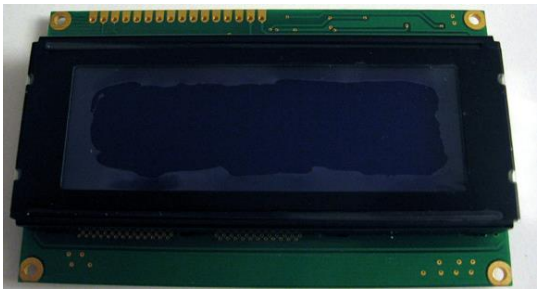
LCD Character display-Intro

Parallel interface, Forbindelsesdiagram, Ram & Rom-struktur,
Biblioteksfunktioner til at styre LCD-skærmen,
Lcd.Print vs Lcd.Write,
Selvdefinerede karakterer, herunder æ, ø & å.
Karaktergenerator,

Seriel display mangler:

Der findes flere typer af LCD karakter-displays, fra forskellige firmaer, med de virker typisk ens.

Her er vist en type, der er blå, og med parallel interface, men der findes også seriel interface, der kan bruges, hvis der skal spares på pins.



Pins: Nummer 1 fra venstre

Parallel interface:

Fælles for parallel interface til et LCD-modul er, at det er nødvendig med flere parallelle forbindelser fra uC-en. 2 kontrolsignaler, og mindst data-4 bit.

Her er vist en nærmere beskrivelse af de forskellige ben.

På det printudlæg, der kan downloades til uC og LCD-display, er der udlagt, således, at ledningerne blot skal forbindes "lige over".

Pin 3 er til at justere skærmens kontrast. Denne pin skal tilsluttes en justerbar spænding mellem 0 og 5 Volt.
Der kan bruges et potmeter på fx 4,7 eller 10 KOhm.

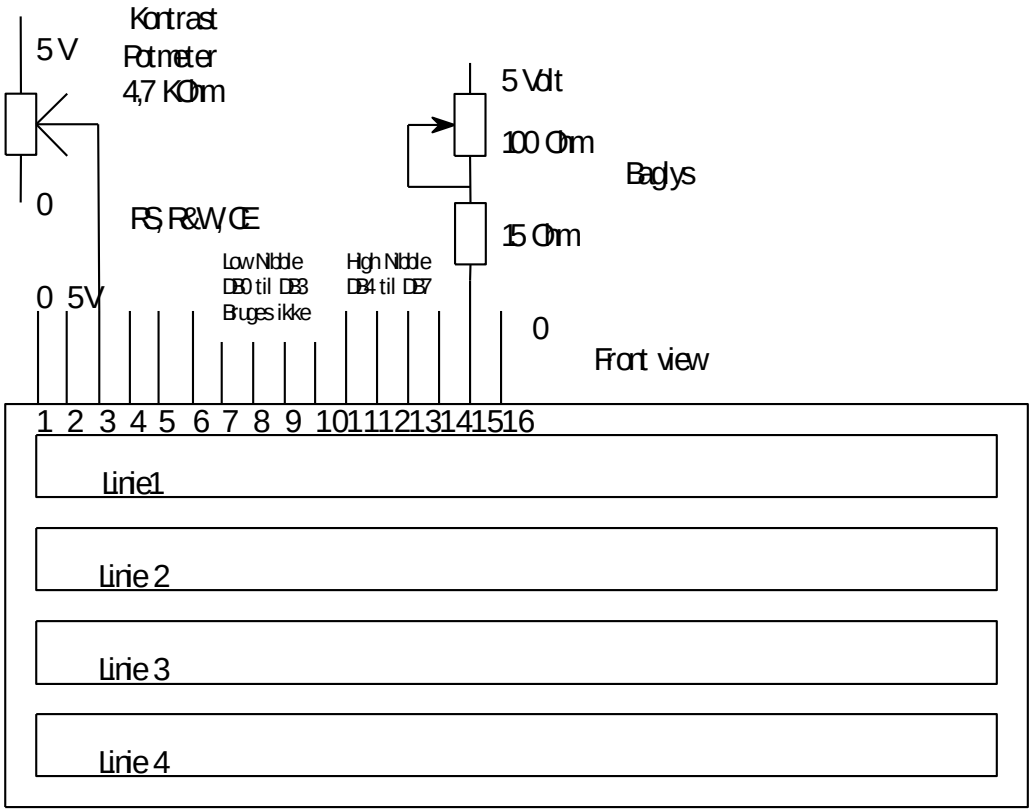
Skærmen bruges normalt i 4 bit mode, dvs. at low nible ikke bruges, altså pin 7 – 10 fra venstre.

Pin 15 og 16 er forbundet til lysdioder bag skærmen. De oplyser LCD'en bagfra – Baglys-, og det er praktisk ved brug om natten.

Pin No.	Symbol	Level	Description
1	V_{SS}	0V	GND
2	V_{DD}	5.0V	Supply Voltage for logic
3	VO	(Variable)	Contrast Adjustment
4	RS	H/L	Register select signal
5	R/W	H/L	H: Read(MPU→Module) L: Write(MPU→Module)
6	E	H,H→L	Chip enable signal
7	DB0	H/L	Data bus line
8	DB1	H/L	Data bus line
9	DB2	H/L	Data bus line
10	DB3	H/L	Data bus line
11	DB4	H/L	Data bus line
12	DB5	H/L	Data bus line
13	DB6	H/L	Data bus line
14	DB7	H/L	Data bus line
15	A/Vee	–	+4.2V for LED (RA=0ohm)/Negative Voltage output
16	K	–	Power supply for B/L(0V)

Top ↑

Forbindelsesdiagram for LCD-skærmen:



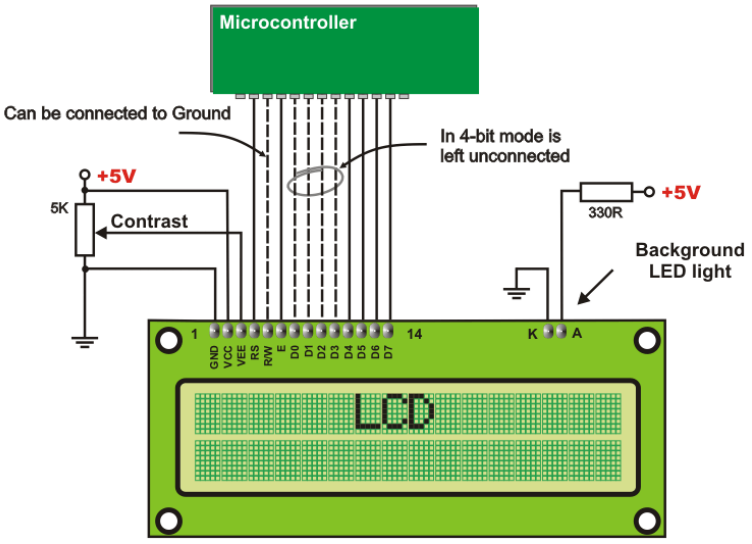
Som det ses herover har skærmen 4 linjer. På hver linje er der plads til 20 karakterer.

Winstar WH2004A

Her er vist et andet diagram-eksempel

K og A er vist byttet om på denne skitse:

Arduinos bibliotek til Liquid Christal bruger 4-bit mode.



Top ↑

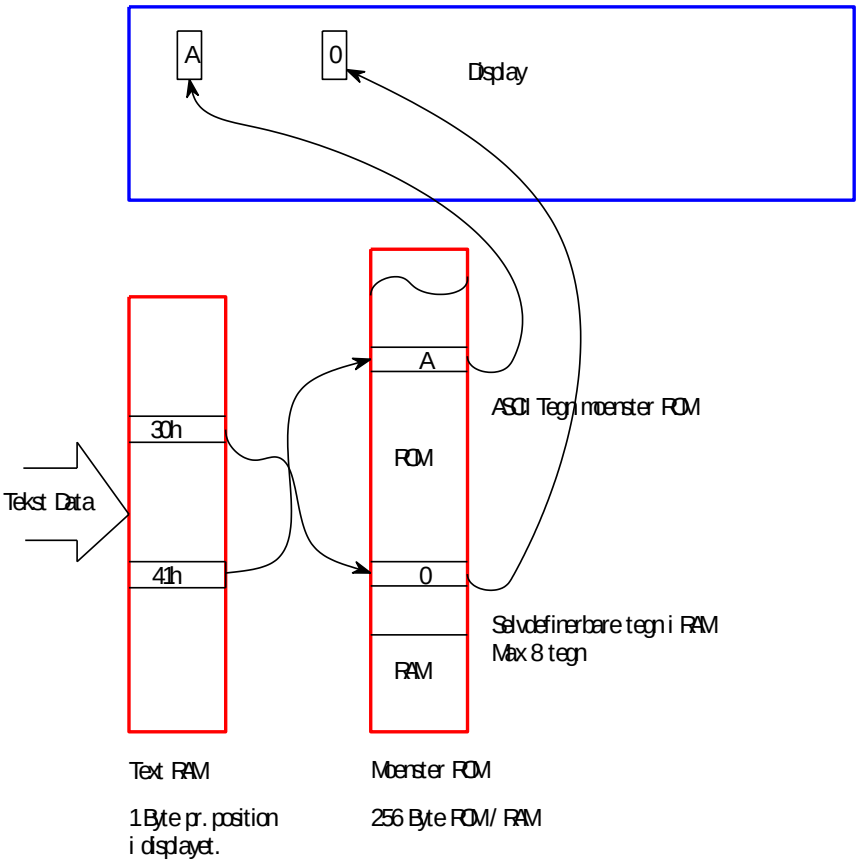
RAM / ROM-struktur i LCD-Panelet.

Når der skrives data til et LCD-display, skal man først placere en cursor, dvs. der, hvor en tekst skal skrives.

Data gemmes i en RAM-adresse, der gælder for pågældende placering.

Hver plads i displayet har sin egen RAM.

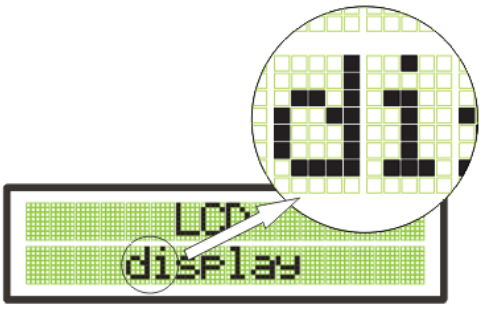
Værdien der gemmes i pladsens RAM-adresser, refererer herefter til en adresse i en ROM, hvor der er defineret et mønster for ønsket karakter. Dvs. at de enkelte pixels på pladsen tændes, så ønsket karakter kan ses.



Bag hver karakter plads i displayet er der en 8-bit RAM. Hvis der skrives en værdi i denne, vil den fungere som en pointer til en indbygget mønstertabel, hvis mønster vil fremkomme på LCD-skærmen. Mønsteret bliver til bogstaver og tegn, jfr. ASCII tabellen, så fx 30h sendt til displayet viser et nul-tal, 31h viser et 1-tal osv. Som angivet i ASCII-tabellen.

Men det betyder også, at der ikke er mønstre for de specielle danske karakterer.

Men det er der råd for. Se senere:



Et skema der viser ram-adresser i displayet, der refererer til hver karakter på de 4 linjer.

DISPLAY POSITION

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FIRST LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
SECOND LINE	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
THIRD LINE	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
FOURTH LINE	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

DD RAM ADDRESS

Af skemaet fremgår, at hvis der skrives for lang en string til linje 0, fortsætter den på linje 2.

Top ↑

ASCII - The **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

LCD'en har en Cursor, der viser, hvilken Karakterplads med tilhørende RAM, der er selected. Når data sendes til Displayet, placeres de automatisk i RAM'en for cursorens position.

Når der er sendt en byte til displayet, flytter cursoren automatisk 1 fremad.

Ud over data til at skrive tekst på displayet, kan der sendes kontrolkoder til at styre displayets opførsel. Det styres af bittet Registerselect, RS. Se senere.

Clear al tekst i LCD-en

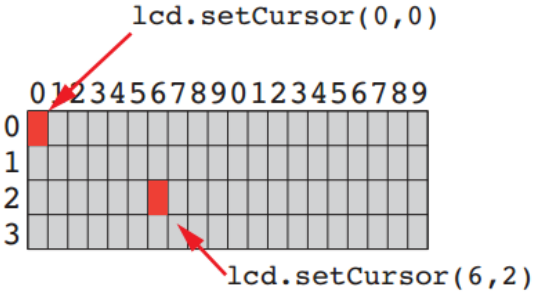
```
lcd.clear();          // Clearer alle 4 linjer
```

Placer cursoren:

```
lcd.setCursor(x,y);   // max 19,3
```

Her er vist en funktion, der ligger i biblioteket, til at placere cursoren.

```
lcd.setCursor(x,y);
```



Top ↑

Arduino-Biblioteksfunktioner til LCD

```
#include <LiquidCrystal.h>  // include the library code:

const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;  // initialize the library

LiquidCrystal lcd(rs, en, d4, d5, d6, d7);


void setup() {
    lcd.begin(20, 4);   // set up the LCD's number of columns and rows:

}


// manipulating screen:

lcd.setCursor(0, 0); // set the cursor to (0,0):  ( vandret: 0 til 19, nedad: 0 til 3 )

lcd.clear();        // clear screen
lcd.home();     // go to (0,0):

lcd.print(thisChar);    // Print variable.
lcd.print("hello, world!"); // Print string.

lcd.cursor();       // turn on the cursor:
lcd.noCursor(); // Turn off the cursor:

lcd.autoscroll();       // set the display to automatically scroll:
lcd.noAutoscroll(); // turn off automatic scrolling

lcd.scrollDisplayLeft();  // scroll one position left:

lcd.scrollDisplayRight();  // scroll one position right:

lcd.blink();    // Turn on the blinking cursor:
lcd.noBlink();

lcd.noDisplay();    // Turn off the display:
lcd.display();

lcd.rightToLeft();      // go right for the next letter
lcd.leftToRight();      // go left for the next letter
```

Top ↑

LCD.Print vs. LCD.Write:

Der er to måder I Arduino-verdenen at skrive tekst til en LCD-skærm.

Print-metoden konverterer selv fx en variabel om, så man kan læse variablens værdi på skærmen.

Dvs. at fx en variabel har værdien 28, dvs. den er gemt som 0001 1100 binært, vil blive omdannet til 2 bytes, 32h og 38h før de sendes til skærmen. Se ASCII-tabellen !!.

LCD.Write

lcd.write(thisLetter); Skriver en hexværdi til en plads, - uden at konvertere den til ASCII.

Top ↑

Selvdefinerede karakterer:

Upload af selvdefinerede karakterer. Se eksempel i Arduino IDE: Fil / Eksempler / LicuidCrystal / CustumCharacter.

Her ses et eksempel på hvordan karakterer vises i et LCD-display. Det behøver ikke nødvendigvis være ens for forskellige typer display.

Men fælles er, at der på de 8 laveste pladser, her vist med grøn, er plads til, at man selv kan uploade et mønster, så det er muligt at lave sine egne bogstaver eller karakterer.

Skemaet skal forstås sådan:

Øverst vises vandret Upper Nibble, og lodret lower nibble.

Dvs. af fx 0100-1000 b vil vise et 'H'.

Kilde: http://www.lcdinterfacing.info/Custom-HD44780U-characters.php

Men der er plads til 8 stk. selvdefinerbare tegn. De skal sendes til LCD'ens ROM-adresse 0 til 7.

Dvs. at man kan uploade selvdefinerede karakterer. Der er plads til 8 stk. på adresserne 0 til 7.

På nettet findes flere steder, hvor man kan få hjælp til at definere sine egne karakterer.

Karaktererne defineres i et 5 x 8 matrix

De skal kopieres ind i en Sketch, og uploades til LCD-en i Setup() eller evt. "on the fly".

Nederste række bør være = 00h pga. at cursoren normalt er her.

Pixels

Output

```
byte customChar[8] = {
        0b00000,
        0b01100,
        0b01100,
        0b00000,
        0b00010,
        0b00010,
        0b11110,
        0b00000
};
```

Selvdefinerede tegn kan evt. erstatte de, jeg har lavet i følgende eksempler, i linje 0 og 7.

Her følger 3 eksempler: Vælg evt. én og kopier ind i en Arduino-sketch.

Eksempel 1:

```
/*  LCD-displayet har en RAM-adresse knyttet til alle displayets
    20 x 4 karakter-positioner.
    Den værdi, der skrives til pågældende RAM, refererer til en
    bitmønstertabel.

    Heldigvis er det lavet sådan, at hvis der fx skrives værdien
    65 decimal, eller 41 hex, som i ASCII-tabellen svarer til et A,
    vil der skrives et A på displayet.

    I ASCII-tabellen er der ikke bit-mønstre for de danske karakterer.
    Men heldigvis er LCD-displayet's indbyggede Mønster-ROM udlagt som RAM
    på pladserne  0 til 7.

    Dette gør det muligt, at uploade 8 selv-definerede mønstre til LCD-en.

    / Valle  16/3-2013, 4/6-2019

*/


#include <LiquidCrystal.h>      // include the library code:

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // rs, en, D4, D5, D6, D7 )

//  Definer 8 karakter-mønstre

byte newChar0[8] = {  B10000,B01000,B00100,B00010,B00001,B00010,B00100,B00000}; // >
byte Lae[8] ={  B00000,B00000,B11010,B00101,B01111,B10100,B11111,B00000}; // æ
byte Loe[8] ={  B00000,B00001,B01110,B10101,B10101,B01110,B10000,B00000}; // ø
byte Laa[8] ={  B00100,B00000,B01110,B00001,B01111,B10001,B01111,B00000}; // å
byte Sae[8] ={  B01111,B10100,B10100,B11110,B10100,B10100,B10111,B00000}; // Æ
byte Soe[8] ={  B00001,B01110,B10011,B10101,B11001,B01110,B10000,B00000}; // Ø
byte Saa[8] ={  B00100,B00000,B01110,B10001,B11111,B10001,B10001,B00000}; // Å
byte VT[8] = {  B11111,B10101,B10101,B01110,B01110,B00100,B00100,B00000}; // VT

//  >, æ, ø, å, Æ, Ø, Å, VT  er placeret i RAMadresse 0, 1, 2, 3, 4, 5, 6, 7

//----------------------

void setup()
{
  lcd.createChar(0, newChar0);   // upload 8 selvdef. karakterer
  lcd.createChar(1, Lae);  // æ
  lcd.createChar(2, Loe);  // ø
  lcd.createChar(3, Laa);  // å
  lcd.createChar(4, Sae);  // Æ
  lcd.createChar(5, Soe);  // Ø
  lcd.createChar(6, Saa);  // Å
  lcd.createChar(7, VT);   // VT  ( ;-)

  lcd.clear();  // Nødvendig efter Upload

  lcd.begin( 20, 4 );   //
}
```

```
//--------------------------------

  void loop()
  {
    ////.......danske karakterer skrives som:............////
    lcd.clear();
    lcd.print(char(0));  // Skriver >
    lcd.print(char(1));  // Skriver æ
    lcd.print(char(2));  // skriver ø
    lcd.write(3);        // skriver å
    lcd.write(4);        // skriver Æ
    lcd.write(5);        // skriver Ø
    lcd.write(6);        // skriver Å
    lcd.write(7);        // skriver VT

    lcd.setCursor( 3, 2 );    // Flyt cursor: ( pos, lin ), 0-19, 0-3

    lcd.print("S" );      // Skriv Sønderborg
    lcd.print( char(2));
    lcd.print("nderborg");

  while(1); // stop looping
};
```

Eksempel 2: Her er upload lagt ud i en funktion

```
/*  LCD-displayet har en RAM-adresse knyttet til alle displayets
    20 x 4 karakter-positioner.
    Den værdi, der skrives til pågældende RAM, refererer til en
    bitmønstertabel.

    Heldigvis er det lavet sådan, at hvis der fx skrives værdien
    65 decimal, eller 41 hex, som i ASCII-tabellen svarer til et A,
    vil der skrives et A på displayet.

    I ASCII-tabellen er der ikke bit-mønstre for de danske karakterer.
    Men heldigvis er LCD-displayet's indbyggede Mønster-ROM udlagt som RAM
    på pladserne  0 til 7.

    Dette gør det muligt, at uploade 8 selv-definerede mønstre til LCD-en.

    Denne sketch viser hvordan

    / Valle Thorø, d. 16/3-2013 & 4/6-2019
*/


#include <LiquidCrystal.h>      // include the library code:

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // rs, en, D4, D5, D6, D7 )


void setup()
{
  upload_char();       // kald upload-funktion

 lcd.begin( 20, 4 );   //

}
```

```
  void loop()
  {
    ////.......danske karakterer skrives som:.............////

    lcd.print(char(0));  // Skriver >
    lcd.print(char(1));  // Skriver æ
    lcd.print(char(2));  // skriver ø
    lcd.write(3);        // skriver å
    lcd.write(4);        // skriver Æ
    lcd.write(5);        // skriver Ø
    lcd.write(6);        // skriver Å
    lcd.write(7);        // skriver VT

    lcd.setCursor( 3, 2 );

    lcd.write("S");
    lcd.print( char(2));
    lcd.print("nderborg");    // Skriv Sønderborg



  while(1); // stop looping
};

void upload_char()  // Funktion
{
//  Definer 8 karakter-mønstre
byte _0[8] = {  B10000,B01000,B00100,B00010,B00001,B00010,B00100,B00000}; // >
byte Lae[8] ={  B00000,B00000,B11010,B00101,B01111,B10100,B11111,B00000}; // æ
byte Loe[8] ={  B00000,B00001,B01110,B10101,B10101,B01110,B10000,B00000}; // ø
byte Laa[8] ={  B00100,B00000,B01110,B00001,B01111,B10001,B01111,B00000}; // å
byte Sae[8] ={  B01111,B10100,B10100,B11110,B10100,B10100,B10111,B00000}; // Æ
byte Soe[8] ={  B00001,B01110,B10011,B10101,B11001,B01110,B10000,B00000}; // Ø
byte Saa[8] ={  B00100,B00000,B01110,B10001,B11111,B10001,B10001,B00000}; // Å
byte VT[8] = {  B11111,B10101,B10101,B01110,B01110,B00100,B00100,B00000}; // VT
//  >, æ, ø, å, Æ, Ø, Å, VT  er placeret i RAMadresse 0, 1, 2, 3, 4, 5, 6, 7

// Upload:

  lcd.createChar(0, _0);   // upload 8 selvdef. karakterer
  lcd.createChar(1, Lae);  // æ
  lcd.createChar(2, Loe);  // ø
  lcd.createChar(3, Laa);  // å
  lcd.createChar(4, Sae);  // Æ
  lcd.createChar(5, Soe);  // Ø
  lcd.createChar(6, Saa);  // Å
  lcd.createChar(7, VT);   // VT  ( ;-)
  lcd.clear();
}
```

Eksempel 3: Upload i funktion, og upload-data er lagt i et Array.

```
/*  LCD-displayet har en RAM-adresse knyttet til alle displayets
 20 x 4 karakter-positioner.
 Den værdi, der skrives til pågældende RAM, refererer til en
 bitmønstertabel.

 Heldigvis er det lavet sådan, at hvis der fx skrives værdien
 65 decimal, eller 41 hex, som i ASCII-tabellen svarer til et A,
 vil der skrives et A på displayet.

 I ASCII-tabellen er der ikke bit-mønstre for de danske karakterer.
```

```
 Men heldigvis er LCD-displayet's indbyggede Mønster-ROM udlagt som RAM
 på pladserne  0 til 7.

 Dette gør det muligt, at uploade 8 selv-definerede mønstre til LCD-en.

 /Valle Thorø, d. 16/3-2013 & 4/6-2019
 */


#include <LiquidCrystal.h>      // include the library code:

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);  // rs, (rw), en, D4, D5, D6, D7 )

//----------------------------------------



void setup()
{
  upload_char();       // Kald Upload-funktion

  lcd.begin(20,4);
}

//---------------------------------

void loop()
{
  for (char i=0; i < 8; i++){     // eksempel, skriv de 8 uploadede karakterer
    lcd.print(char(i));
  }

  lcd.setCursor( 3, 1 );    // cursor til plads 3, 2. linje

  lcd.print("HTX S" );      // Skriv Sønderborg
  lcd.print( char(2));
  lcd.print("nderborg");

  lcd.setCursor( 2, 3 );
  lcd.print("Made by Valle");
  lcd.setCursor( 17, 3 );
  lcd.print(char(7));           // Skriv mønstret i RAM # 7

    while(1); // stop looping
}

//---------------------------------


void upload_char()       // Funktion, Upload selvdefinerede karakterer til LCD
{
  // Definer 8 karakter-mønster:

  byte custom[8][8] = {
    { B00000,B00000,B00000,B00000,B00000,B00000,B11111,B00000}, // ??
    { B00000,B00000,B11010,B00101,B01111,B10100,B11111,B00000}, // æ
    { B00000,B00001,B01110,B10101,B10101,B01110,B10000,B00000}, // ø
    { B00100,B00000,B01110,B00001,B01111,B10001,B01111,B00000}, // å
    { B01111,B10100,B10100,B11110,B10100,B10100,B10111,B00000}, // Æ
    { B00001,B01110,B10011,B10101,B11001,B01110,B10000,B00000}, // Ø
    { B00100,B00000,B01110,B10001,B11111,B10001,B10001,B00000}, // Å
    { B11111,B10101,B10101,B01110,B01110,B00100,B00100,B00000}  // VT
  };
```

```
  //  ?, æ, ø, å, Æ, Ø, Å, VT  er placeret i RAMadresse 0, 1, 2, 3, 4, 5, 6, 7

  for (int i=0; i < 8; i++){     // Upload 8 selvdef. karakterer til LCD.
    lcd.createChar(i, custom[i]);
  }
  lcd.clear();     // nødvendig efter upload
}
```

Top ↑

Karaktergeneratorer

Se evt.: https://maxpromer.github.io/LCD-Character-Creator/
Eller https://omerk.github.io/lcdchargen/
https://www.microcontroller-project.com/making-custom-characters-on-16x2-lcd.html

Søg fx: *lcd display character generator*