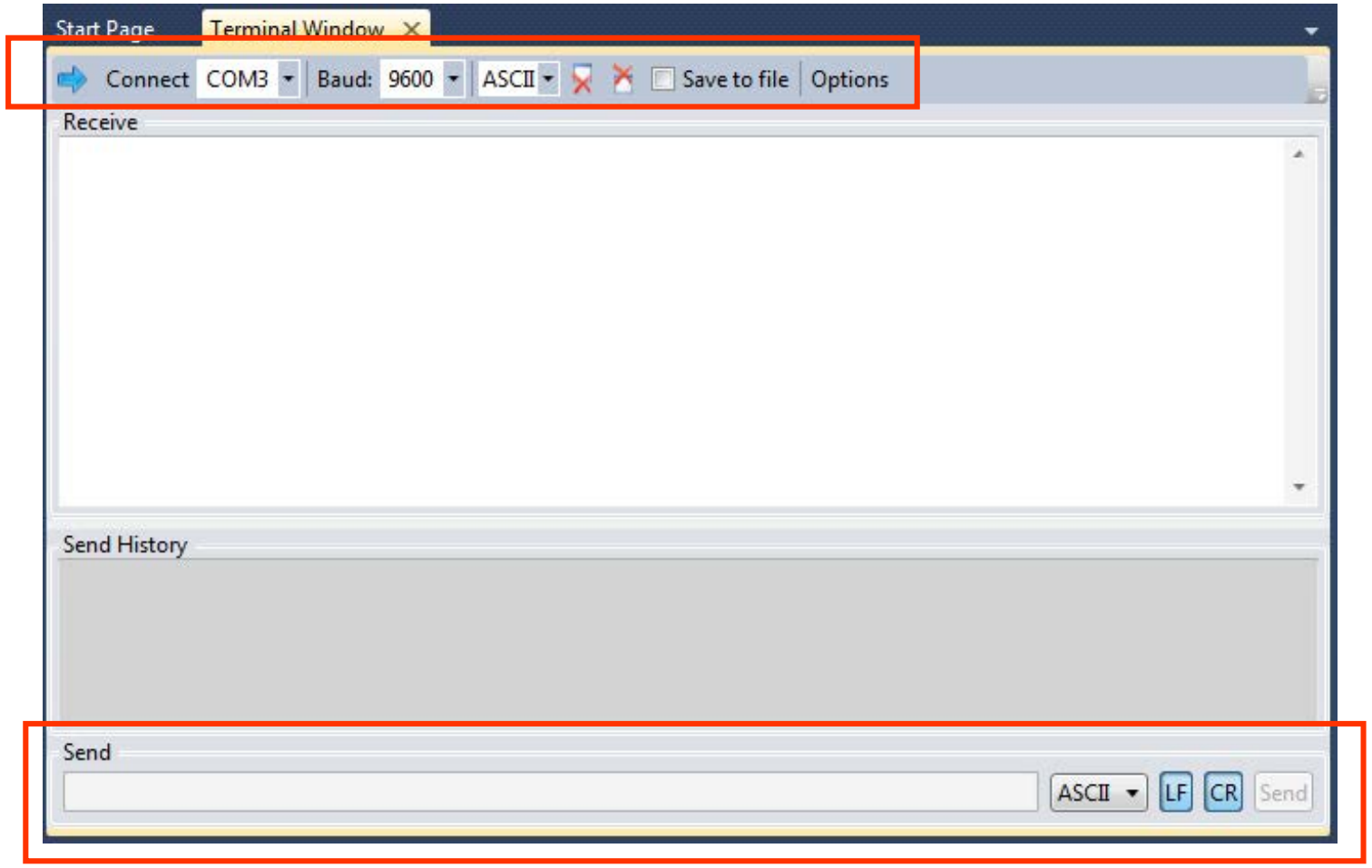
På PC’en startes et terminalprogram (f.eks. det gratis program ”Tera Terminal”).

Når vi trykker en tast på PC tastaturet, sendes tegnets ASCII-kode ud på COM-porten.

Hvis PC’ens COM-port modtager et tegn, vises det i vinduet.

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

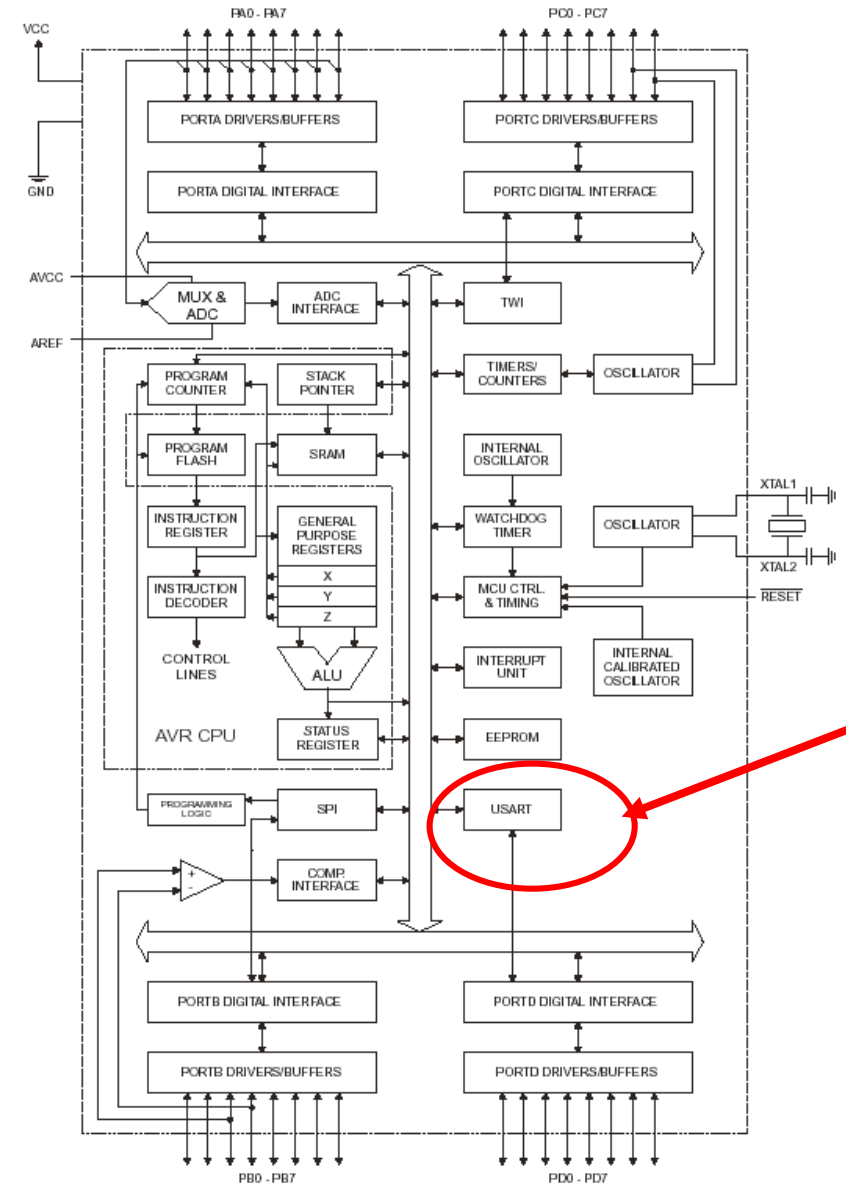
Atmel Studios ”Terminal Vindue”



Kun tilgængelig efter installering af denne extension. Setup-fil på MSYS Blackboard.

Detaljeret blokdiagram

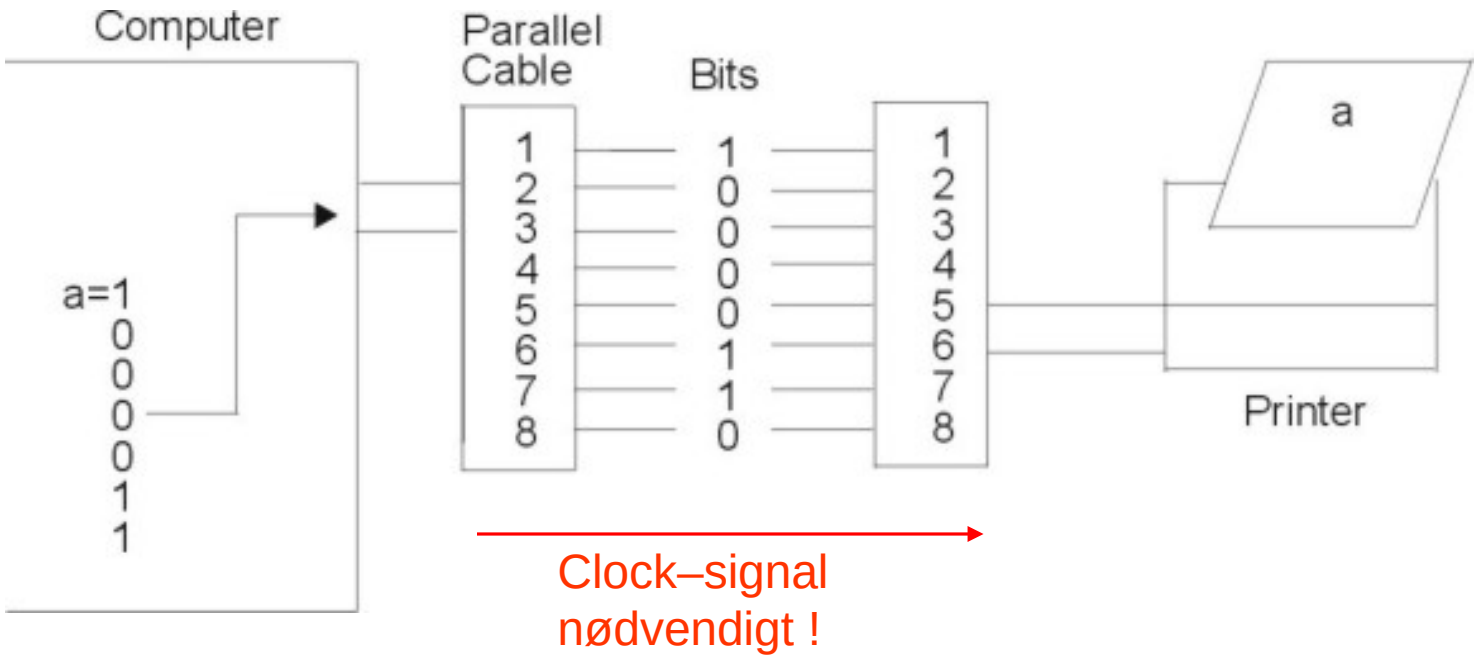
Figure 2. Block Diagram



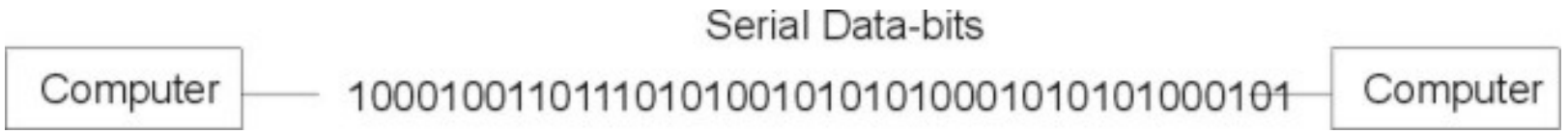
Se det blå hæfte !

USART

Parallel kommunikation



Seriel kommunikation

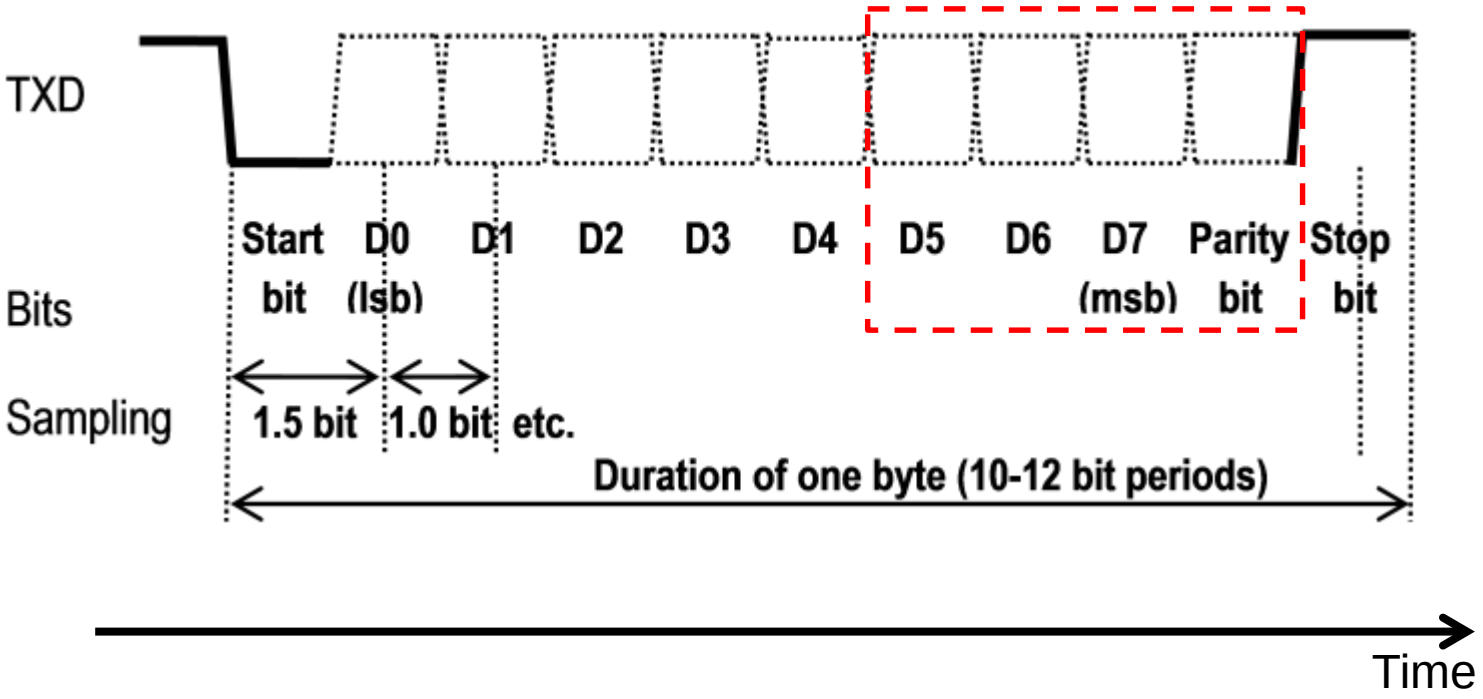


Clock – signal ?

Der er to problemer :

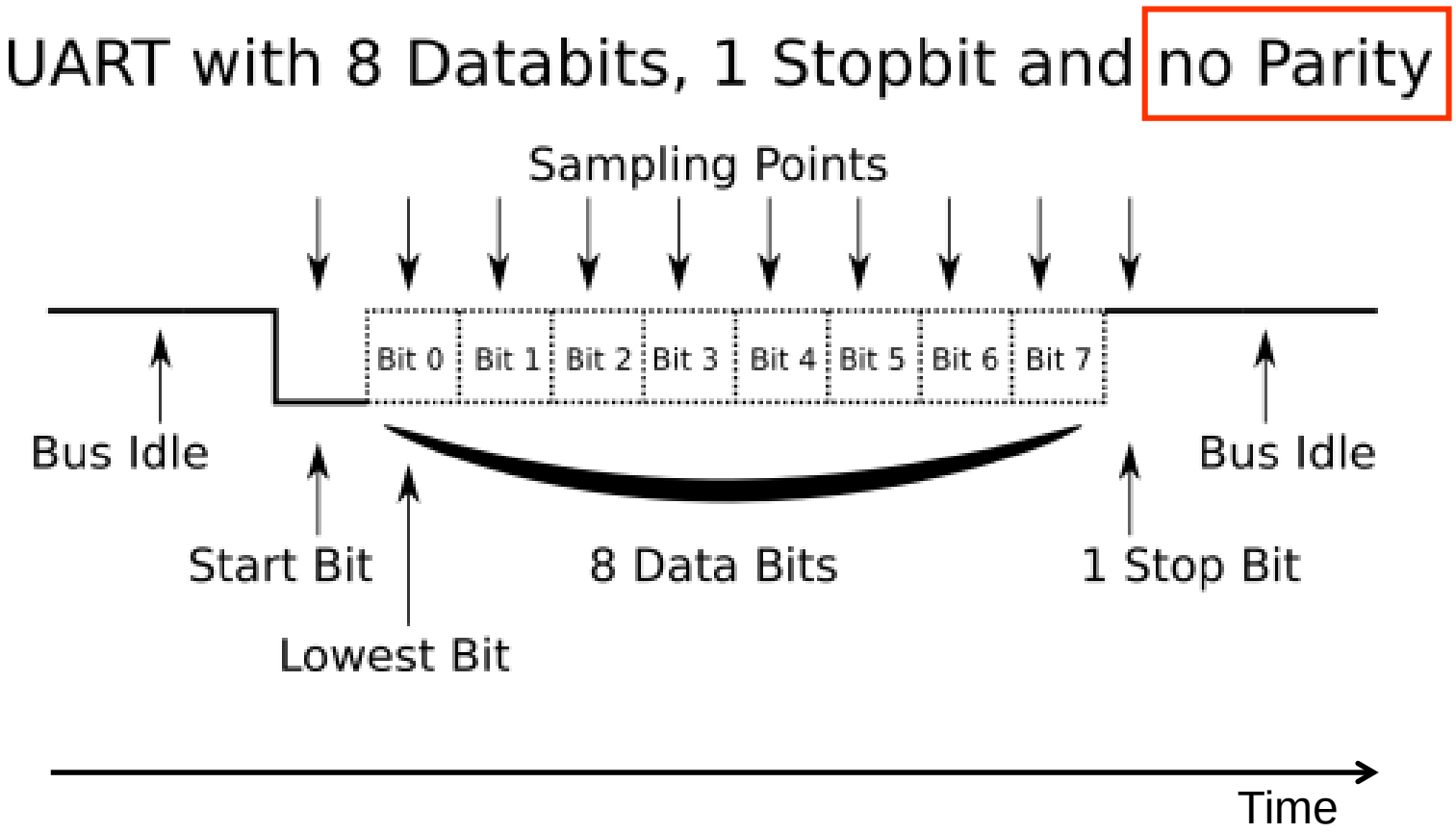
* Bit -synkronisering.
* Byte (”frame”) –synkronisering.

Asynkront format

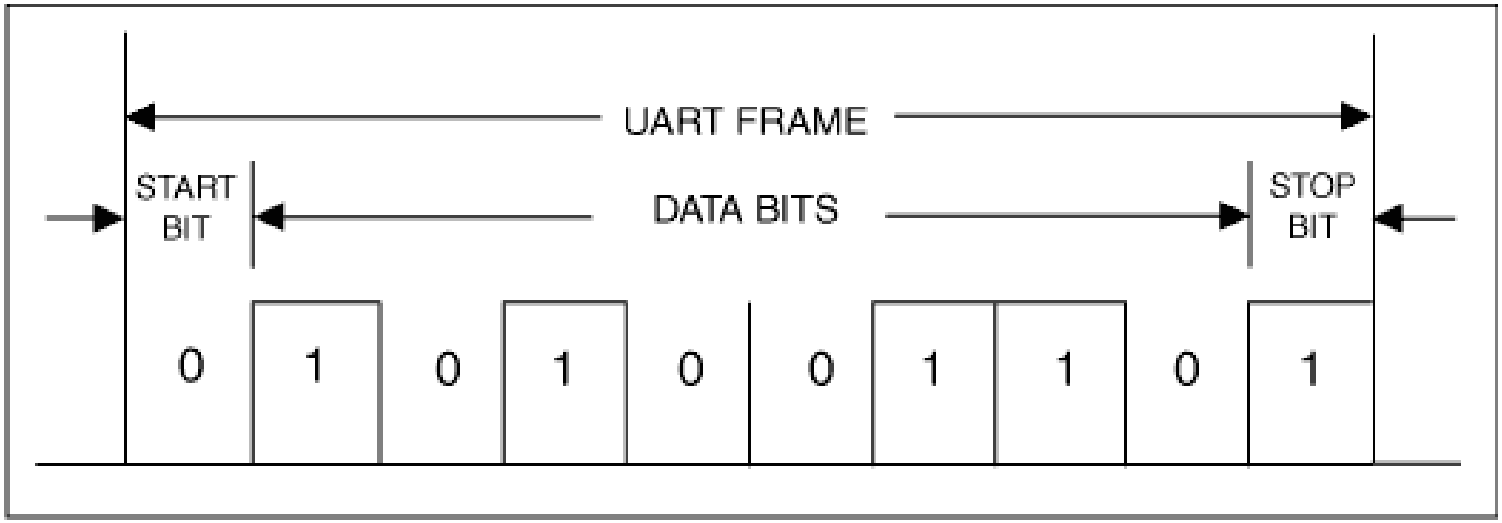


- Data's LSB bliver sendt først
- Antal data bits = 5 – 9 (oftest 8)
- Parity bit er optionel (udelades oftest)

Asynkront format (uden paritet)



Eksempel



Hvilken talværdi sendes ?

ASCII: ”Blot” en måde at kode på

Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]
1	1	1	1	[START OF HEADING]
2	2	10	2	[START OF TEXT]
3	3	11	3	[END OF TEXT]
4	4	100	4	[END OF TRANSMISSION]
5	5	101	5	[ENQUIRY]
6	6	110	6	[ACKNOWLEDGE]
7	7	111	7	[BELL]
8	8	1000	10	[BACKSPACE]
9	9	1001	11	[HORIZONTAL TAB]
10	A	1010	12	[LINE FEED]
11	B	1011	13	[VERTICAL TAB]
12	C	1100	14	[FORM FEED]
13	D	1101	15	[CARRIAGE RETURN]
14	E	1110	16	[SHIFT OUT]
15	F	1111	17	[SHIFT IN]
16	10	10000	20	[DATA LINK ESCAPE]
17	11	10001	21	[DEVICE CONTROL 1]
18	12	10010	22	[DEVICE CONTROL 2]
19	13	10011	23	[DEVICE CONTROL 3]
20	14	10100	24	[DEVICE CONTROL 4]
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]
22	16	10110	26	[SYNCHRONOUS IDLE]
23	17	10111	27	[ENG OF TRANS. BLOCK]
24	18	11000	30	[CANCEL]
25	19	11001	31	[END OF MEDIUM]
26	1A	11010	32	[SUBSTITUTE]
27	1B	11011	33	[ESCAPE]
28	1C	11100	34	[FILE SEPARATOR]
29	1D	11101	35	[GROUP SEPARATOR]
30	1E	11110	36	[RECORD SEPARATOR]
31	1F	11111	37	[UNIT SEPARATOR]
32	20	100000	40	[SPACE]
33	21	100001	41	!
34	22	100010	42	"
35	23	100011	43	#
36	24	100100	44	$
37	25	100101	45	%
38	26	100110	46	&
39	27	100111	47	'
40	28	101000	50	(
41	29	101001	51	)
42	2A	101010	52	*
43	2B	101011	53	+
44	2C	101100	54	,
45	2D	101101	55	-
46	2E	101110	56	.
47	2F	101111	57	/
48	30	110000	60	0
49	31	110001	61	1
50	32	110010	62	2
51	33	110011	63	3
52	34	110100	64	4
53	35	110101	65	5
54	36	110110	66	6
55	37	110111	67	7
56	38	111000	70	8
57	39	111001	71	9
58	3A	111010	72	:
59	3B	111011	73	;
60	3C	111100	74	<
61	3D	111101	75	=
62	3E	111110	76	>
63	3F	111111	77	?
64	40	1000000	100	@
65	41	1000001	101	A
66	42	1000010	102	B
67	43	1000011	103	C
68	44	1000100	104	D
69	45	1000101	105	E
70	46	1000110	106	F
71	47	1000111	107	G
72	48	1001000	110	H
73	49	1001001	111	I
74	4A	1001010	112	J
75	4B	1001011	113	K
76	4C	1001100	114	L
77	4D	1001101	115	M
78	4E	1001110	116	N
79	4F	1001111	117	O
80	50	1010000	120	P
81	51	1010001	121	Q
82	52	1010010	122	R
83	53	1010011	123	S
84	54	1010100	124	T
85	55	1010101	125	U
86	56	1010110	126	V
87	57	1010111	127	W
88	58	1011000	130	X
89	59	1011001	131	Y
90	5A	1011010	132	Z
91	5B	1011011	133	[
92	5C	1011100	134	\
93	5D	1011101	135	]
94	5E	1011110	136	^
95	5F	1011111	137	_
96	60	1100000	140	`
97	61	1100001	141	a
98	62	1100010	142	b
99	63	1100011	143	c
100	64	1100100	144	d
101	65	1100101	145	e
102	66	1100110	146	f
103	67	1100111	147	g
104	68	1101000	150	h
105	69	1101001	151	i
106	6A	1101010	152	j
107	6B	1101011	153	k
108	6C	1101100	154	l
109	6D	1101101	155	m
110	6E	1101110	156	n
111	6F	1101111	157	o
112	70	1110000	160	p
113	71	1110001	161	q
114	72	1110010	162	r
115	73	1110011	163	s
116	74	1110100	164	t
117	75	1110101	165	u
118	76	1110110	166	v
119	77	1110111	167	w
120	78	1111000	170	x
121	79	1111001	171	y
122	7A	1111010	172	z
123	7B	1111011	173	{
124	7C	1111100	174	|
125	7D	1111101	175	}
126	7E	1111110	176	~
127	7F	1111111	177	[DEL]

Baud Rate

- Baud Rate = Antal bits per sekund.
- Baud Rate = 1 / bit-tiden.
- Eksempel : 9600 Baud ~ bit-tid = 104 uS.

- Standard Baud Rates :
  110, 300, 600, 1200, 2400, 4800
  9600, 14400, 19200
  38400, 57600, 115200

Test (”socrative.com”: Room = MSYS)

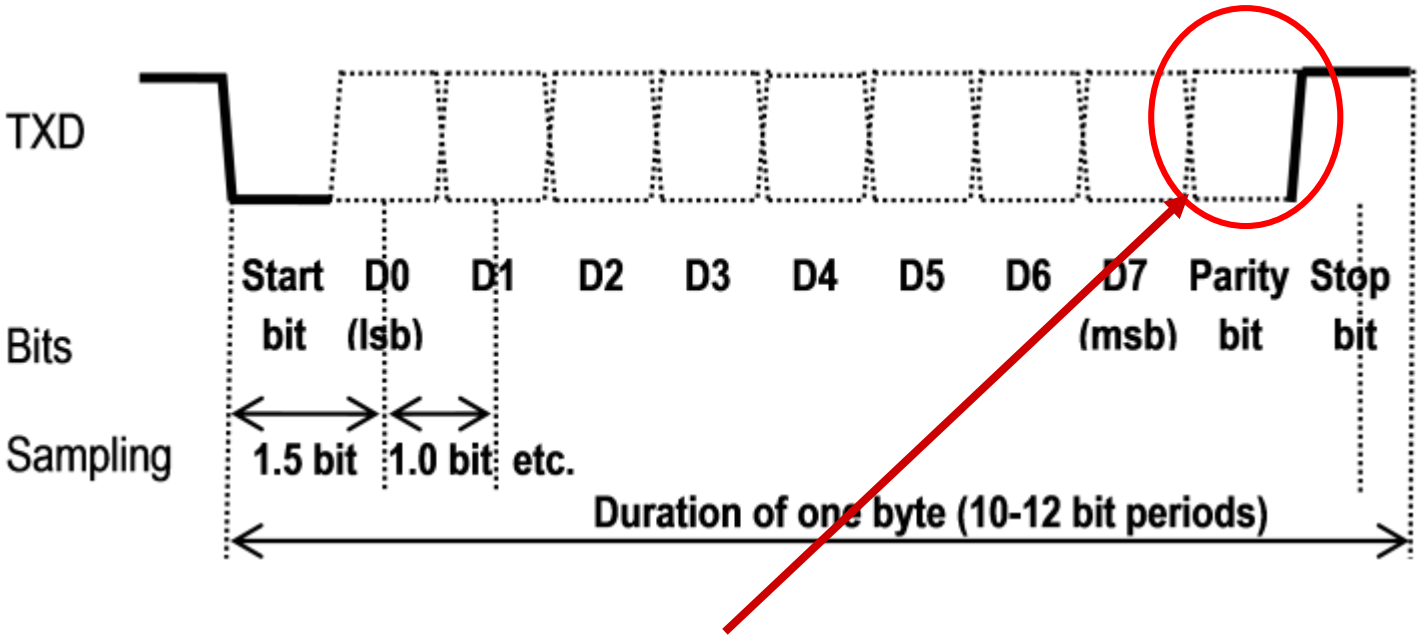
Hvor lang tid tager det at sende et tegn via en UART, når der anvendes:

* Baud rate = 9600 bit/s
* 1 startbit
* 8 databit
* Ingen paritet
* 1 stopbit

- A: Cirka 10 ms.
- B: Cirka 100 us.
- C: Cirka 1 ms.
- D: Cirka 10 us.

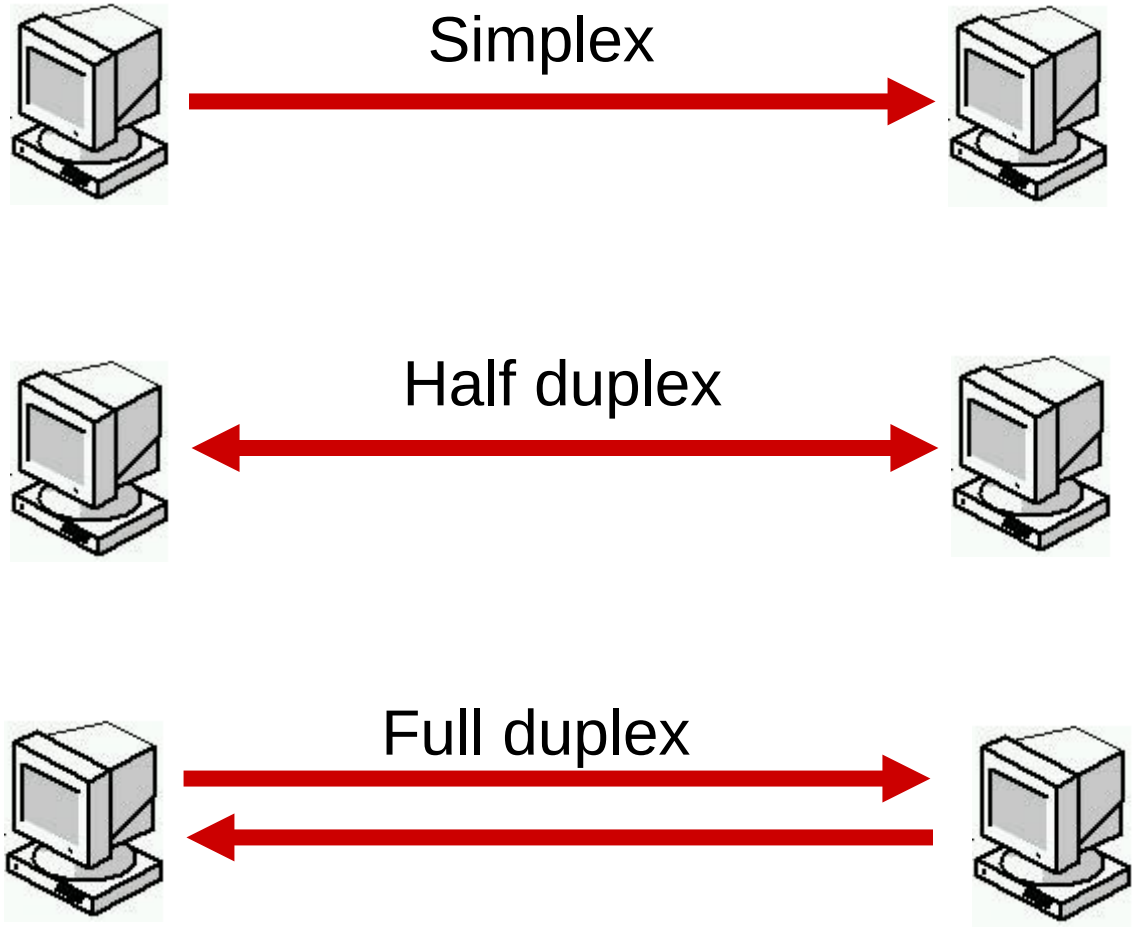
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Paritet

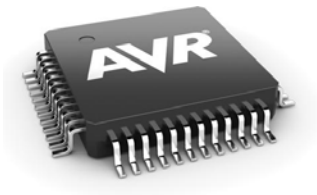


- "**Even** parity" : **Antallet af 1-taller** (i data og P-bit) skal være et **lige** tal.
- "**Odd** parity" : **Antallet af 1-taller** (i data og P-bit) skal være **ulige** tal.
- "**No parity**" anvendes oftest (ingen paritets-bit).

Simplex / duplex

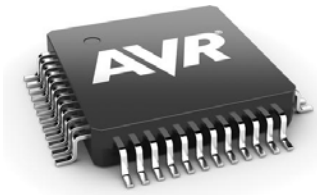


Kommunikations-standarder



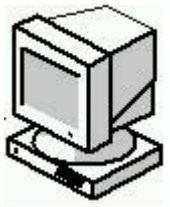
TTL/CMOS (0 - 5 volt)

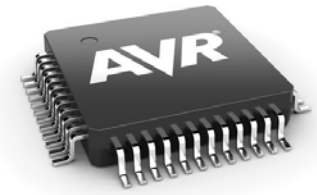
UART **UART**



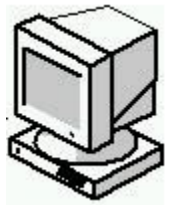
UART / RS232 **RS232** **UART / RS232**

UART / RS232 **RS232** **USB / RS232**





UART / USB **USB**



AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Ofte anvendt standard: RS232

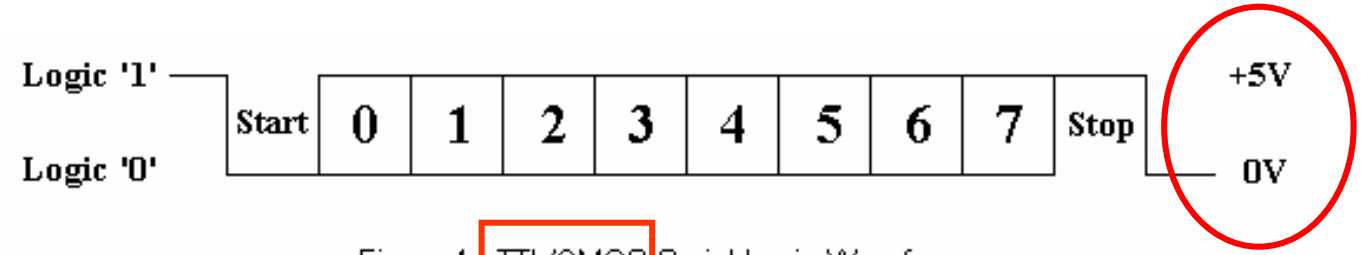


Figure 4 TTL/CMOS Serial Logic Waveform

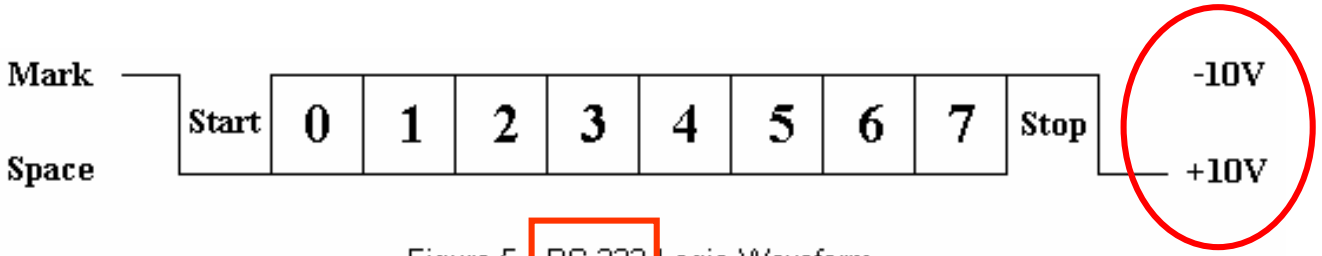
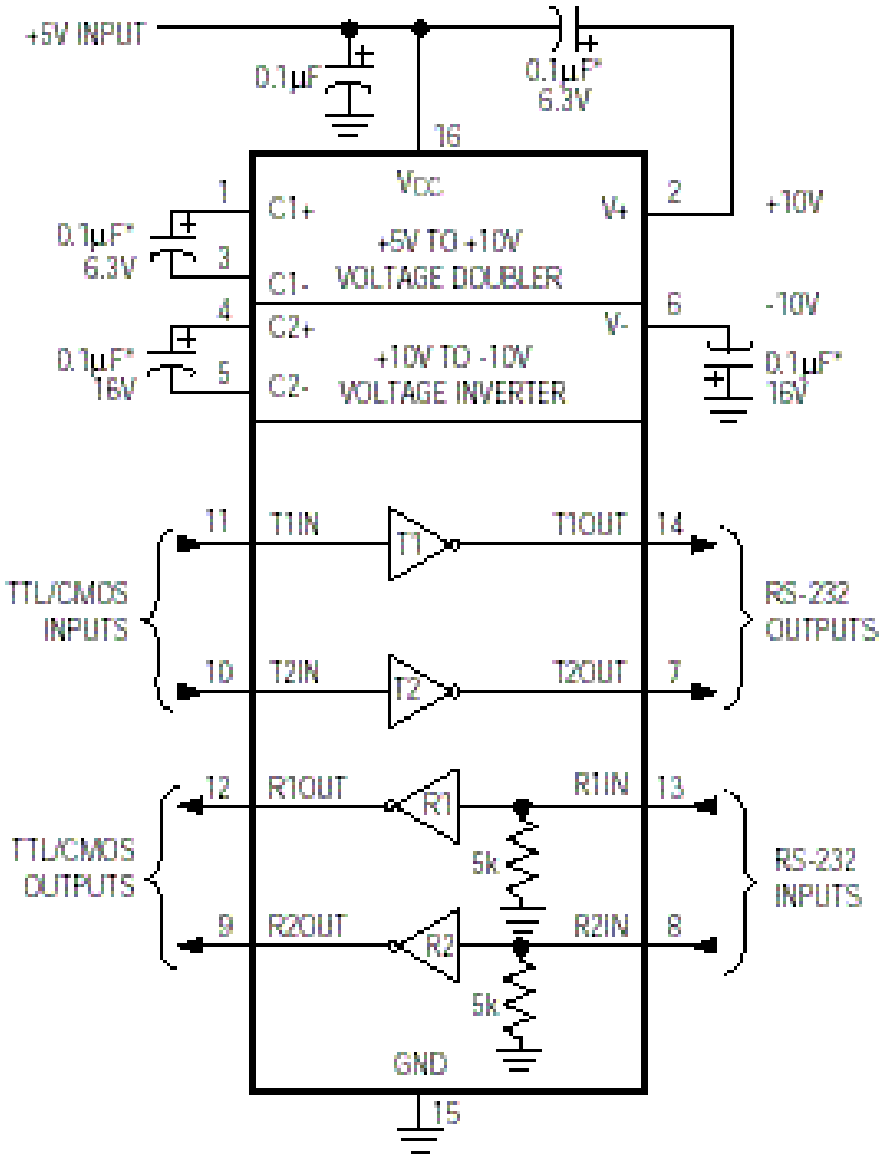


Figure 5 RS-232 Logic Waveform

- Logisk 0 : +3 til +25 volt.
- Logisk 1 : -3 til -25 volt.

MAX232 tranceiver-kreds



Pris ~ 1-2 kroner

Eksempel med RS232 tranceivers

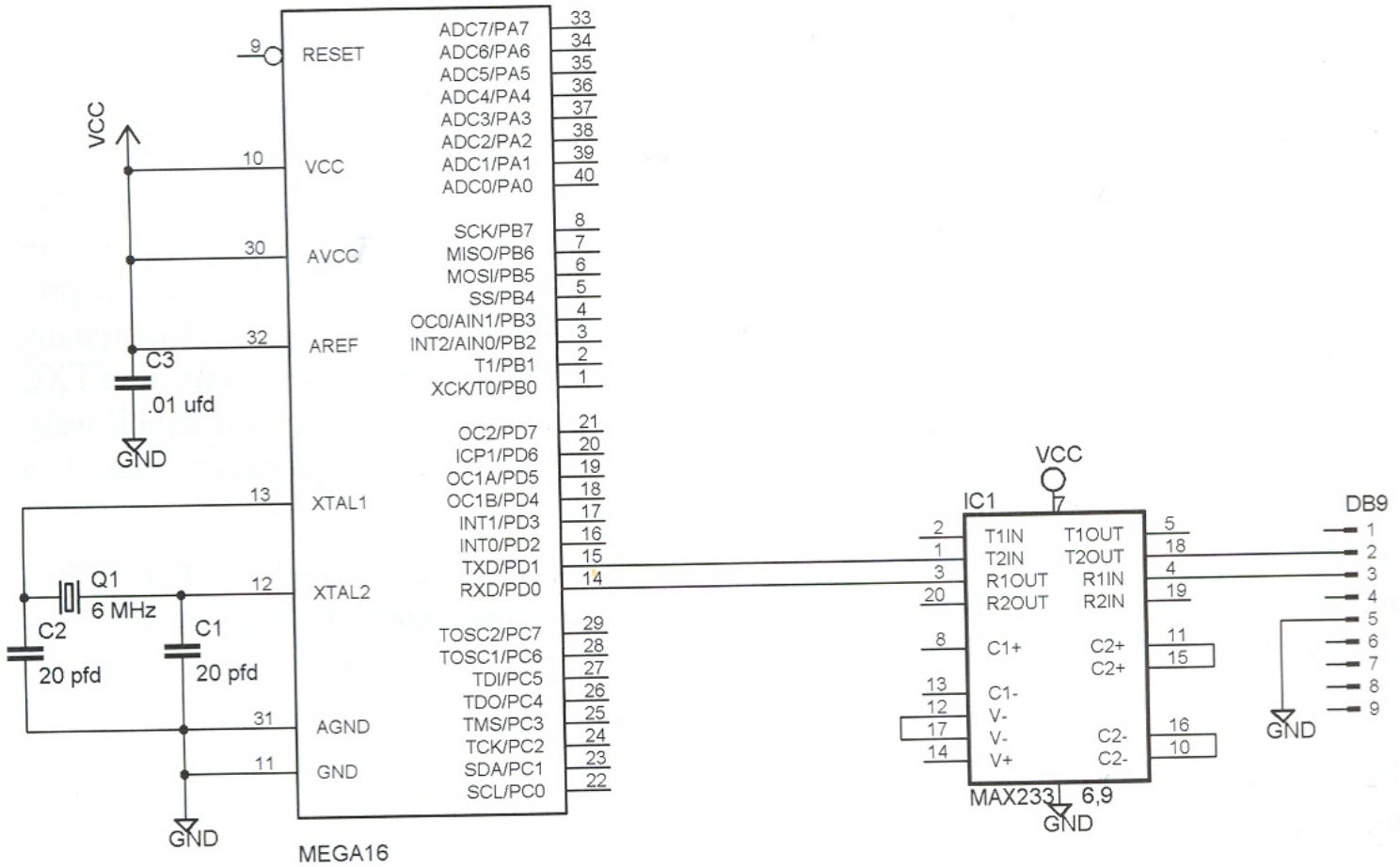
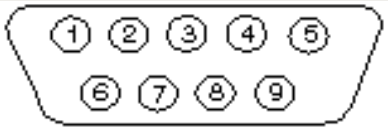


Figure 2–39 *Serial Communication Example Hardware*

Typisk RS232 port (COM-port)

	9 Pin Connector on a DTE device (PC connection)
Male RS232 DB9	
Pin Number	**Direction of signal:**
1	Carrier Detect (CD) (from DCE) Incoming signal from a modem
2	Received Data (RD) Incoming Data from a DCE
3	Transmitted Data (TD) Outgoing Data to a DCE
4	Data Terminal Ready (DTR) Outgoing handshaking signal
5	Signal Ground Common reference voltage
6	Data Set Ready (DSR) Incoming handshaking signal
7	Request To Send (RTS) Outgoing flow control signal
8	Clear To Send (CTS) Incoming flow control signal
9	Ring Indicator (RI) (from DCE) Incoming signal from a modem

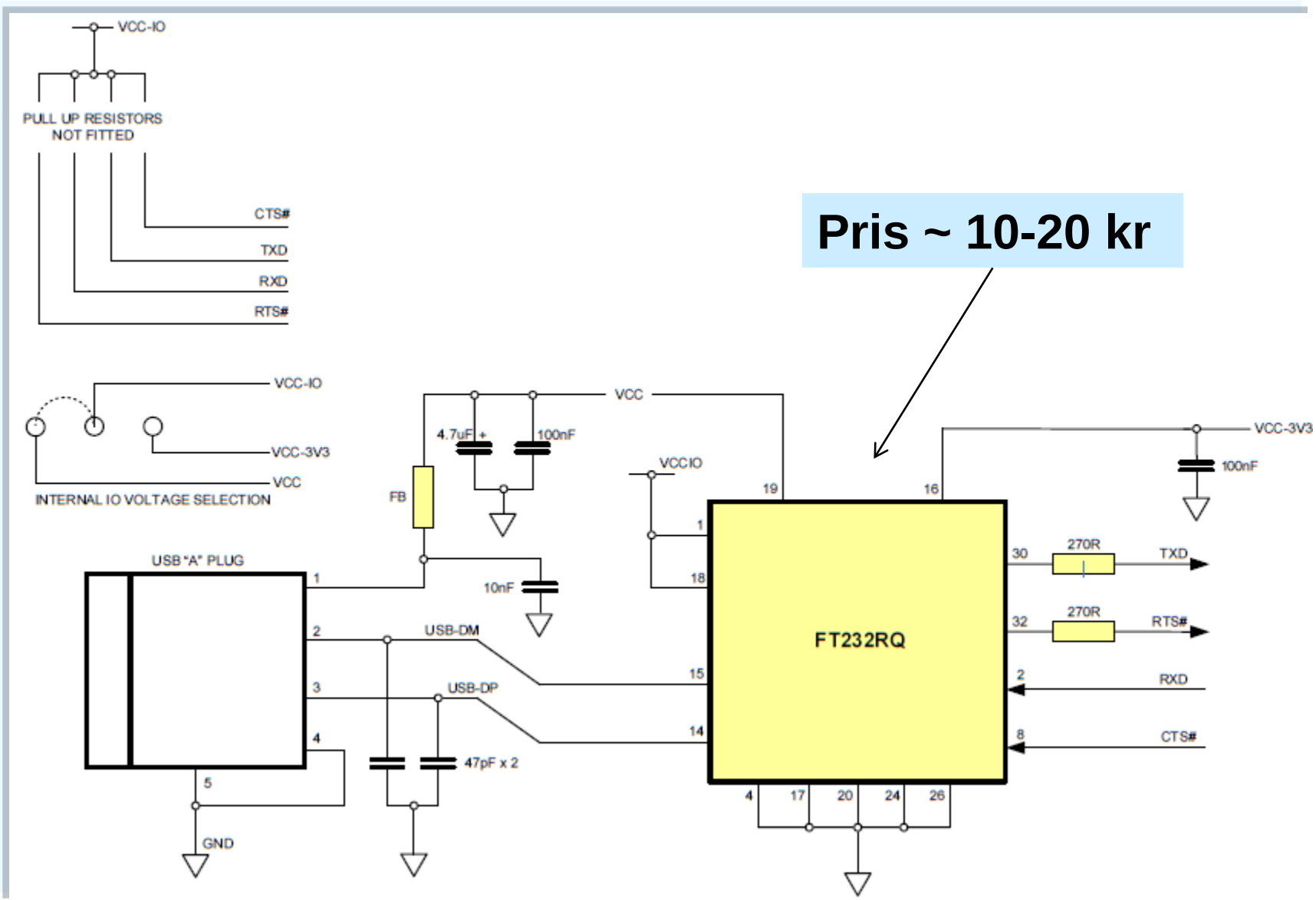
De øvrige signaler KAN anvendes til ”handshake”.

RS232 max kabellængde

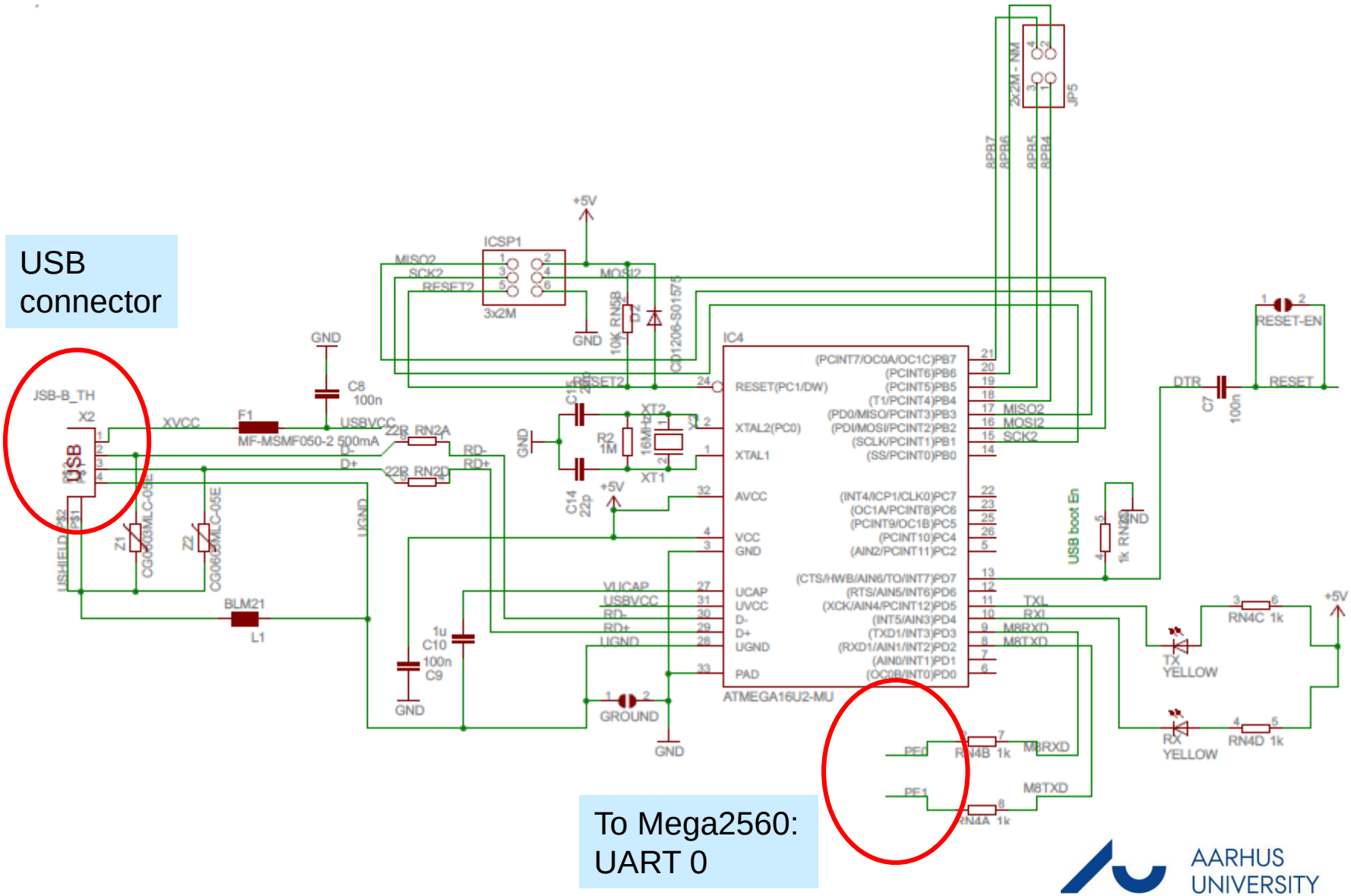
Baud Rate	Shielded Cable Length	Unshielded Cable Length
110	5000	1000
300	4000	1000
1200	3000	500
2400	2000	500
4800	500	250
9600	250	100

(Målt i fødder !)

USB/UART Converter



USB/UART konverter i Arduino Mega2560

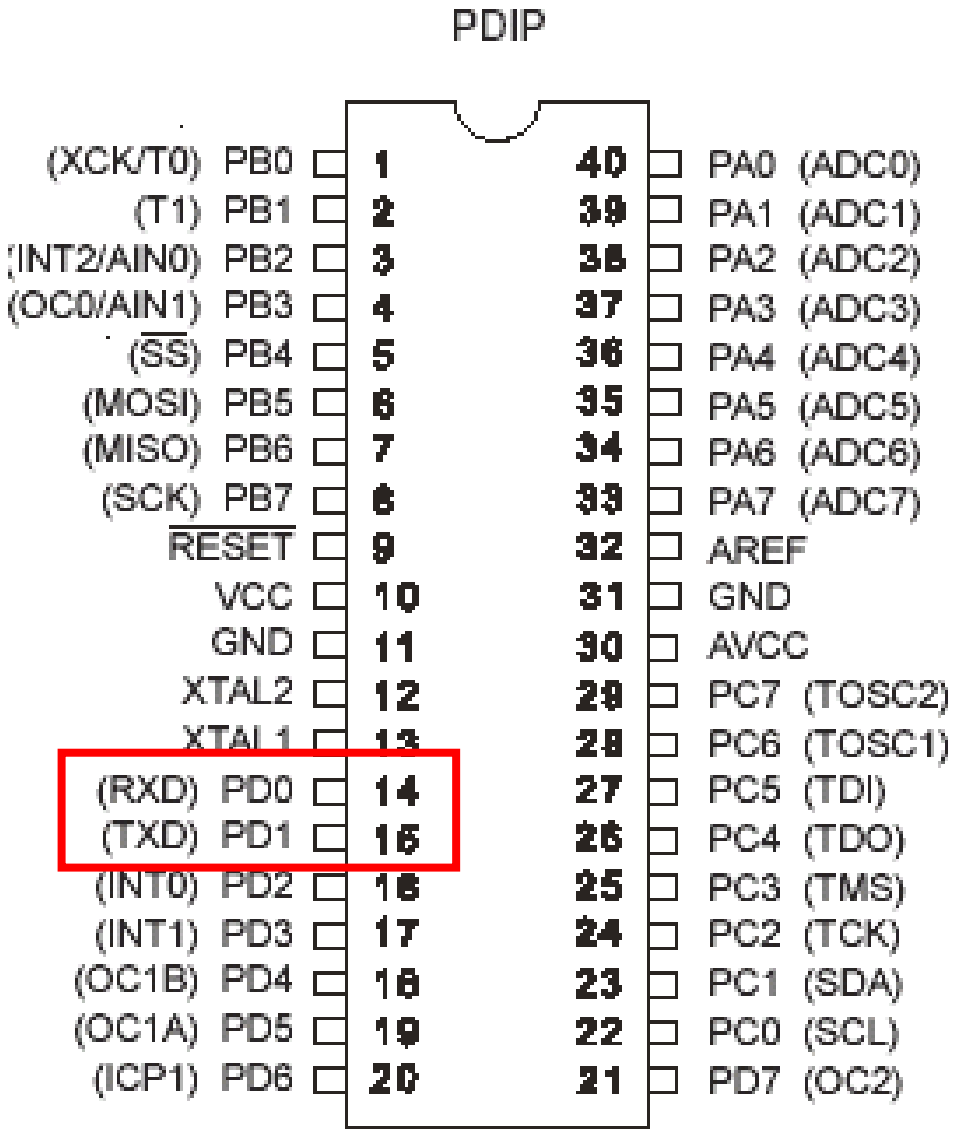


AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

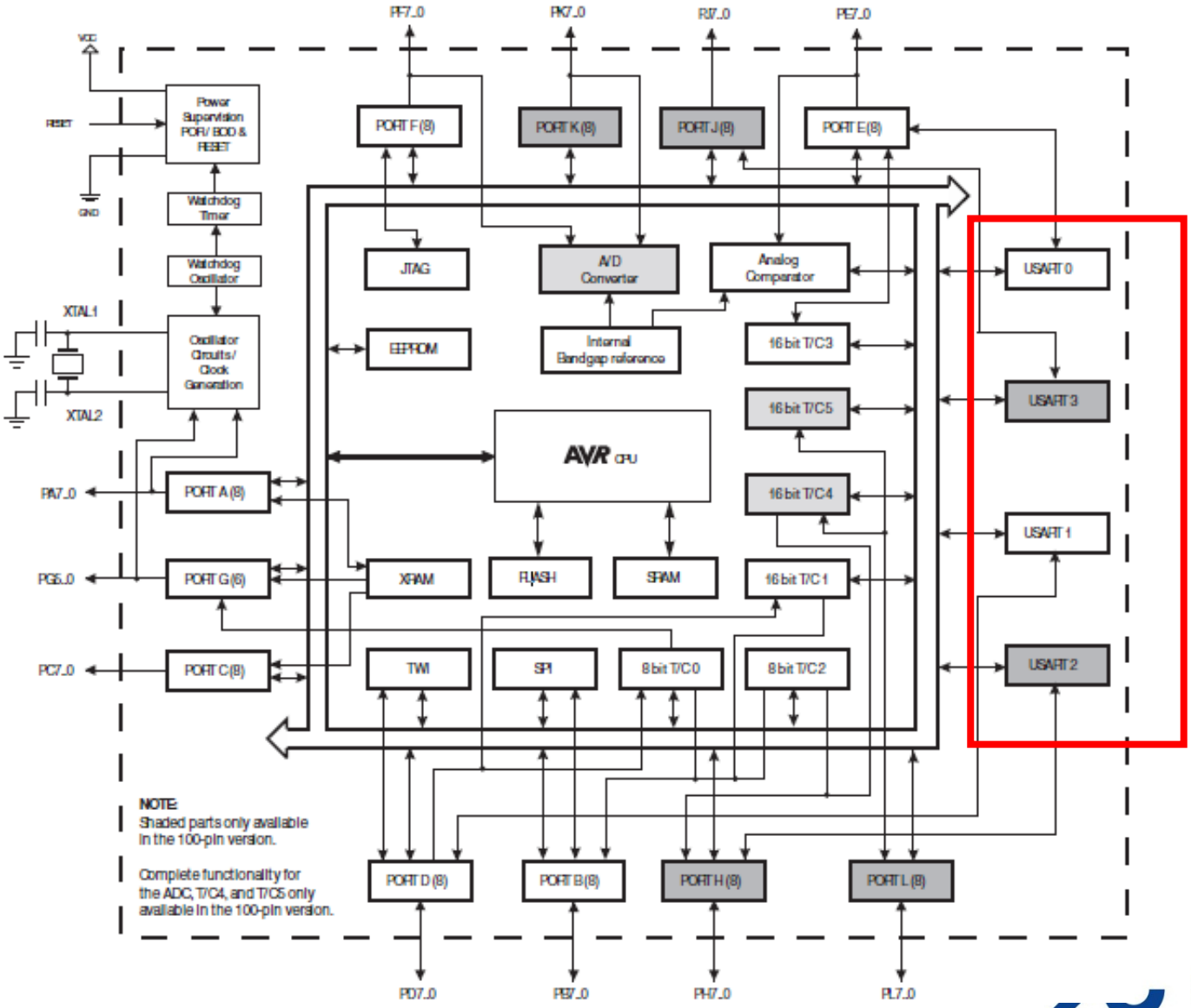
Mega32: RXD og TXD pins

RXD = PD ben 0.

TXD = PD ben 1.



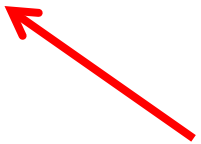
Mega2560: 4 USART'er



Mega2560: USART pins

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

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

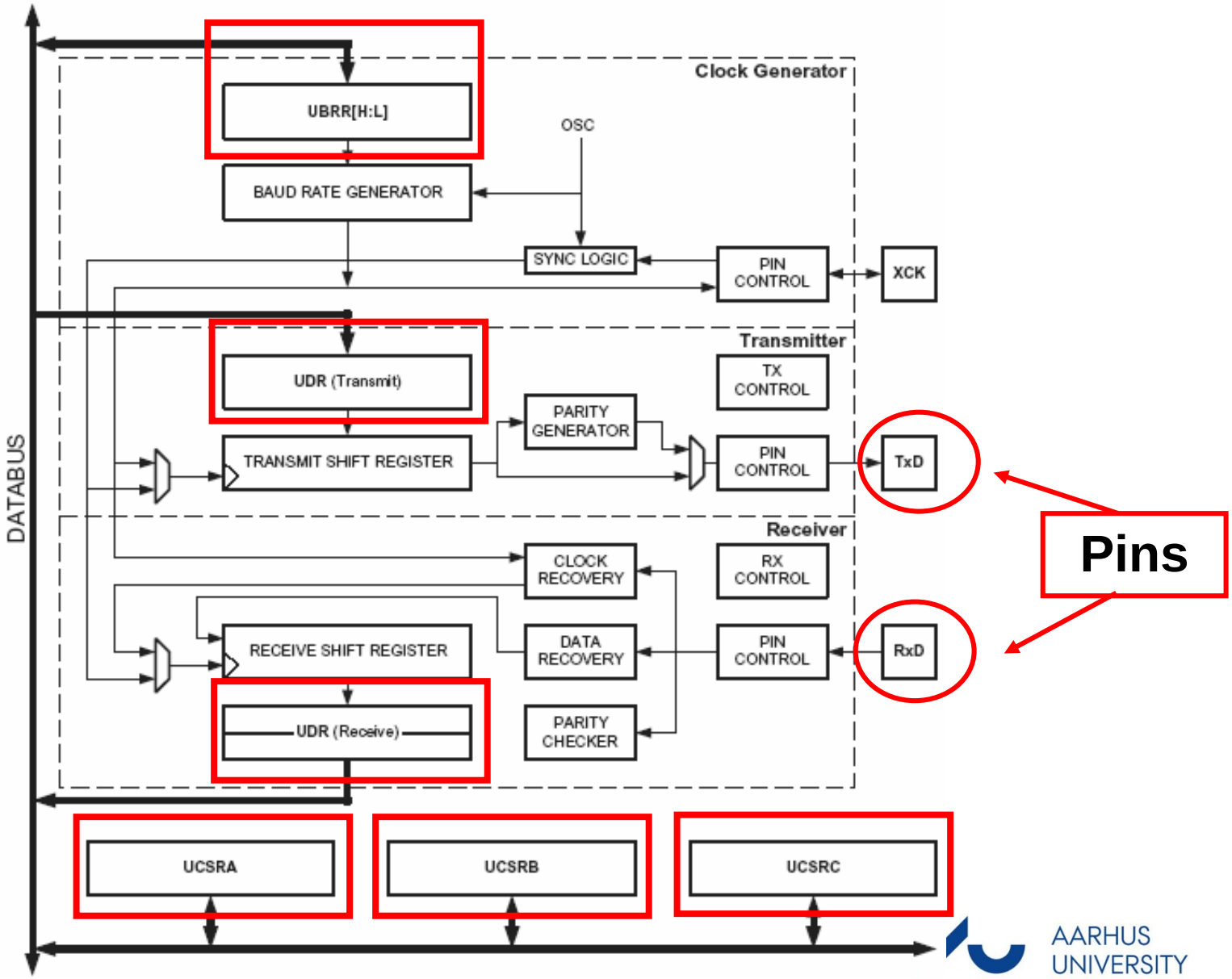


Arduino Mega2560
USB - stikket

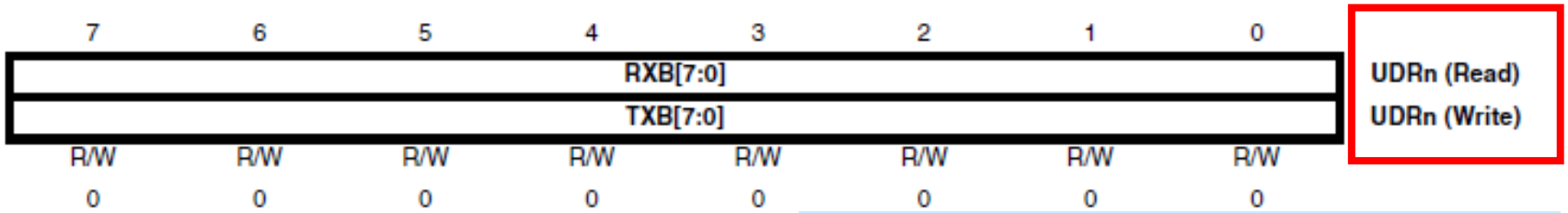
USART2 :
TXD2 = PH, ben 1
RXD2 = PH, ben 0

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

Mega32/Mega2560 USART



Mega2560: UDRn. Usart Data Registers



UDR0, UDR1, UDR2 eller UDR3

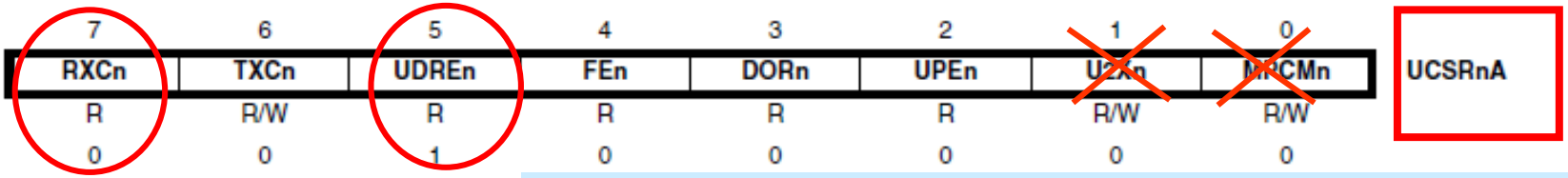
Bemærk: Fysisk to forskellige registre med samme navn (UDRn)

```
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 UDR”).

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 UDR”).

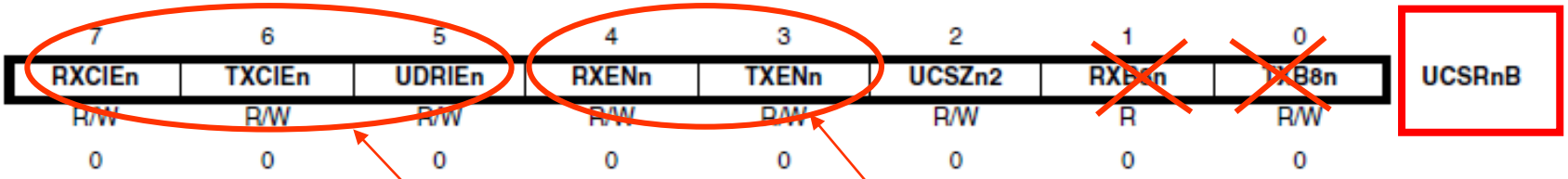
FEn : ”Framing Error” (modtaget tegn har fejl i stop bit).

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

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

Mega2560: UCSRnB. Control and Status Register B

UCSR0B, UCSR1B, UCSR2B eller UCSR3B



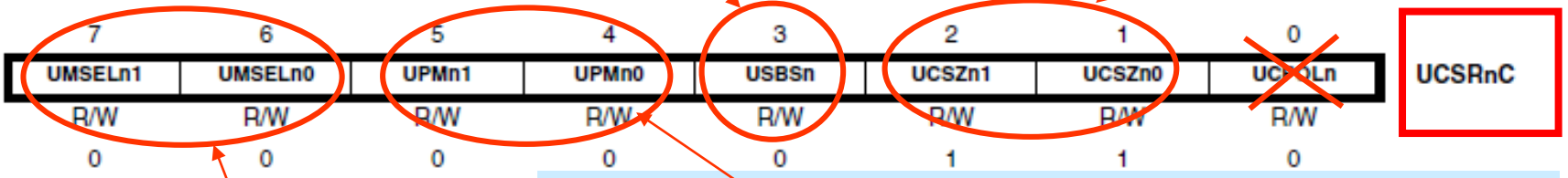
Bruges ved interrupt -styret USART

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

Mega2560: UCSRnC. Control and Status Register C

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

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



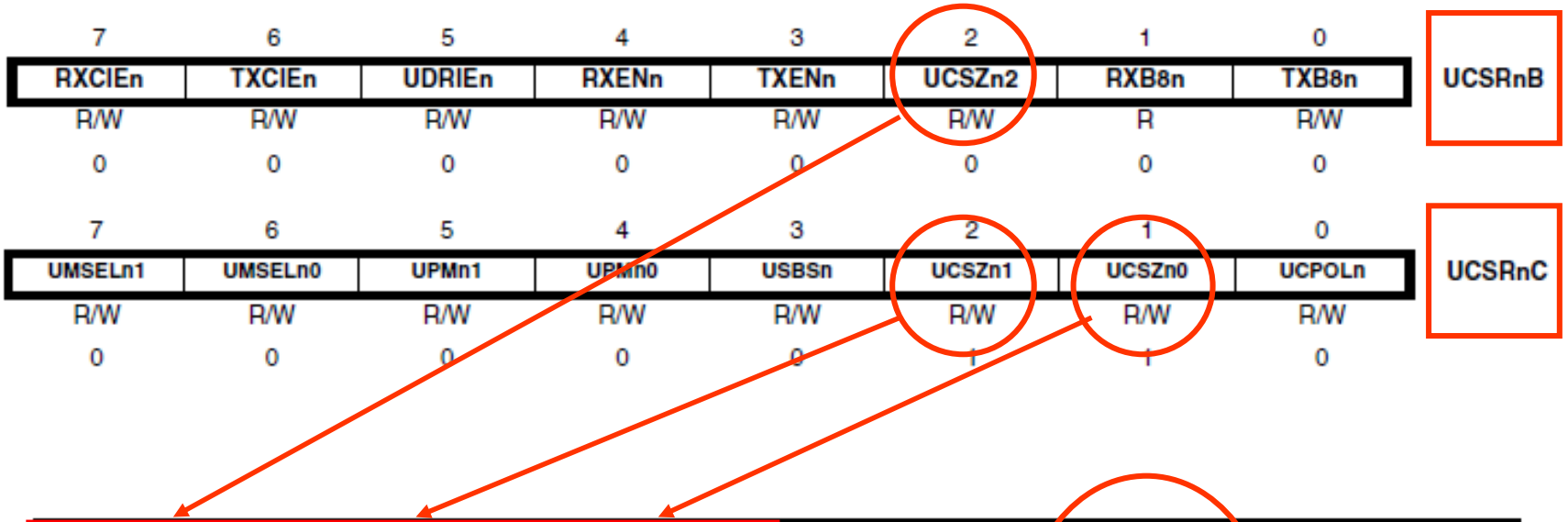
UCSR0C, UCSR1C, UCSR2C eller UCSR3C

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

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

Test (”socrative.com”: Room = MSYS)

Hvilket bit i registeret UCSRnA vil gå højt, når UART'en har modtaget et nyt tegn ?

- A: Bit 7: RXC
- B: Bit 6: TXC
- C: Bit 5: UDRE
- D: Bit 4: FE
- E: Bit 2: PE

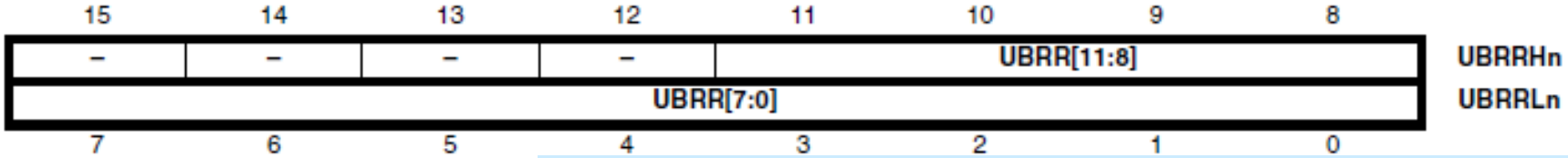
Test (”socrative.com”: Room = MSYS)

Før man må sende et nyt tegn via UART 0, skal man sikre sig, at senderen er klar til at modtage tegnet. Hvilken metode er korrekt ?

- A: while ((UCSR0A | 0b00100000) == 0)
  { }
- B: while ((UCSR0A & 0b00100000) == 0)
  { }
- C: while ((UCSR0A & 0b00100000) != 0)
  { }
- D: while (UDR0 != 0)
  { }

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)	$BAUD = \frac{f_{OSC}}{8(UBRRn+1)}$	$UBRRn = \frac{f_{OSC}}{8BAUD} - 1$

”UBRRn” = (256 * UBRRHn) + UBRRLn

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Baud rate afrundingsfejl ved 16 MHz

Baud Rate [bps]	f_{osc} = 16.0000MHz			
	U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4K	68	0.6%	138	-0.1%
19.2K	51	0.2%	103	0.2%
28.8K	34	-0.8%	68	0.6%
38.4K	25	0.2%	51	0.2%
57.6K	16	2.1%	34	-0.8%
76.8K	12	0.2%	25	0.2%
115.2K	8	-3.5%	16	2.1%
230.4K	3	8.5%	8	-3.5%
250K	3	0.0%	7	0.0%
0.5M	1	0.0%	3	0.0%
1M	0	0.0%	1	0.0%
Max.(1)	1Mbps		2Mbps	

”UBRR” = (256 * UBRRH) + UBRRL

Test (”socrative.com”: Room = MSYS)

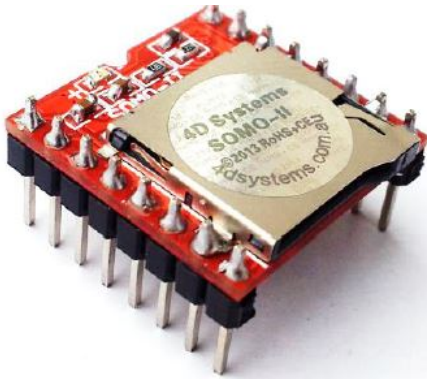
Under initiering af en Mega2560's UART 1 skrives følgende værdi til UBRR1:
UBRR1 = 3332;

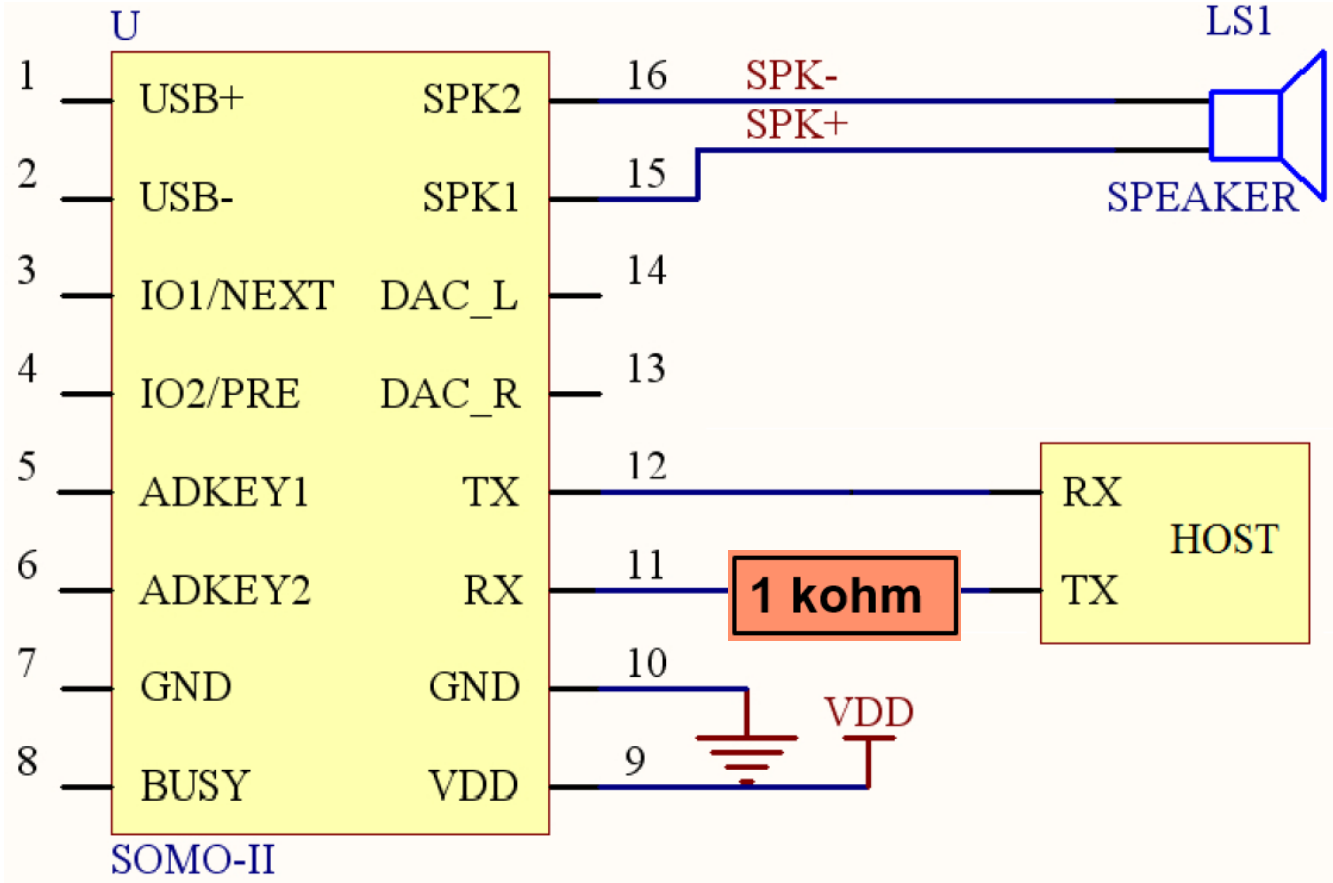
Mega2560's CPU clockfrekvens er 16 MHz.
Hvilken baud rate anvendes ?

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

AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Eksempel: SOMO-II MP3 player





Eksempel: SOMO-II MP3 player

FORMAT: $S, CMD, Feedback, Para1, Para2, Checksum1, Checksum2, $0		
$S	Start Character $S is 0x7E in HEX	Every command starts with this
CMD	Command Code	Every command has a unique command code, which determines the operation
Feedback	Command Feedback	Specifies whether feedback is required by the host microcontroller in reply to the command. 1 = Feedback, 0 = No Feedback
Para1	Parameter #1	First parameter of the specific Command Code
Para2	Parameter #2	Second parameter of the specific Command Code
Checksum1	Checksum #1	First byte of the checksum. Checksum calculation shown below.
Checksum2	Checksum #2	Second byte of the checksum. Checksum calculation shown below.
$0	End Character $0 is 0xEF in HEX	Every command ends with this

Checksum Calculation:

The checksum is calculated using the following formula.

Checksum (2 bytes) = 0xFFFF – (CMD + Feedback + Para1 + Para2) + 1

Eksempel: SOMO-II MP3 player

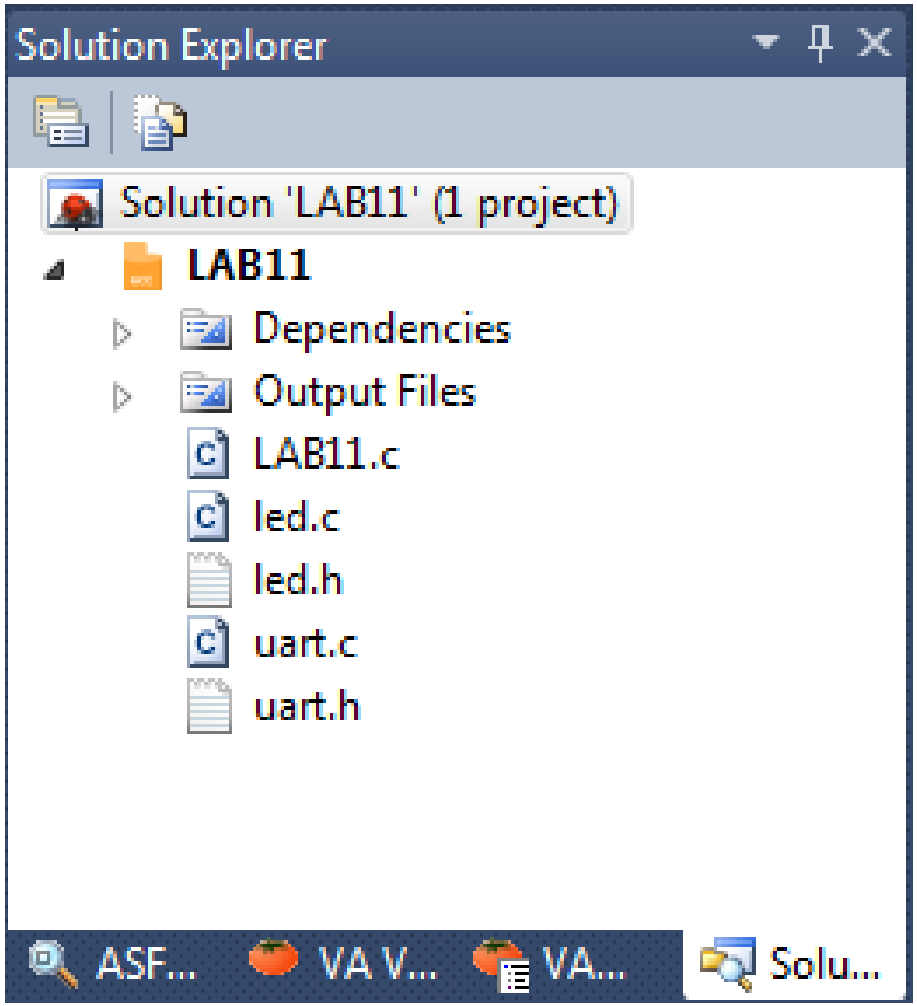
Function	Serial Command	Description
NEXT	7E 01 00 00 00 FF FF EF	If no track is currently playing, issuing the **NEXT** command will start playing the first track copied to the media (see Section 6). If the SOMO-II is currently playing a song or has previously played a song, this will play the next song in the order copied on to the media.
PREVIOUS	7E 02 00 00 00 FF FE EF	If no track is currently playing, issuing the **PREVIOUS** command will start playing the last track copied to the media (see Section 6). If the SOMO-II is currently playing a song or has previously played a song, this will play the previous song in the order copied on to the media.
SPECIFY TRACK #	7E 03 00 00 **01** FF FC EF 7E 03 00 00 **02** FF FB EF 7E 03 00 00 **0A** FF F3 EF	Start playing the first track copied to the media. (See Section 6) This will start playing the second track copied to the media. This will start playing the tenth track copied to the media.
VOLUME +	7E 04 00 00 00 FF FC EF	This will increase the volume by 1
VOLUME -	7E 05 00 00 00 FF FB EF	This will decrease the volume by 1
VOLUME #	7E 06 00 00 **1E** FF DC EF 7E 06 00 00 **05** FF F5 EF	This will set the volume to be 30 (30 is the Max) This will set the volume to be 5
SPECIFY EQ	7E 07 00 00 **01** FF F8 EF 7E 07 00 00 **04** FF F5 EF	This will set the EQ to pop This will set the EQ to classic (0/1/2/3/4/5 Normal, Pop, Rock, Jazz, Classic, Bass)
REPEAT A TRACK	7E 08 00 00 **01** FF F7 EF 7E 08 00 00 **02** FF F6 EF 7E 08 00 00 **1F** FF D9 EF	This will repeat the first track copied to the media. (See Section 6) This will repeat the second track copied to the media. This will repeat the thirty first track copied to the media.

LAB11: UART driver

```
/*****************************************
* "uart.h":                              *
* Header file for Mega2560 UART driver. *
* Using UART 0.                          *
* Henning Hargaard, 4/11 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.

LAB11 filer


Solution Explorer
Solution 'LAB11' (1 project)
LAB11
Dependencies
Output Files
LAB11.c
led.c
led.h
uart.c
uart.h
ASF...
VA V...
VA...
Solu...

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 UBRR0, skal i funktionen beregnes på basis af "BaudRate"-parameteren og Mega2560's CPU clockfrekvens (afrunding kan forekomme).

Desuden skal UART'en initieres til:

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

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).

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;
}
```

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.

Ekstraopgaver for ”nørden”

1. Implementer en SW UART transmitter, der anvender et vilkårligt portben som TX terminal.

2. For ”nørden”:
   Implementer en SW UART receiver, der anvender et vilkårligt portben som RX ben.
   Der kan med fordel anvendes et eksternt interruptben.
   Bemærk, at det er mere kompliceret at implementere en receiver end en transmitter.

Slut på lektion 17