Digitala projekt

Elektro- och informationsteknik

Digitala projekt (I)

- VT2 konstruktionsarbete i projektlabb
- 9 hp motsvarar 6 veckor heltid/student!
- Godkännande; U, G
- Gruppstorlek; 3-4 studenter

- Lektioner; 1 obligatoriska vardera 2 timmar
- Laborationer ; 1 obligatoriska 4 timmar
- Projekt; Under VT2, Dygnet runt, kortaccess

- Lärare ; Christoffer Cederberg Rum E:3150G
  - Bertil Lindvall, kursansvarig Rum E:3132A

Syfte och mål

Syfte

- Syftet med kursen är att illustrera industriellt utvecklingsarbete

Mål

- analysera och beskriva system av låg och medelhög komplexitet
- testa och felsöka en konstruktion på ett systematiskt sätt
- söka upp och tillgodogöra sig relevant information

Färdighet

- realisera digitala system av låg och medelhög komplexitet
- driva ett projekt framåt till en fungerande prototyp
- formulera sig muntligt och skriftligt

Kursens upplägg, föreläsningar

Föreläsning 1:

Introduktion, Gruppindelning, Processor Atmega 1284, skissa på ett enkelt projekt.

Laboration: Bli bekant med utvecklingsverktyget Atmel Studio 7

Hemsidan med all information.

https://www.eit.lth.se/index.php?ciuid=1249&L=0

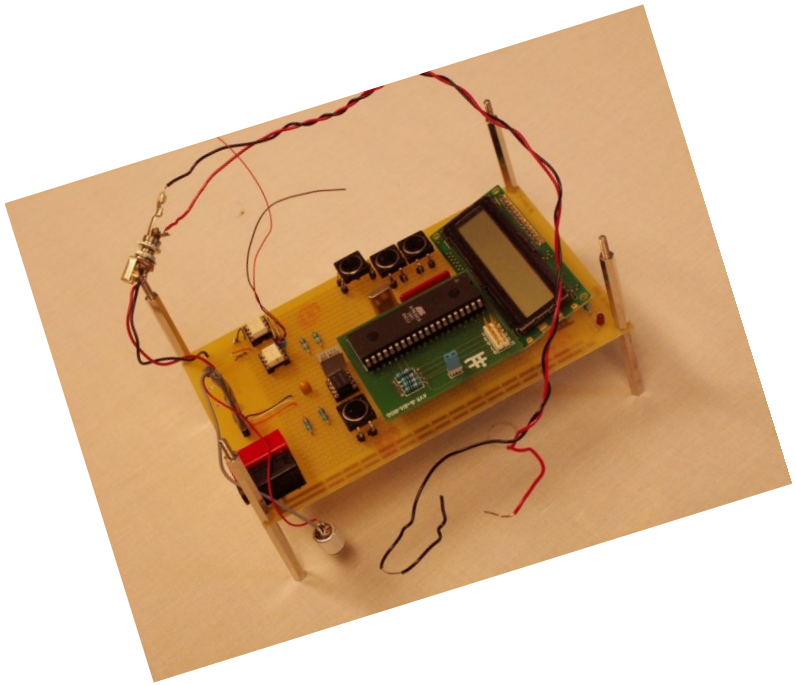
Vad är målet?

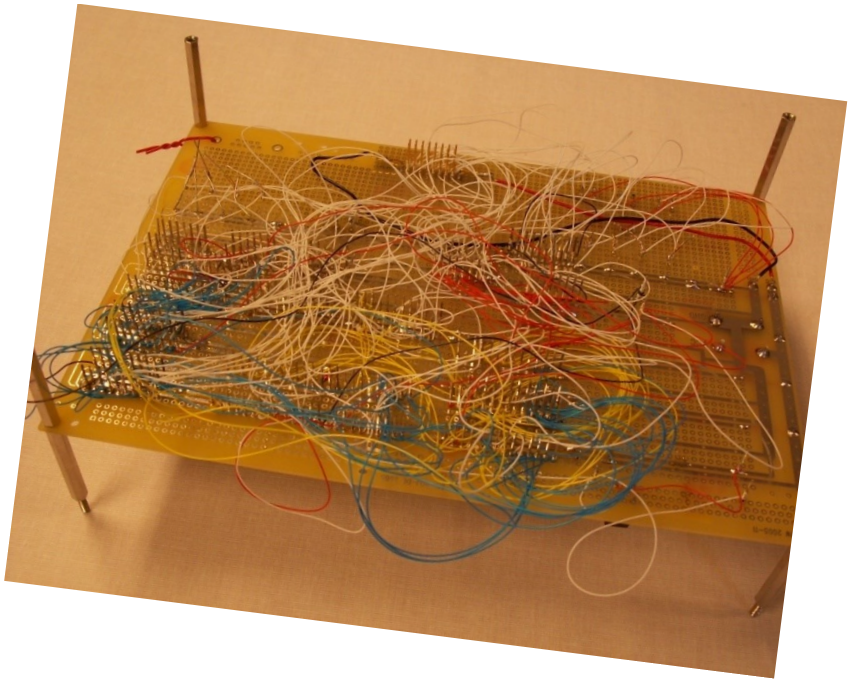

TELIA DIREKT


Brooklyn Bridge-City Hall Station
Nassau St
Park Row
Church St
140
yards
Broadway
0.25 M
1.30 min
10:11 am
10:12 am
31 mph
tomtom

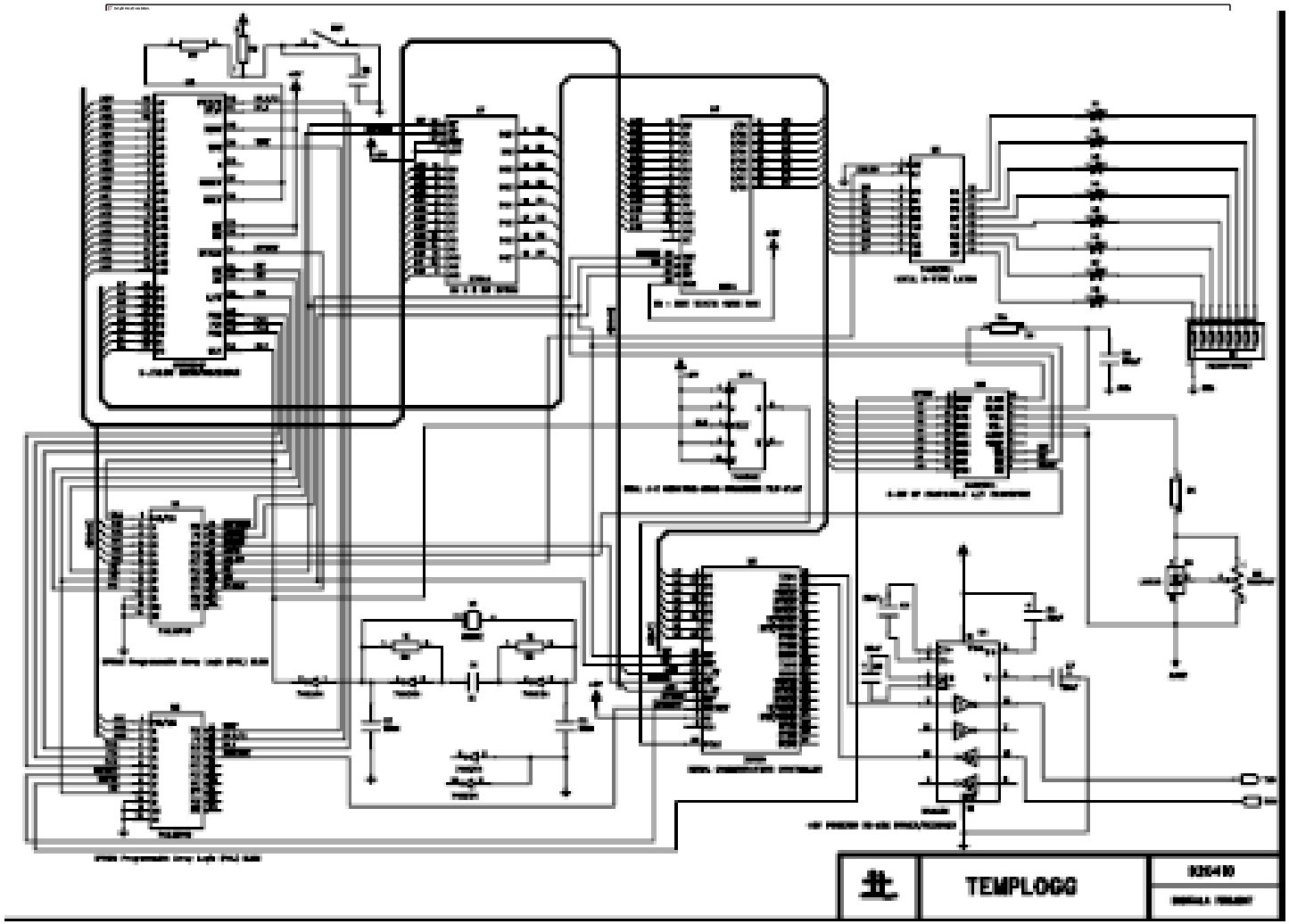

radio controlled
24.7
32
16.0
38
10 13.0
14:03
23.10.TU

Hur kommer vi dit?

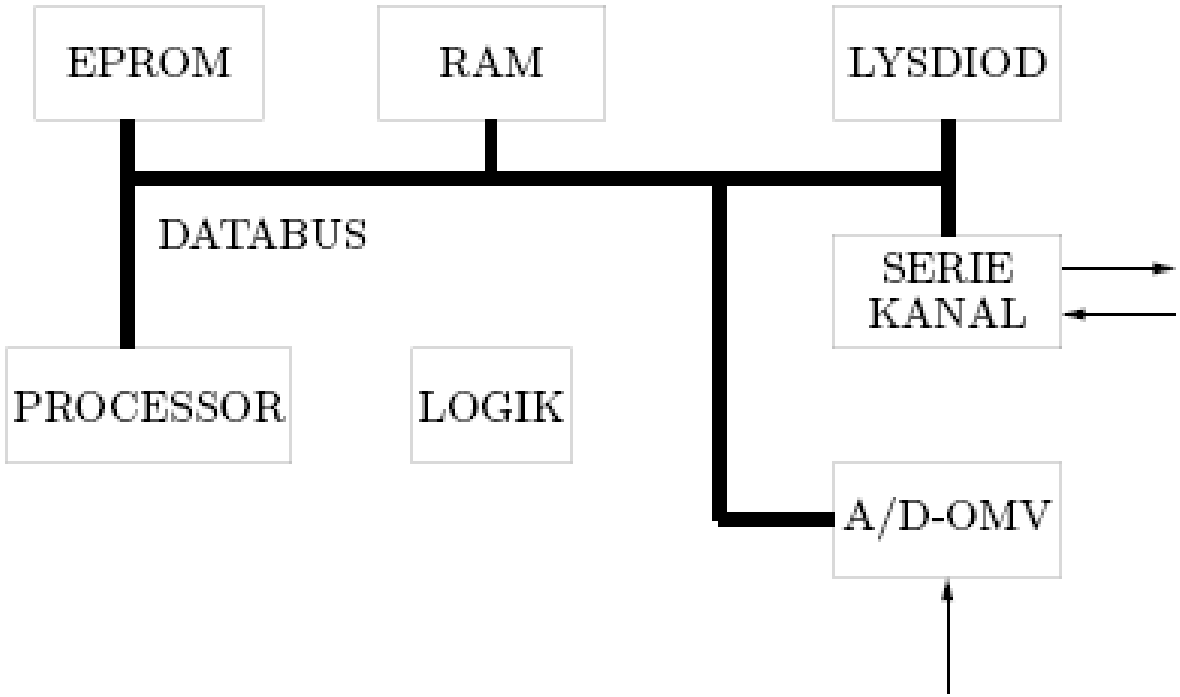




Ritning



Blockschema



Processor AVR 1284 /16 /32

- Datablad till AVR mega 16

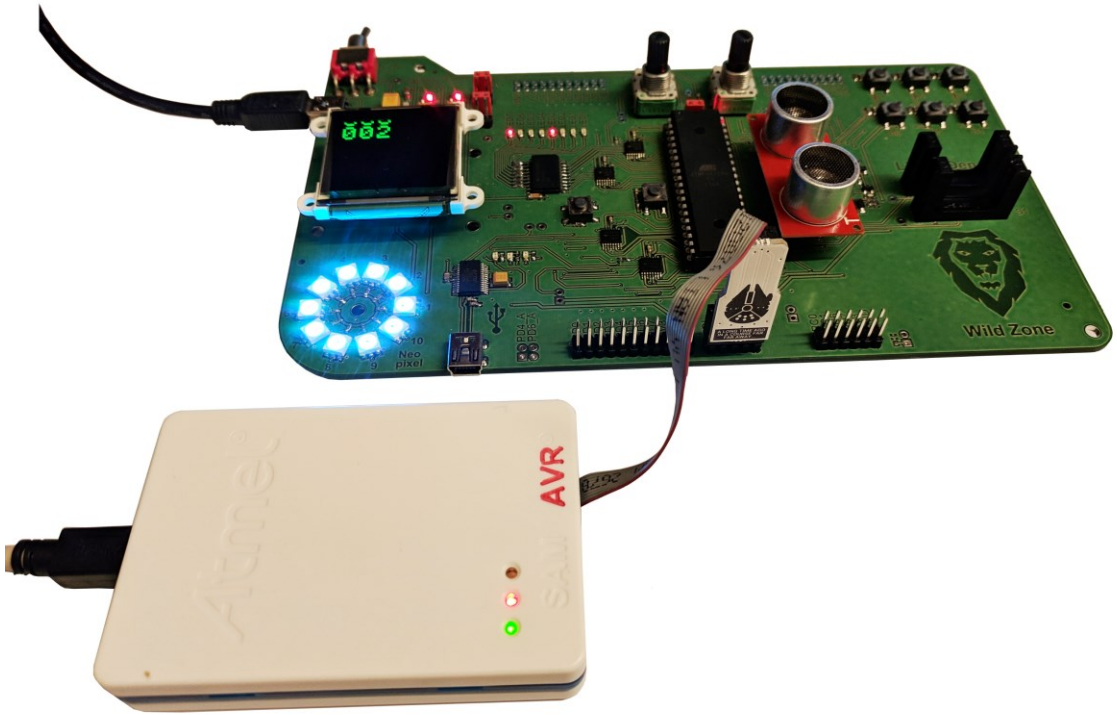
Sammanfattning av funktioner:

Klockfrekvens: 0- 16 MHz

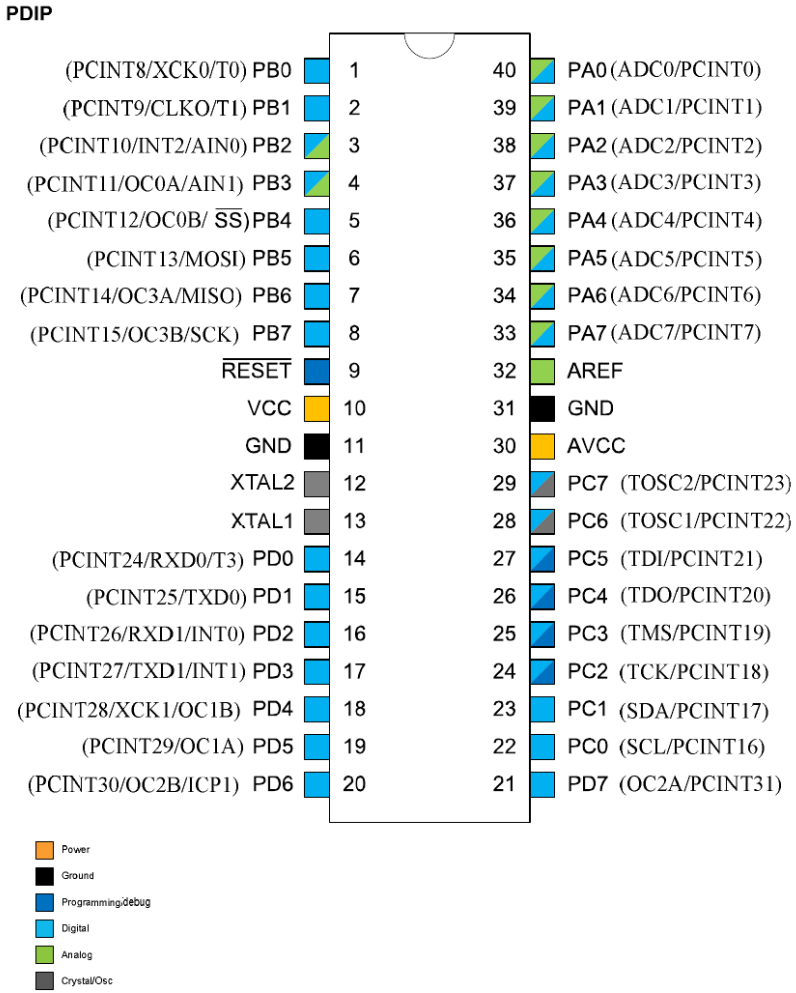
Minne: 16k Programmerbart ROM (BIOS)
1k RWM
512 byte Eeprom (USB)

I/O : 32 bitar in/ut (programmerbara pinnar)
8 kanaler Analoga ingångar (A/D-omvandlare)
2 st. 8-bitars Timer
1 st. 16-bitars Timer
seriell kanal USART
seriell SPI
JTAG felsökning
Avbrott (extern och inter)

Labbkort med JTAG debugger

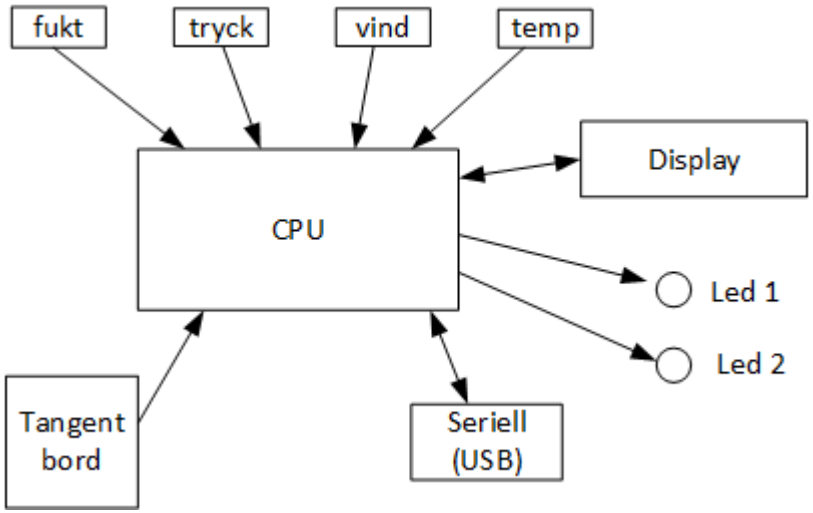


40 pin DIP AT1284



Projekt

- En enkel väderstation



- Vad består CPU av?

Övriga komponenter

Exempelvis Temperatur;

LM 35 vs LM 335

Display;

LCD modul

På liknande sätt för övriga komponenter

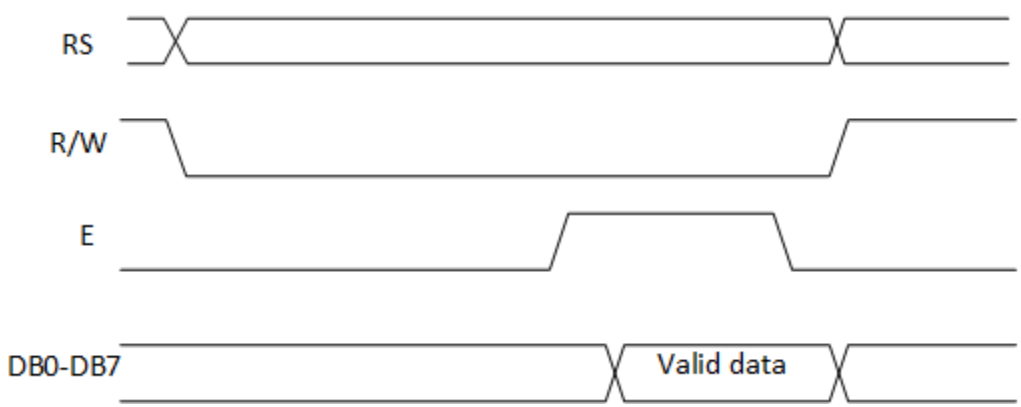
Alfanumerisk display Sharp Dot-Matrix

- Pinkonfiguration

6. Interface pin description

Pin no.	Symbol	External connection	Function
1	Vss	Power supply	Signal ground for LCM
2	V_{DD}		Power supply for logic for LCM
3	V_0		Contrast adjust
4	RS	MPU	Register select signal
5	R/W	MPU	Read/write select signal
6	E	MPU	Operation (data read/write) enable signal
7~10	DB0~DB3	MPU	Four low order bi-directional three-state data bus lines. Used for data transfer between the MPU and the LCM. These four are not used during 4-bit operation.
11~14	DB4~DB7	MPU	Four high order bi-directional three-state data bus lines. Used for data transfer between the MPU
15	LED+	LED BKL power supply	Power supply for BKL
16	LED-		Power supply for BKL

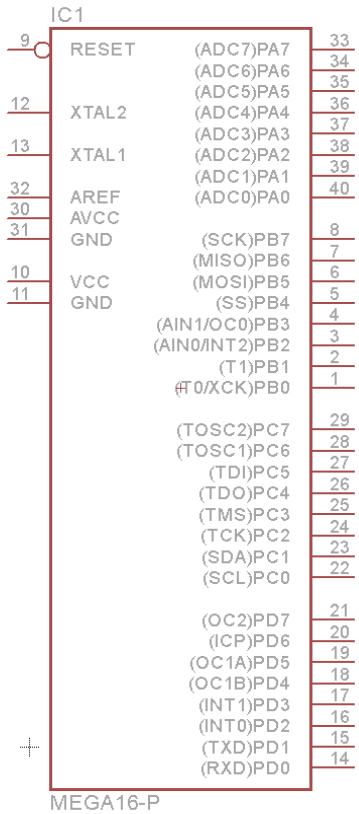
- Timing

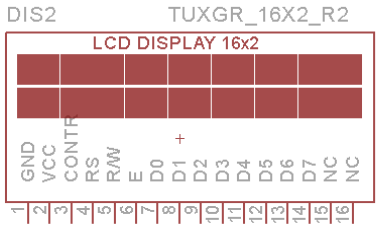


- Instruktioner

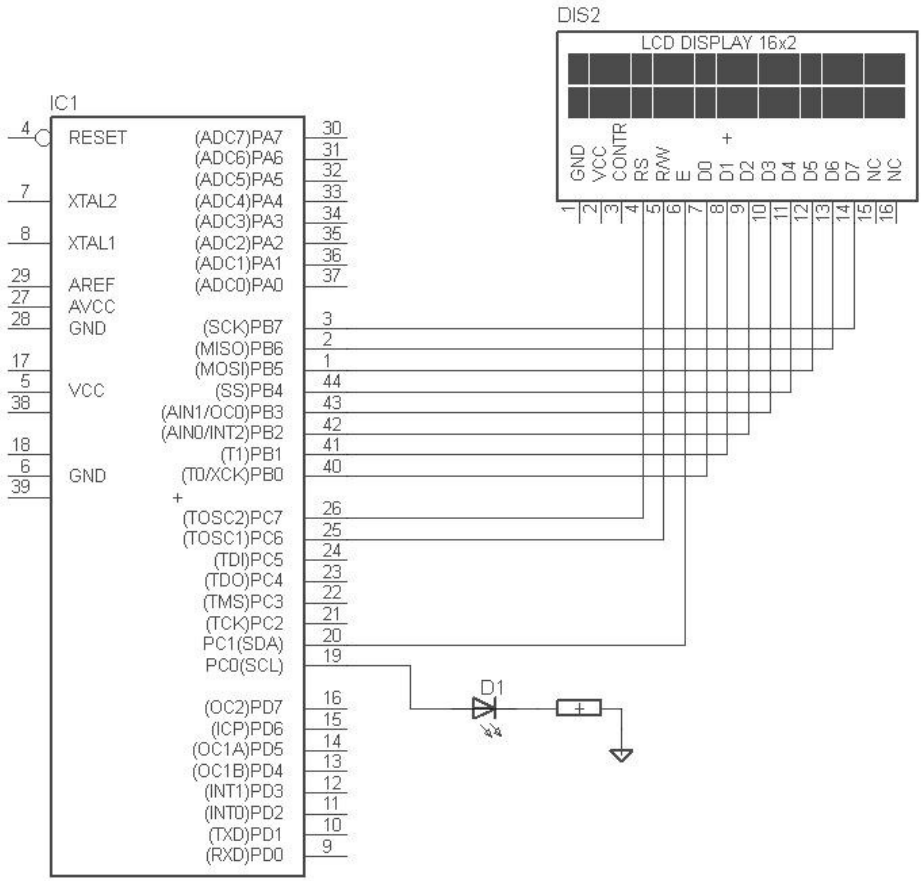
Command	Code										Description	Execution Time
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears the display and returns the cursor to the home position (address 0).	82μs~1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (address 0). Also returns a shifted display to the home position. DD RAM contents remain unchanged.	40μs~1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and enables/disables the display.	40μs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Turns the display ON/OFF (D), or the cursor ON/OFF (C), and blink of the character at the cursor position (B).	40μs
Cursor & Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing the DD RAM contents.	40μs
Function Set	0	0	0	0	1	DL	N$	F	*	#	Sets the data width (DL), the number of lines in the display (L), and the character font (F).	40μs
Set CG RAM Address	0	0	0	1	A_{CG}						Sets the CG RAM address. CG RAM data can be read or altered after making this setting.	40μs
Set DD RAM Address	0	0	1	A_{DD}							Sets the DD RAM address. Data may be written or read after making this setting.	40μs
Read Busy Flag & Address	0	1	BF	AC							Reads the BUSY flag (BF) indicating that an internal operation is being performed and reads the address counter contents.	1μs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	46μs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	46μs
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift. S/C= 1: Display shift S/C = 0: cursor move R/L= 1: Shift to the right. R/L= 0: Shift to the left. DL = 1: 8 bits DL = 0: 4 bits N = 1: 2 lines N = 0: 1 line F = 1: 5x10 dots F = 0: 5 x 7 dots BF = 1: Busy BF = 0: Can accept data *# Set to 1 on 24x4 modules* *$ With KS0072 is Address Mode.*										DD RAM: Display data RAM CG RAM: Character generator RAM A_{CG}: CG RAM Address A_{DD}: DD RAM Address Corresponds to cursor address. AC: Address counter Used for both DD and CG RAM address.	Execution times are typical. If transfers are timed by software and the busy flag is not used, add 10% to the above times.

Processor, display och led





Koppling AVR display och LED



Programmera AVR

Programspråket vi använder är C. Ett maskinnära effektivt språk som genererar effektiv maskinkod.

C-kod (maskinoberoende) kompileras till assembly-kod (processorberoende)

Exempel n= n & 10 -> ANDI r18,$10 Bit-och med register r18 och konstant 10

Assembly-kod assembleras till **maskin-kod** (ettor och nollor)

Exempel: 0111 0001 0010 0000

Programmera i C

- Programmera_i_C.pdf

- Bokref: The ansi C programming language
  ISBN 0-13-110362-8

AVR-libc

- Avr-libc = standard C bibliotek för AVR-GCC
- http://www.eit.lth.se/fileadmin/eit/courses/edi021/Avr-libc-2.0.0/group__stdiodemo.html

```
#include <avr\io.h>
#include <avr/interrupt.h>

int MyValue;

 ISR(SomeVector_vect)
 {
    MyValue++;
 }

 int main(void)
{
SetupInterrupts();   //bland annat sei();

while (MyValue == 0);  // Wait for interrupt
 TurnOnLED();
}
```

C-program

```
#include <avr/io.h>
unsigned char num;  //global variabel

void main(void)
{
int kalle;    //lokal variabel

  num |= (0b00000100);  /bit 2 sätts till ett
  num &= ~(0b00001000); /bit 3 sätts till noll
while(1)
      {
            PORTA = num;
            num++;
      }
   return;
}
```