Intro til uC.

Denne intro til Micro controllere er oprindeligt skrevet til Atmels AT89C2051, der er baseret på en gammel 8051-familie fra INTEL. Den man køber nu hedder AT89C4051. Det er en opgraderet udgave med dobbelt så meget plads til kode-hukommelse. Den kan indeholde 4 K kode (imod kun 2 K)

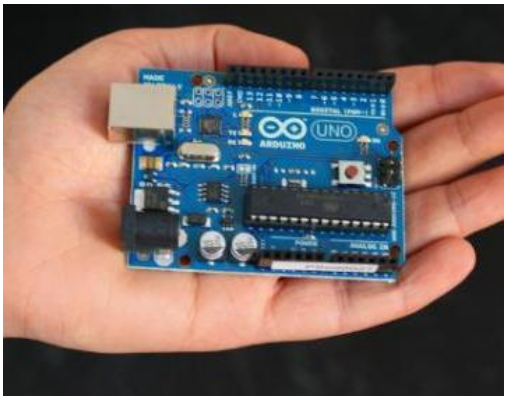
Prisen er ca. 11 kr. pr stk. @ Cypax.

/ Valle

Her introduceres den gamle 8051-familie, - og assemblerprogrammering.

Men senere skal vi programmere i ”C”, på Atmels AVR-processor, ATmega328, der sidder i Arduino-kittet.

Denne processor er en nyere udgave af de tidligere Controllere, men introen kan sagtens bruges idet principperne og indmaden i de forskellige uCér er nogenlunde ens.



Assemblerprogrammering er ” Low Level ” sprog. Her skal man selv tage sig alt, huske en hel masse, fx bestemme hvor i RAMén data, dvs. variable - skal ligge, osv.

Dvs. man selv skal have styr på organiseringen af RAM og også ROM. (Flash-ROM)

I ”C”-verdenen skrives programkoden i et højniveau-sprog, og kildeteksten skal Compileres,

Compileren tager sig så af hvilke Ram-registre, der bruges til variable, og hvor i ROM-en programmet placeres osv.

Microprocessor vs. Microcontroller

Det hele begyndte omkring 1971, hvor Intel udviklede det, der blev kaldt en Mikroprocessor.

Et mikroprocessor-system bygges typisk op af flere separate IC-er.

Et system har en separate beregnings-IC, en separat program-hukommelses-IC, IC-er til RAM-hukommelse, og andre IC-er til at skabe kontakt med verdenen udenfor.

Det bygges op på et printkort.

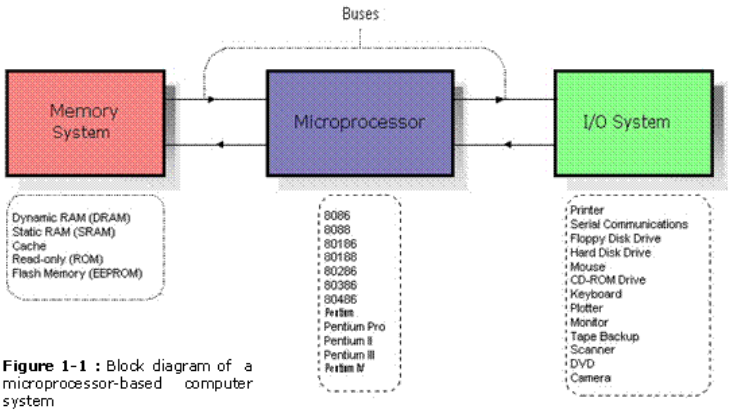
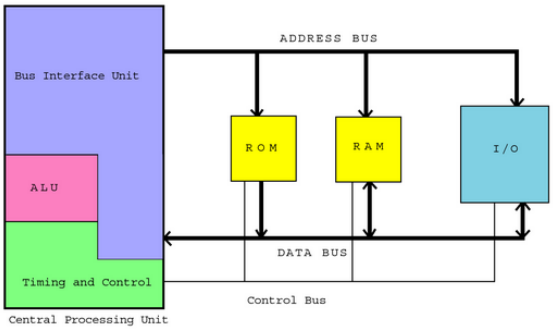


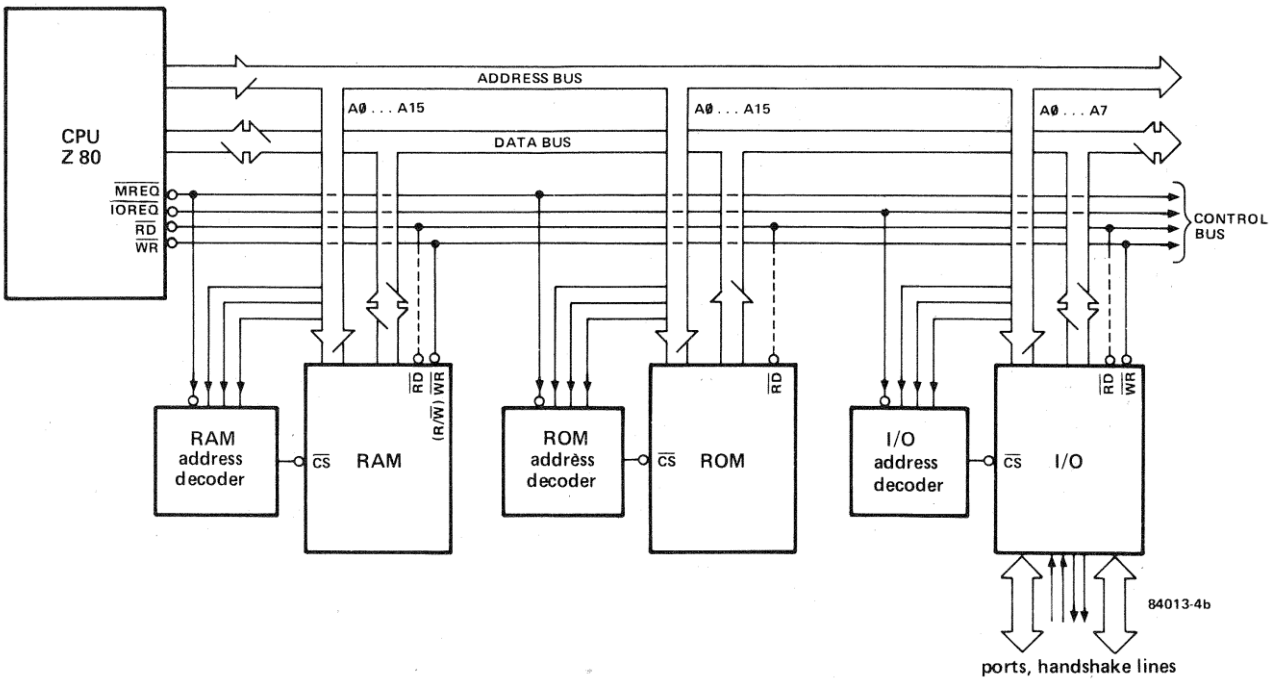
Figure 1-1 : Block diagram of a microprocessor-based computer system

https://www.byclb.com/TR/Tutorials/microprocessors/ch1_1.htm

Enhederne skal forbindes sammen så processoren kan få fat i de rigtige enheder og data.
Det sker via parallelle forbindelser, kaldet for ” Busser ”

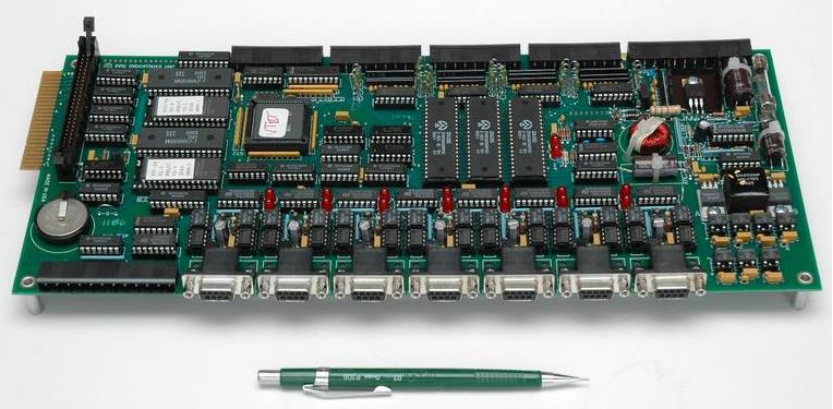


Eksempel på et microprocessor-system:



Hvorfor bruges 16 adresseledninger i adressebussen ???

Her ses et typisk microprocessor-board:



Microcontrollere

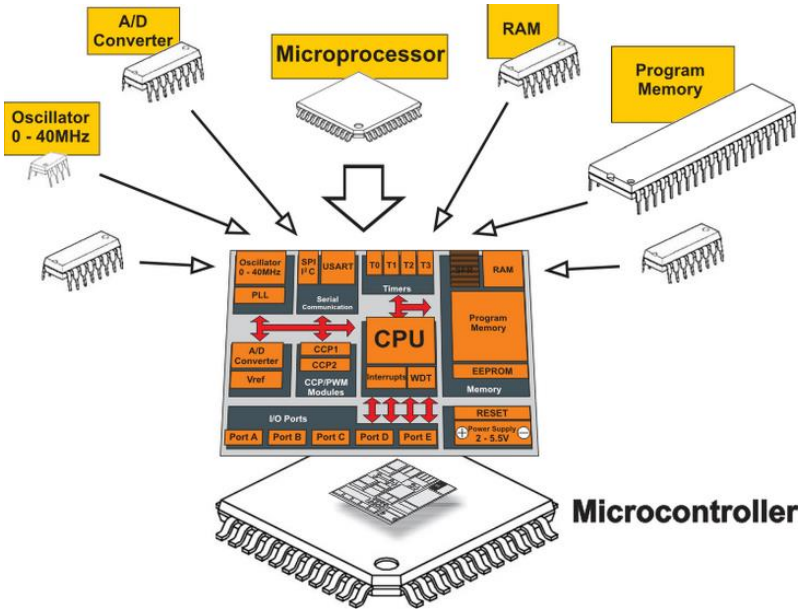
Udviklingen af uC stammer også helt tilbage fra 1970-erne, og er baseret på microprocessorerne.

Det var Intel og TEXAS, der var først. De første kredse var vist kundespecificerede IC-er, beregnet til lommeregnere.

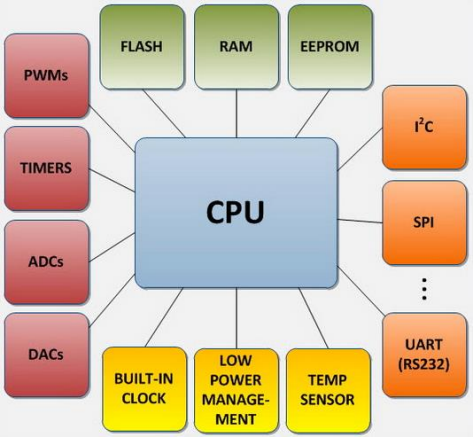
8051-microcontrolleren blev introduceret af Intel i 1980, og er en af de mest populære microcontrollere. Den bruges stadig i forskellige varianter, og anses at være en af de længst levende controllere.

En microcontroller har på selve chippen alle de enheder, der er nødvendige for at den kan fungere.

Derved spares en mængde plads, og forbindelser til eksterne kredse.



http://maxembedded.com/2011/06/mcu-vs-mpu/

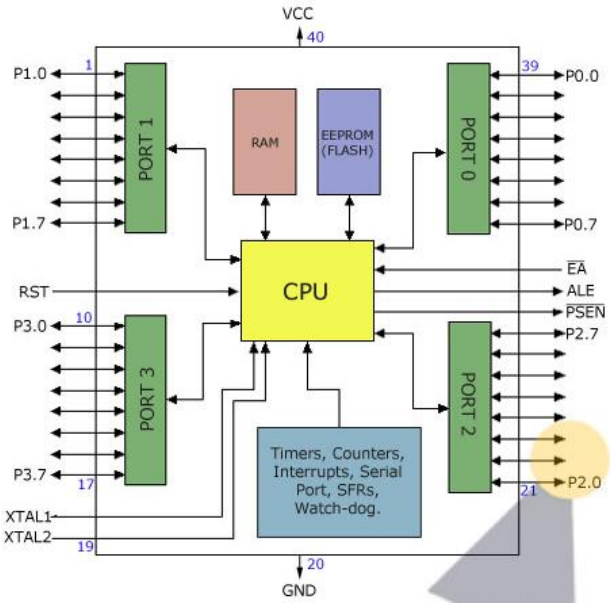


Her en anden skitse, der på blokniveau viser de enheder, der kan være stoppet ind i en uC.

Se kilde:

Her en skitse, hvor pins til omverdenen fra Controlleren er vist.

I hver port er der i realiteten en RAM-adresse, der gør udgangen høj, hvis tilsvarende bit i pågældende RAM er et 1-tal.



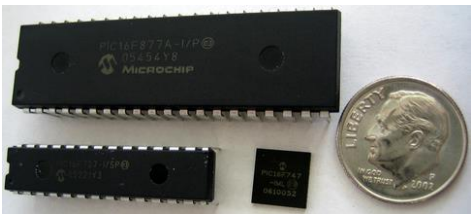
http://embeditknow.blogspot.dk/2012/07/microprocessor-vs-microcontroller.html

Der findes et hav af uC-familier fra forskellige fabrikker.

Se evt. liste: https://en.wikipedia.org/wiki/List_of_common_microcontrollers

Dem vi kan " lege med " skal jo være i et hus, vi kan håndtere, dvs. i DIL-huse.

Vi skulle gerne kunne lave print til eksamen, indeholdende en uC.

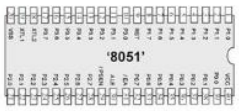


Her er vist forskellige uC familier:

8051
AVR
PIC
80196
8086
Z80 (1976)
ARM (Acorn RISC Machine)
68xx fra Motorola
PowerPC
OSV.

Microcontroller Families

- 68H12: Motorola 68H11, 68HC12, …
- 8051: Intel 8051, 8052, 80251,…
- PIC: Microchip PIC16F628, 18F452, 16F877, …
- AVR: Atmel ATmega128, ATtiny28L, AT90S8515,…

Nogle af de processorer, der anvendes i dag, er baseret på ældre versioner. Nogle har færre funktioner, nogle har andre funktioner end den oprindelige, og de fines i forskellige type huse osv.

Eksempel på afledte typer med forskellige delmængder, - eller typer med ekstra egenskaber.

Nogle fås som OTP, dvs.
One Time Programmable.

Andre kan genprogrammeres, osv.

Nyere versioner kan programmeres serielt, dvs. man kan In-Circuit-programmere dem, dvs. mens de sidder i printet.

Comparison of the 8051 Family Members

- ROM type
  - 8031 no ROM
  - 80xx mask ROM
  - 87xx EPROM
  - 89xx Flash EEPROM
- 89xx
  - 8951
  - 8952
  - 8953
  - 8955
  - 898252
  - 891051
  - 892051
- Example (AT89C51,AT89LV51,AT89S51)
  - AT= ATMEL(Manufacture)
  - C = CMOS technology
  - LV= Low Power(3.0v)

Den første processor, vi ser på, er en 8051-variant, en Atmel AT89C4051, evt. en AT89S8253 !!

Derefter går vi over i Arduino-verdenen.

Arduino uno burger en ATmega328P som er en AVR med 8-bit RISC arkitektur.

Det, der er rigtigt genialt, er at der på Arduino-boardet er en USB-chip, der kan skabe direkte adgang til PC-ens USB-port. Det betyder at man let kan overføre et microcontroller-program fra ens PC til IC-en.

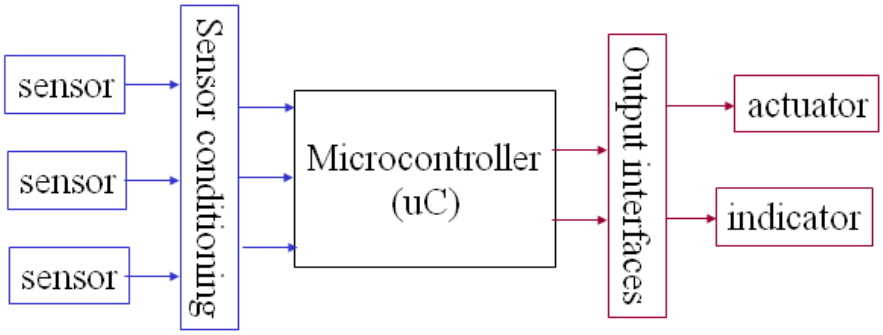
Ganske god intro: http://www.handsonresearch.org/2012/PDF/IntroductionToArduino.pdf

Begge uC-er er 8-bit som har været brugt så længe. I dag fås også 16 og 32 Bit controllere. Men 8 bit uC nægter at dø.

8 bit – passer fint med vores kendskab til Talsystemer:

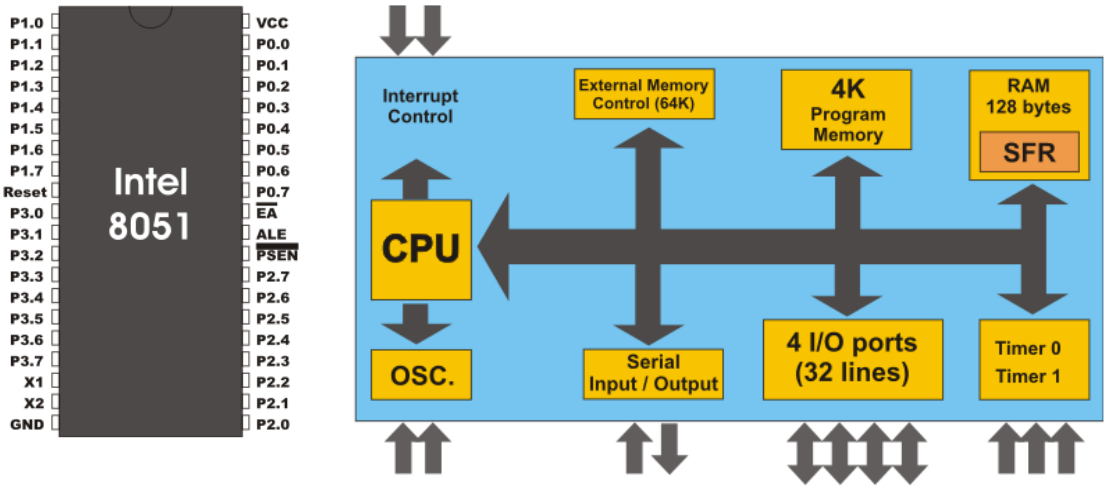
Det vi skal i gang med er at koble en uC sammen med eksterne enheder:

Generel Blok- diagram for et microcontroller system



Her ses konceptet i den originale 8051. Der er 4 8-bits porte, hvoraf nogle har flere funktioner.

Kilde: [1]

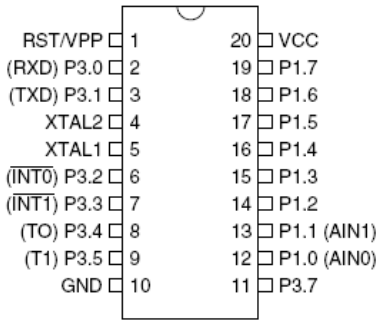


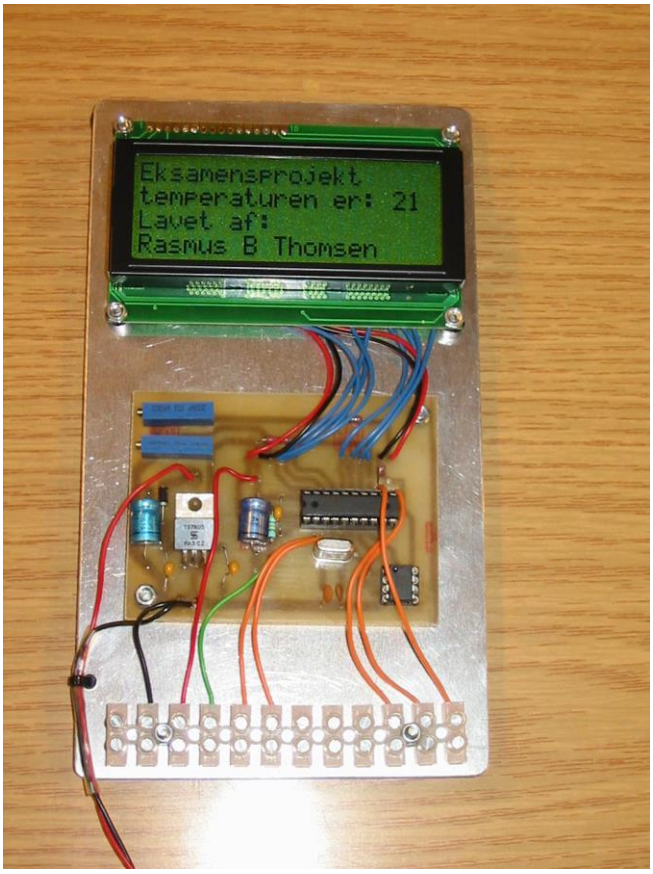
Den lille – og billige – uC vi her skal starte med er en AT89c4051, der er en delmængde af 8051:

AT-familierne blev oprindelig lavet af firmaet Atmel, men firmaet blev opkøbt af Microchip.

Jeg har til arbejdet med at lære programmering af uC-er lavet nogle kits. Når et program er lavet, skal det overføres det til uC-en med en brænder, og kan herefter testes i mine kits.

[1] http://www.mikroe.com/en/books/8051book/ch1/

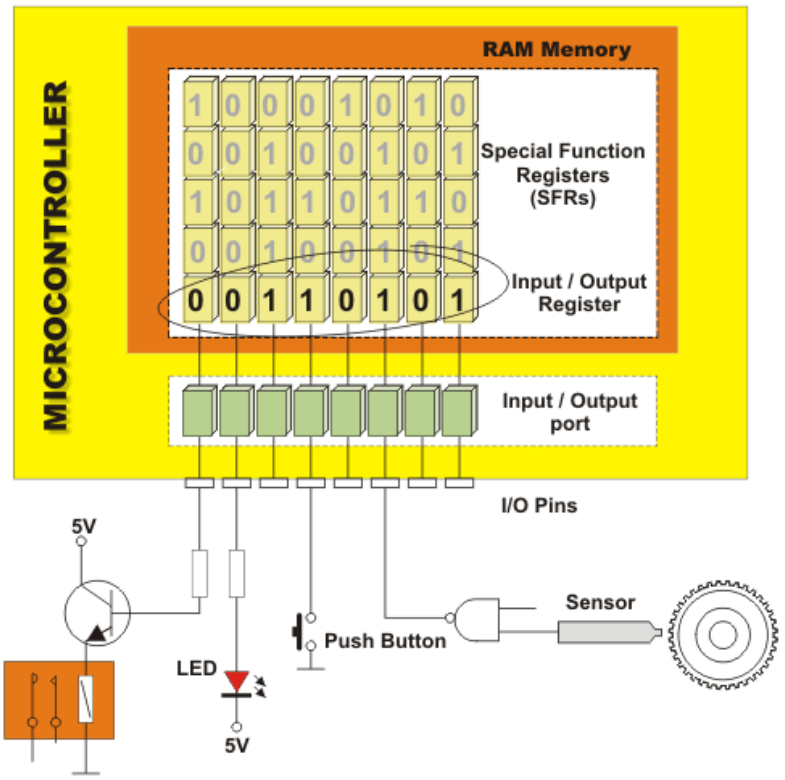




Vi har også nogle større uC-er fra samme familie. Med flere portben.

AT89C4051 fås også i en "storebror", der hedder AT89S8253

Den kan nogenlunde det samme, har blot flere I/O.

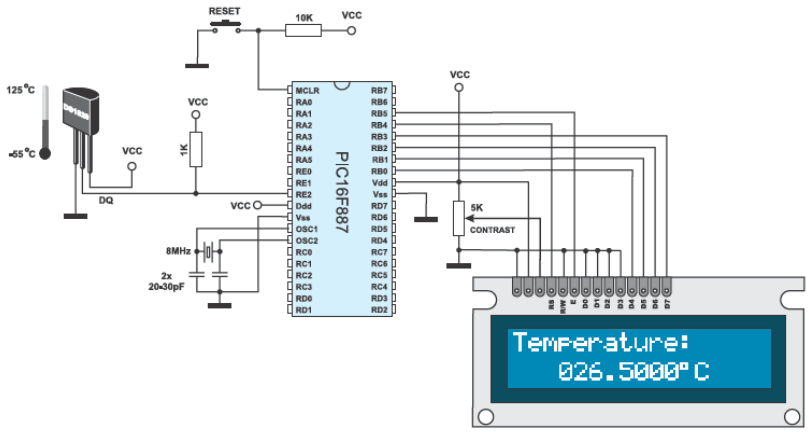


Microcontrollerens pins kan tage digitale signaller ind fra tilsluttede enheder, og også sende signaler ud.

0 eller 5 Volt.

Kontrolleret af et program lagt ind i controlleren.

http://www.mikroe.com/en/books/8051book/ch1/



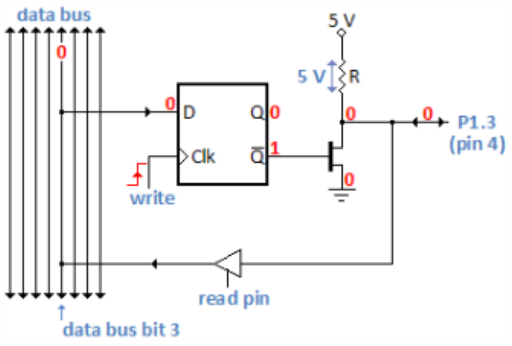
Her vist et eksempel, men bemærk, processoren er fra en anden familie.

Opbygning af en port-ud / indgang.

Her sendes et 0 ud. Udgangstransistoren i 8051-familien kan kun synke strøm.

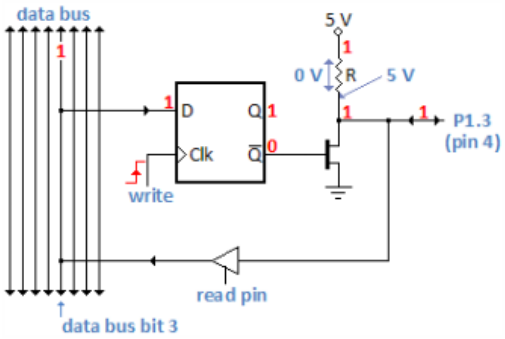
Bemærk: Udgangene er Sink Only.

Open Collector, eller her vist som Open Drain.



Her er udgangen høj.

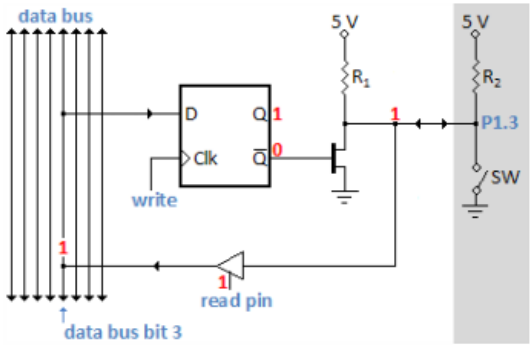
FF'en holder udgangen på 1 indtil programmet sender noget andet ud !!



Setting Pin Three on Port One

Og endelig en portpin brugt som indgang.

Kilde:
http://www.scribd.com/doc/15924903/EdSim51s-Guide-to-the-8051

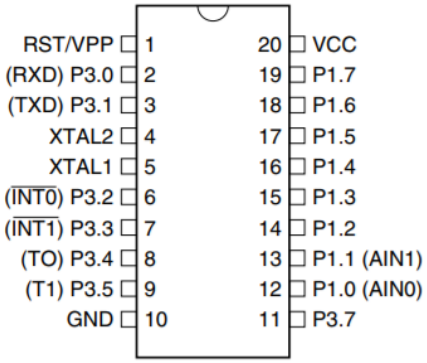


Altså kan uC'ens pins bruges både som udgang og indgang.

Men bemærk, at pga. opbygningen af denne type uC, skal en indgang normalt være høj, og en switch skal trække den lav!! Dvs. der skal bruges en extern Pull Up modstand.

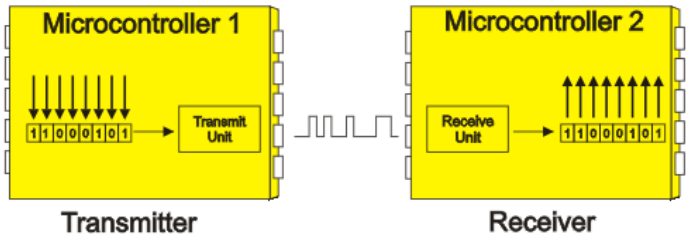
Nogle pins har flere funktioner:

Både i den originale 8051 og i vores delmængde, dvs. AT89C4051, der kun har port P1 og P3.

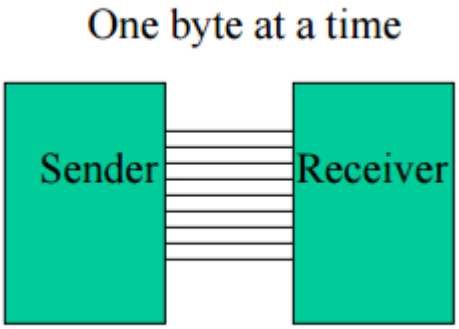


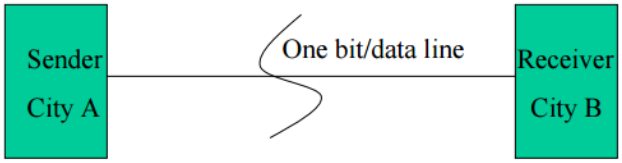
En af de store styrker er, at man kan koble uC-er sammen serielt, og sende data imellem dem.

To microcontrollere er her vist forbundet sammen via seriel transmission.



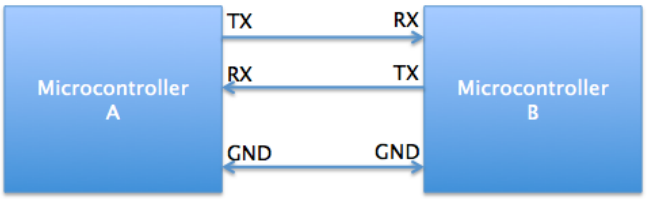
http://www.mikroe.com/en/books/8051book/ch1/





Data kan sendes med alle 8 bit ad gangen, - men det kræver færre portpins at sende data på 1 linje, - men så kun sende de 8 bit 1 ad gangen.

Parallel transmission bruger for mange pins !!



Microcontrollere har indbygget UART, med hhv. sender og modtager.

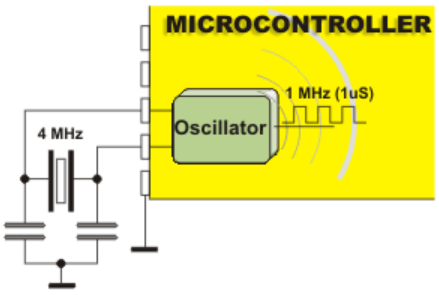
Det står for:

Universal Asyncron Reciever / Transmitter

Dvs. der kan sendes begge veje.

En uC skal have en clock-frekvens for at arbejde.

Typisk bruger vi 12 MHz
Eller 11,059 MHz.



http://www.mikroe.com/en/books/8051book/ch1/

Opsamling: Samlede Specifikationer for AT89C4051:

Koster ca. 10 kr.

8-bit Microcontroller

4K Bytes Flash ROM and 128 Byte RAM

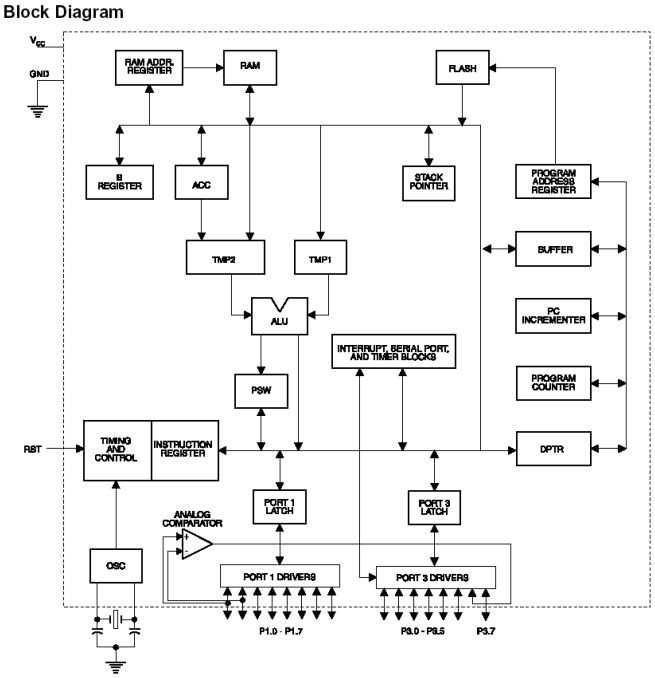
2 Ports (15 bit) Port 1, and Port 3

UART (Serially communication)

2 x Timers / Counters

Help: http://www.8052.com/
http://www.mikroe.com/en/books/8051book/ch1/

Blokdiagram over indmaden i AT89C4051:



ROM	Et Program gemmes i Flash ROM.
RAM	Arbejdshukommelse, variable
2 Porte 8 + 7 bit	Pins til verden udenfor – ligger i RAM
Program Counter / Pointer	– har styr på, hvor langt man er nået i program-afviklingen.
Stack Pointer	Bruges i forbindelse med interrupts. Når der hoppes til en subrutine, skal programmet kunne finde tilbage igen.
Timer / Counter	Et antal. Kan bruges til at tælle. Evt. eksterne pulser, eller krystallets pulser. (til timer-funktioner)
UART	Til serial transmission. (Universal Asyncron Reciever / Transmitter)

Procedure for programmering:

Program skrives i "C" eller Assembler.

→ kildeteksten skal oversættes til en HEX kode - fil,

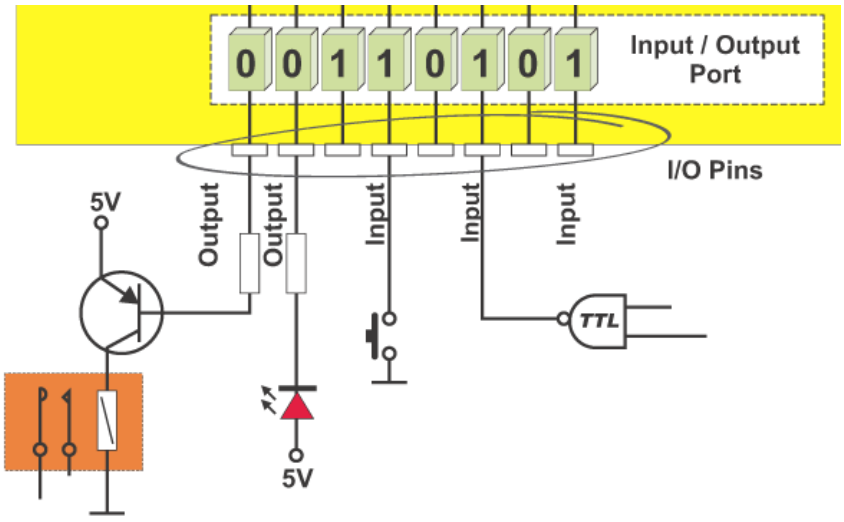
→ som skal overføres til uC.

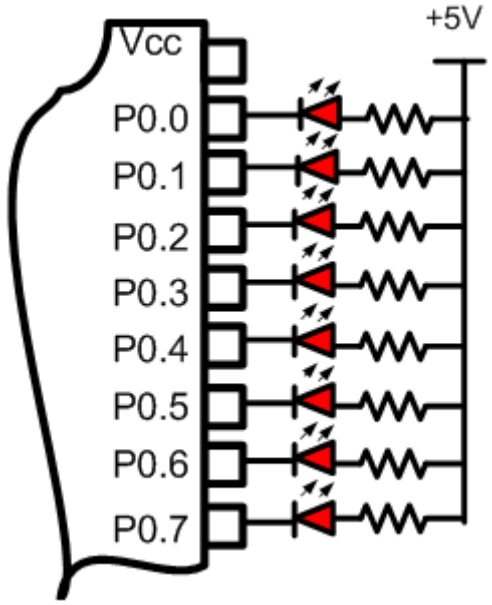
- - - - - - - - - - -

Ideen er så at skrive et program, der kan kontrollere handlinger på dens udgange, fx afhængig af inputsignaler.

Eksempler:

Forskellige input & output "der registrerer eller kontrollerer "





Eksempel på kredsløb !!

Bemærk, at denne type uC kun kan synke strøm.

Data og Programhukommelsen:

Se lidt på de interne Memmory: RAM & ROM

ROM er lavet i Flash, som er re-programmerbar.

Registre og RAM: Et register er en ram-adresse på 8 bit.

User RAM

Op til 7Fh=127d	
Etc	
0Bh	
0Ah	
09h	
08h	

07h	R7
06h	R6
05h	R5
04h	R4
03h	Også R3
02h	Også R2
01h	Også R1
Ram Adr 00h	Også R0

Ram-adresser fra 00h til 07h kan også kaldes R0 til R7.

Ramadresser fra 20h til 30h kan bruges til Bitflag.

Derfor brug helst kun RAM fra adresse 0 til 7, og fra 30h og opefter !!

Eksempler på 1 bit flag.

Flagene har numre fra 0 til 127, eller 0 til 7F Hex.

Eks:

Jb 29, label

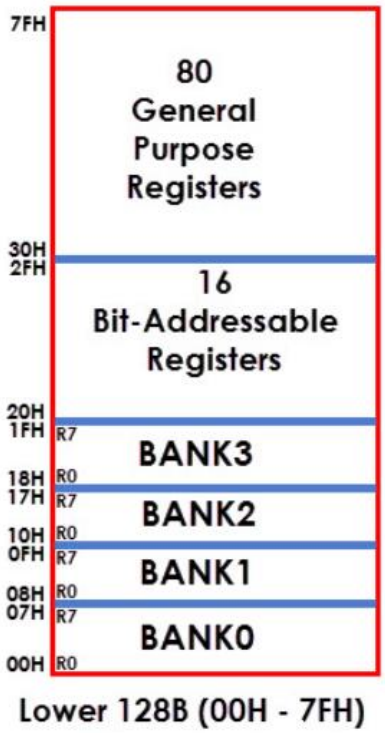
Setb 2Dh

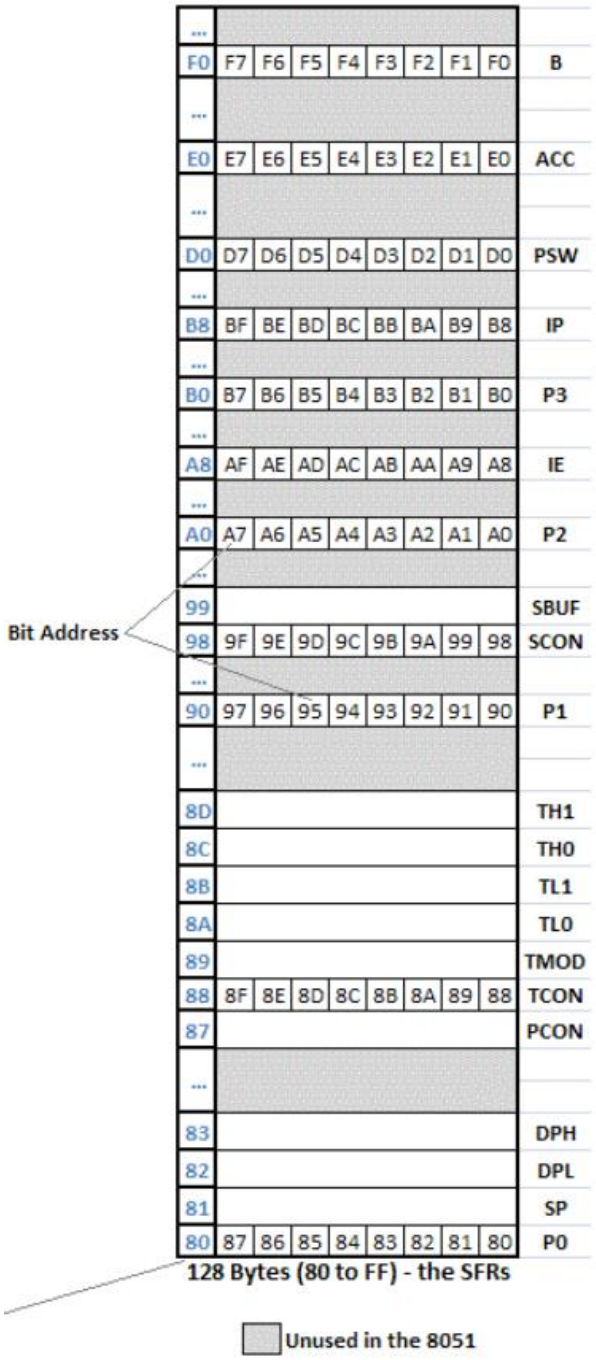
Clr 0 ; Svært at huske hvilke bitflag, der ligger hvor, derfor giv den navne!!

Det er dem, der bruges til Boolean variable i ”C”

7F	General Purpose RAM							
30								
2F	7F	7E	7D	7C	7B	7A	79	78
2E	77	76	75	74	73	72	71	70
2D	6F	6E	6D	6C	6B	6A	69	68
2C	67	66	65	64	63	62	61	60
2B	5F	5E	5D	5C	5B	5A	59	58
2A	57	56	55	54	53	52	51	50
28	4F	4E	4D	4C	4B	4A	49	48
28	47	46	45	44	43	42	41	40
27	3F	3E	3D	3C	3B	3A	39	38
26	37	36	35	34	33	32	31	30
25	2F	2E	2D	2C	2B	2A	29	28
24	27	26	25	24	23	22	21	20
23	1F	1E	1D	1C	1B	1A	19	18
22	17	16	15	14	13	12	11	10
21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00
1F	Register Bank 3							
18								
17	Register Bank 2							
10								
0F	Register Bank 1							
08								
07	Default Register Bank (Bank 0)							
00								

RAM





Der er indbygget nogle RAM-adresser over adresse 7Fh

De bruges til Special Functions.

En af RAM-adresserne er forbundet ud til port P1, en anden til port P3.

Se mere herunder.

SFR (Special Function Registre)

Alle de forskellige enheder der er indbygget, har tilknyttet RAM-adresser !!

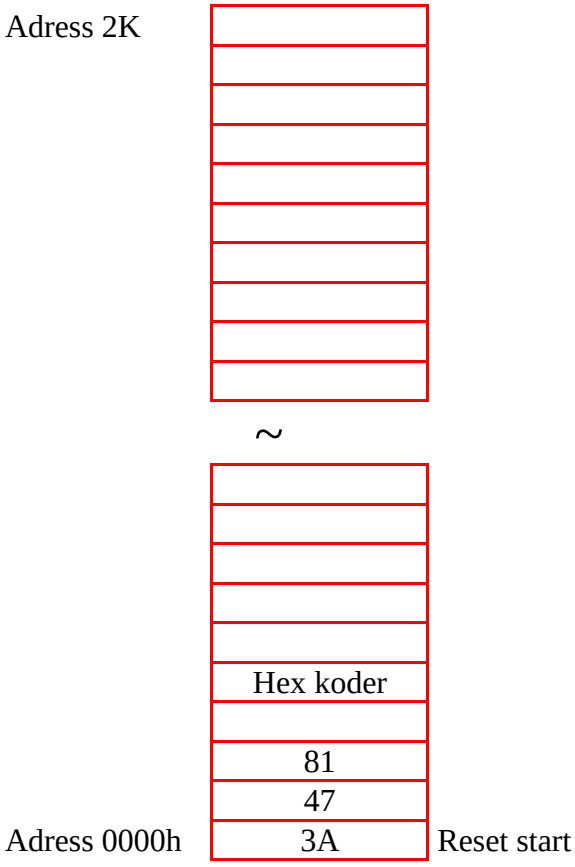
Programhukommelse i FLASH ROM

Det program, der skal afvikles, gemmes på chippen i Flash ROM. Det betyder, det kan slettes og genprogrammeres mange gange.

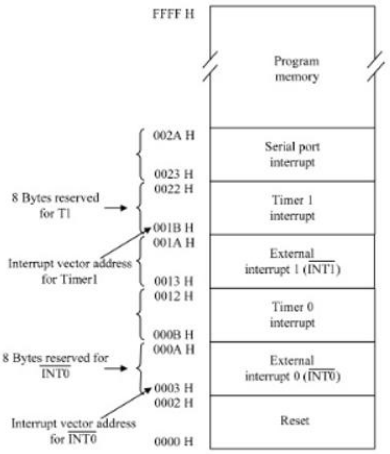
Vores AT89C4051 har plads til 2 kByte program-hukommelse.

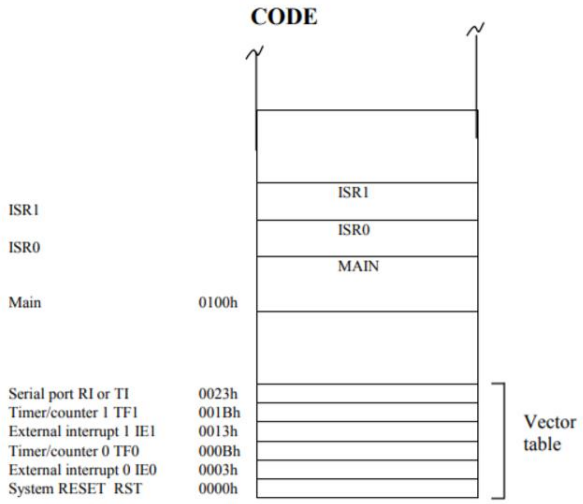
Egentlig nærmere 2048 Bytes. ($2^x = 2048$)

Et program består blot af et antal 8-bit tal, som fungerer som koder, processoren forstår.



Interrupt vektorer og deres default adresse i ROM.





Adresse 0h: - er altid reset start. Her kan kode også starte hvis der ikke bruges interrupt i koden

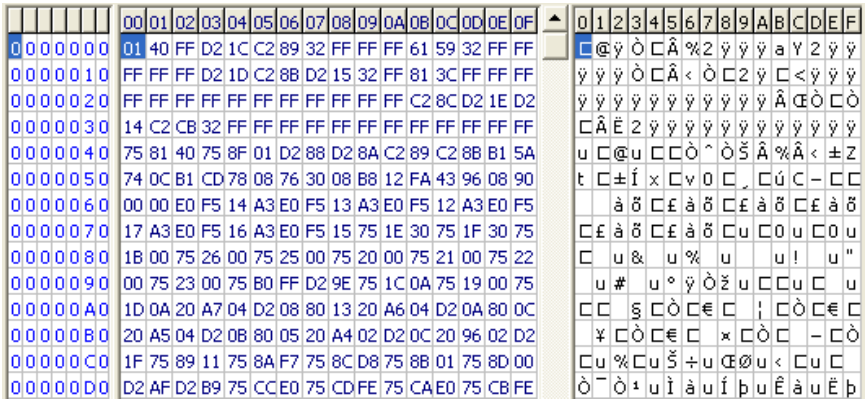
Ellers interruptvektorer fra adresse 03h til 2Fh:

Eks. På interrupt: I gang med at binde snørebånd, og så kommer der en fodbold imod en.

Fra adresse 30h: Kode og subrutiner, evt. til interrupts.

Et eksempel på et program.

Vist som Hex.

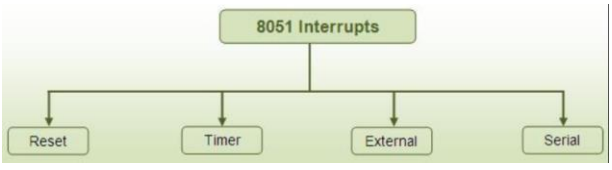


	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000000	01	40	FF	D2	1C	C2	89	32	FF	FF	FF	61	59	32	FF	FF
0000010	FF	FF	FF	D2	1D	C2	8B	D2	15	32	FF	81	3C	FF	FF	FF
0000020	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	C2	8C	D2	1E	D2
0000030	14	C2	CB	32	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
0000040	75	81	40	75	8F	01	D2	88	D2	8A	C2	89	C2	8B	B1	5A
0000050	74	0C	B1	CD	78	08	76	30	08	B8	12	FA	43	96	08	90
0000060	00	00	E0	F5	14	A3	E0	F5	13	A3	E0	F5	12	A3	E0	F5
0000070	17	A3	E0	F5	16	A3	E0	F5	15	75	1E	30	75	1F	30	75
0000080	1B	00	75	26	00	75	25	00	75	20	00	75	21	00	75	22
0000090	00	75	23	00	75	B0	FF	D2	9E	75	1C	0A	75	19	00	75
00000A0	1D	0A	20	A7	04	D2	08	80	13	20	A6	04	D2	0A	80	0C
00000B0	20	A5	04	D2	0B	80	05	20	A4	02	D2	0C	20	96	02	D2
00000C0	1F	75	89	11	75	8A	F7	75	8C	D8	75	8B	01	75	8D	00
00000D0	D2	AF	D2	B9	75	CC	E0	75	CD	FE	75	CA	E0	75	CB	FE

I højre side er koderne så vidt muligt oversat efter ASCII-tabellen.

Interrupt-vektorer.

I processorverdenen arbejdes der ofte med interrupts.



Det kan fx optræde hvis man laver et stopur.
Her skal der ske noget bestemt hver gang der er gået 1/100 del af et sekund.

Interrupts kan stamme fra forskellige events

Det betyder, man i starten af et program skal igangsætte nogle interne tællere, der kan tælle krystallets frekvens. Og hvis et bestemt antal pulser er talt, er der jo gået en bestemt tid.

Det kan så indrettes således, at når tællerne har nået en bestemt værdi, udløses et interrupt.

Dvs. den igangværende programafvikling stoppes, og et andet – kort – del-program skal udføres.

(Der skal fx lægges 1 til 1/100 del sekunder, tjekkes for om værdien er blevet større end 9, osv.)

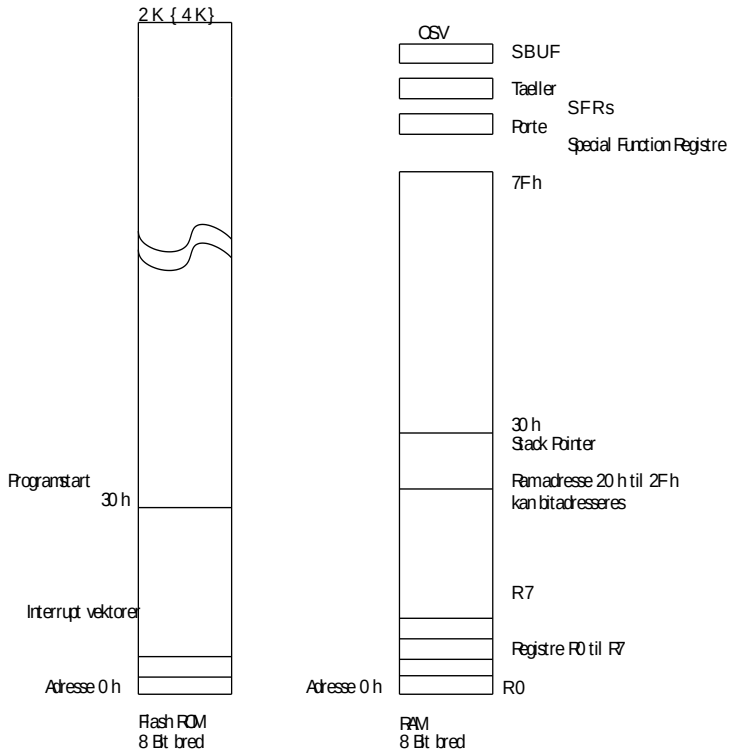
Den måde det er implementeret på, er, at processoren hopper til en bestemt adresse kaldet en interrupt-vektor.

Herfra kan der så hoppes videre til den programdel, der skal udføres hver 1/100 del sekund.

De forskellige vektorer er placeret i ROMén fra adresse 0003h til 002Fh.

Derfor, - hvis man benytter interrupts i sit program, skal man tage højde herfor.

Samlet Memory oversigt



Program gemmes af brænderen i ROM, Data gemmes af processoren i RAM.

uVision

Den IDE (Integrated Development Environment, Integreret udviklingsmiljø) vi bruger hedder uVision, fra firmaet **KEIL**

Det er et gratis evaluation-Program til Windows, men kan i den gratis version kun bruges til små kode-størrelser, - på max 2K.

Der skal skrives en kildetekst, (dvs. en Source Code).

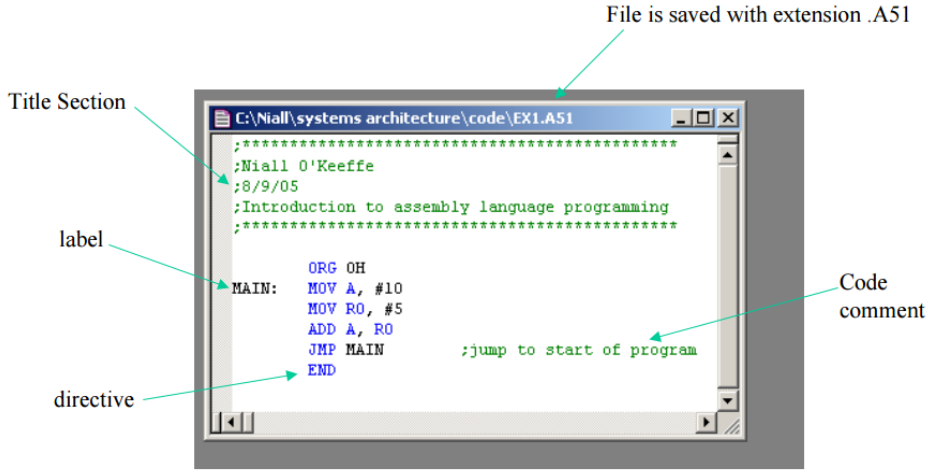
Her bruger vi assembler, senere leger vi med højniveau-sproget ”C”.

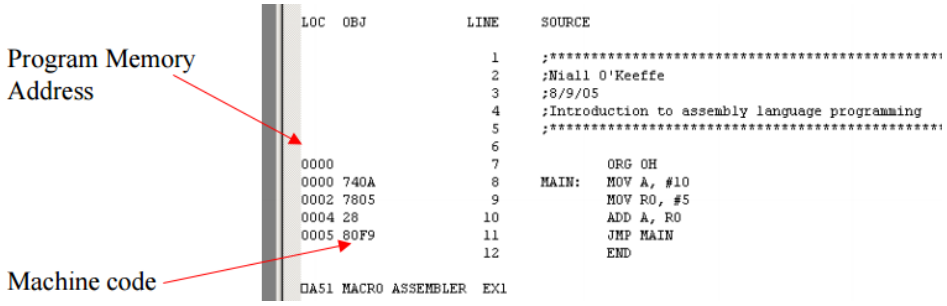
Kildeteksten oversættes til HEX koder, som skal ”Brændes” over i en Micro-controller

Herefter skal Programmet testes.

Eksempler på Assembler-kode:

Her et eksempel på en assemblerkode





Og hvilke hexkoder, det bliver oversat til.

Et andet eksempel:

```
        ORG   0H          ;start (origin) at location 0
        MOV   R5,#25H     ;load 25H into R5
        MOV   R7,#34H     ;load 34H into R7
        MOV   A,#0        ;load 0 into A
        ADD   A,R5        ;add contents of R5 to A
                          ;now A = A + R5
        ADD   A,R7        ;add contents of R7 to A
                          ;now A = A + R7
        ADD   A,#12H      ;add to A value 12H
                          ;now A = A + 12H
HERE:   SJMP  HERE        ;stay in this loop
        END               ;end of asm source file
```

MOV A, #3Ah ; Flyt værdien 3Ah ind i reg A

```
Mov P1, A          ; Flyt en kopi af reg A til reg P1

JMP …….

Mov R3, #00h       ; Move tallet 00 into reg R3

Mov A, P1          ; Flyt kopi fra Port1 til Acc.

Mov a, 07          Uden #, betyder RAM adr. 07
```

Eksempler på tal.

#123	Decimal
#123d	Decimal
#3Ah	Hexadecimal
#0A4h	Hex (Et tal må ikke starte med en bogstav)
#11110000b	Binary

Alfabetisk Instruktionsliste

- **ACALL**: Absolute Call
- **ADD, ADDC**: Add Accumulator (With Carry)
- **AJMP**: Absolute Jump
- **ANL**: Bitwise AND
- **CJNE**: Compare and Jump if Not Equal
- **CLR**: Clear Register
- **CPL**: Complement Register
- **DA**: Decimal Adjust
- **DEC**: Decrement Register
- **DIV**: Divide Accumulator by B
- **DJNZ**: Decrement Register and Jump if Not Zero
- **INC**: Increment Register
- **JB**: Jump if Bit Set
- **JBC**: Jump if Bit Set and Clear Bit
- **JC**: Jump if Carry Set
- **JMP**: Jump to Address
- **JNB**: Jump if Bit Not Set
- **JNC**: Jump if Carry Not Set
- **JNZ**: Jump if Accumulator Not Zero
- **JZ**: Jump if Accumulator Zero
- **LCALL**: Long Call
- **LJMP**: Long Jump
- **MOV**: Move Memory
- **MOVC**: Move Code Memory
- **MOVX**: Move Extended Memory
- **MUL**: Multiply Accumulator by B
- **NOP**: No Operation
- **ORL**: Bitwise OR
- **POP**: Pop Value From Stack
- **PUSH**: Push Value Onto Stack
- **RET**: Return From Subroutine
- **RETI**: Return From Interrupt
- **RL**: Rotate Accumulator Left
- **RLC**: Rotate Accumulator Left Through Carry
- **RR**: Rotate Accumulator Right
- **RRC**: Rotate Accumulator Right Through Carry

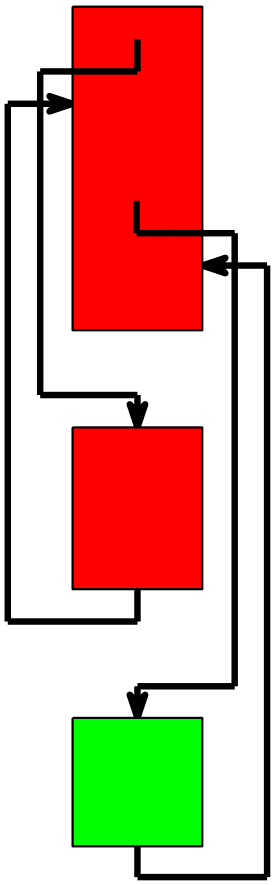
- **SETB**: Set Bit
- **SJMP**: Short Jump
- **SUBB**: Subtract From Accumulator With Borrow
- **SWAP**: Swap Accumulator Nibbles
- **XCH**: Exchange Bytes
- **XCHD**: Exchange Digits
- **XRL**: Bitwise Exclusive OR

For assembly-tutorial, se fx:
http://www.polyengineeringtutor.com/8051%20Assembly%20Programming.pdf

Program struktur

Brug mange sub-rutiner

Det giver bedre program struktur



Main Program

Call Subrutine

Call Subrutine

Subrutine 1

Return

Subrutine 2

Return

Keil: uVision setup

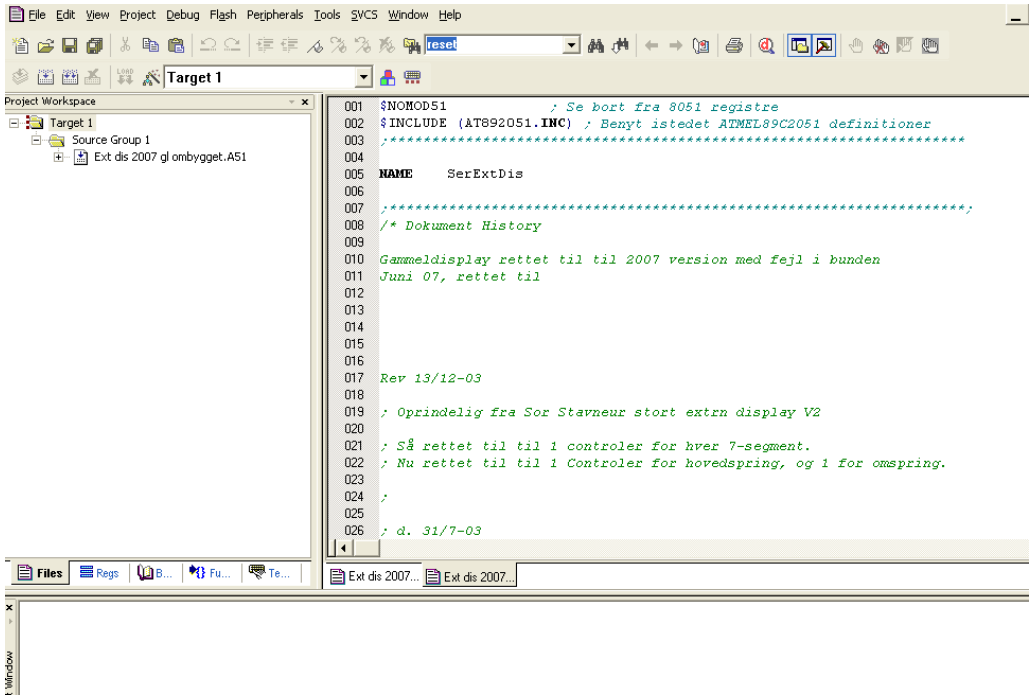
Her er vist et skærmbillede fra uVision

I venstre side ses:

Projekt struktur

I Højre side:

Source code



Her ses et eksempel på opbygning af kildetekst

Evt. kode for et interrupt (kan) placeres sidst !!

```
/*  Program header

Af:

*/

; def af pins

    Count_pin equ P3.2

; def af Ram_adresser

    LCD_ENABLE      EQU P1.5

; def af  Konstanter

    Offset_taeller  Equ 60h

; programstart
    Org 0
    Jmp Start

; plads til interruptvektorer

Start:
    Org 30h

    Mov DPTR, #Vaerdier ; Få Datapointeren til at pege i Rom-en på Tabellen
Forfra:
    Mov Offset_taeller, #0h ; nulstil Offset-tæller
Igen:
    Mov a, Offset_taeller
    MOVC a, @A+DPTR     ; Hent ind i reg A fra ( tabellen + Offset )

    CJNE a, #00h, Fortsaet  ; Tjek for EOT
```

Obs.:
Ingen æ, ø eller å. Kun evt. i kommentarer.

Kommentarer gerne på engelsk.

1 linje kommentar: efter ; eller vist //
Flere linjer /* til */

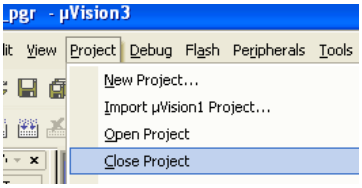
Mange forklarende linjer.

Husk definition af variable og pind mm.

Labels: Bare steder i et program. Ingen mellemrum!!

Opstart af et Projekt i uVision fra Keil:

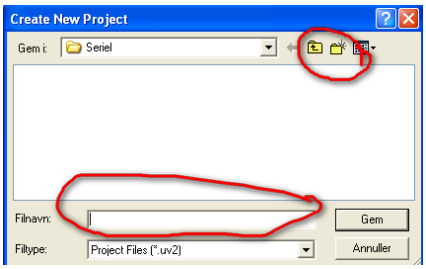
Hvis der er åbnet et project, Vælges: Projekt, Projekt Close



Skab et nyt project. Vælg Project / New Project

Vælg navn og mappe, projektet skal oprettes i. Det er vigtigt, alle projektets filer kommer i samme mappe.

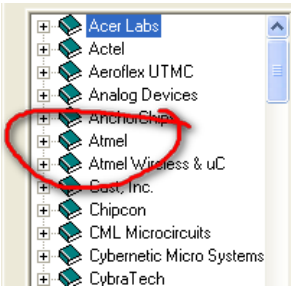
Lav ny mappe til hver projekt.
Brug aldrig de danske æ, ø eller å.



Der spørges nu efter hvilken type uC, der skal arbejdes med.

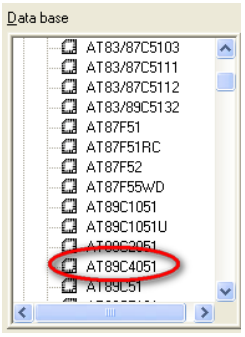
(Select device for Target).

Vælg Atmel > Vælg AT89C4051



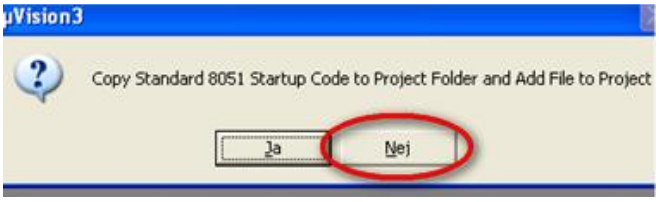
Scroll ned blandt Atmels uControllere.

Vælg AT89C4051.



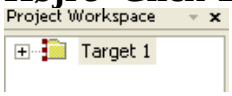
Der spørges nu, om der skal kopieres startup-kode.

Vælg NEJ.

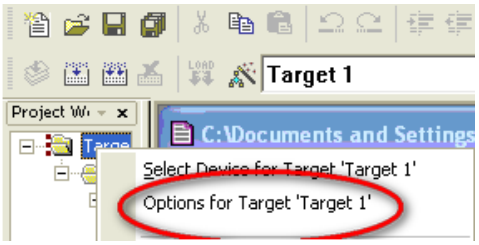


Vælg NO / NEJ.
Vigtigt !

Højre- Click Target 1



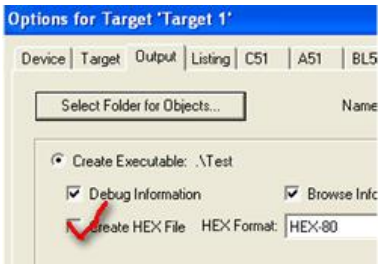
Vælg
"Options for "Target 1" / Output



Sæt hak I boksen

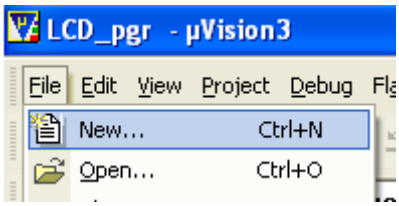
"Create Hex File"

Luk.



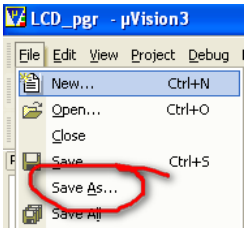
Nu skal der skabes en ny kildetekst-fil.

Create new file



File Save As.

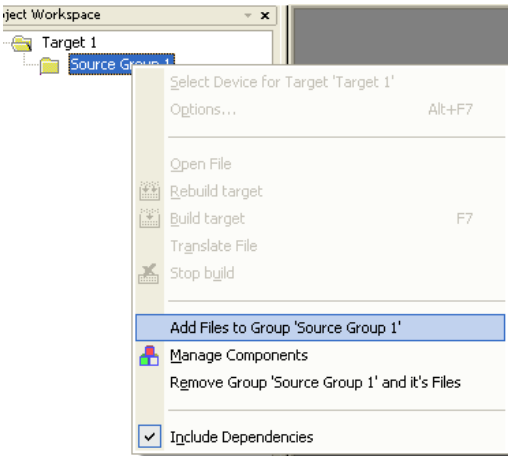
Gem filen med same fornavn som projektet, men med efternavnet .A51



Åben target 1

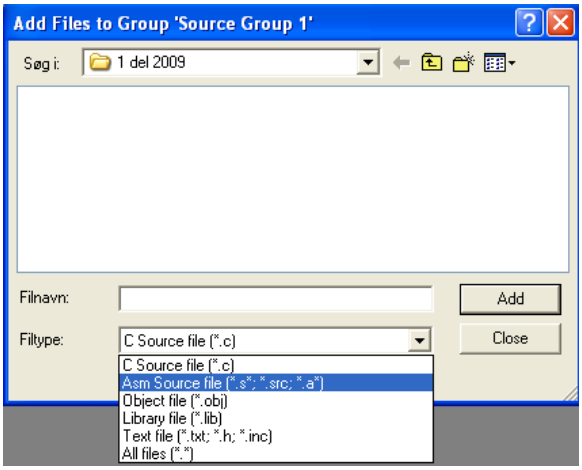
H-Klik på Source group 1 for at addere den just skabte kildetekst-fil.

Vælg "Add file to source group"



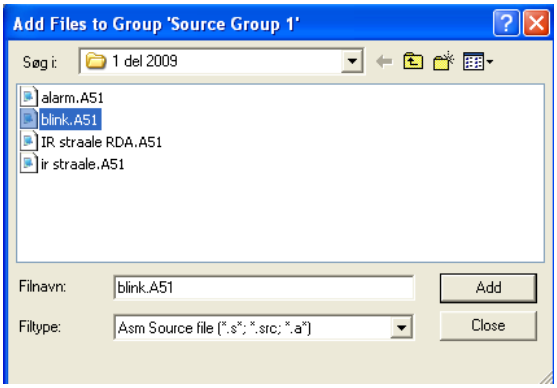
Find og adder kildetekstfilen.

Vælg *.a* for at vise filer, der ender på .A51



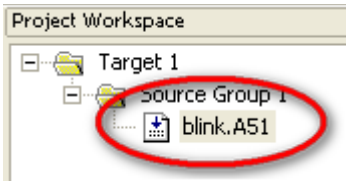
Her er vist et antal A51-filer i mappen.

Vælg den fil, der er oprettet, Klik på Add og Close.



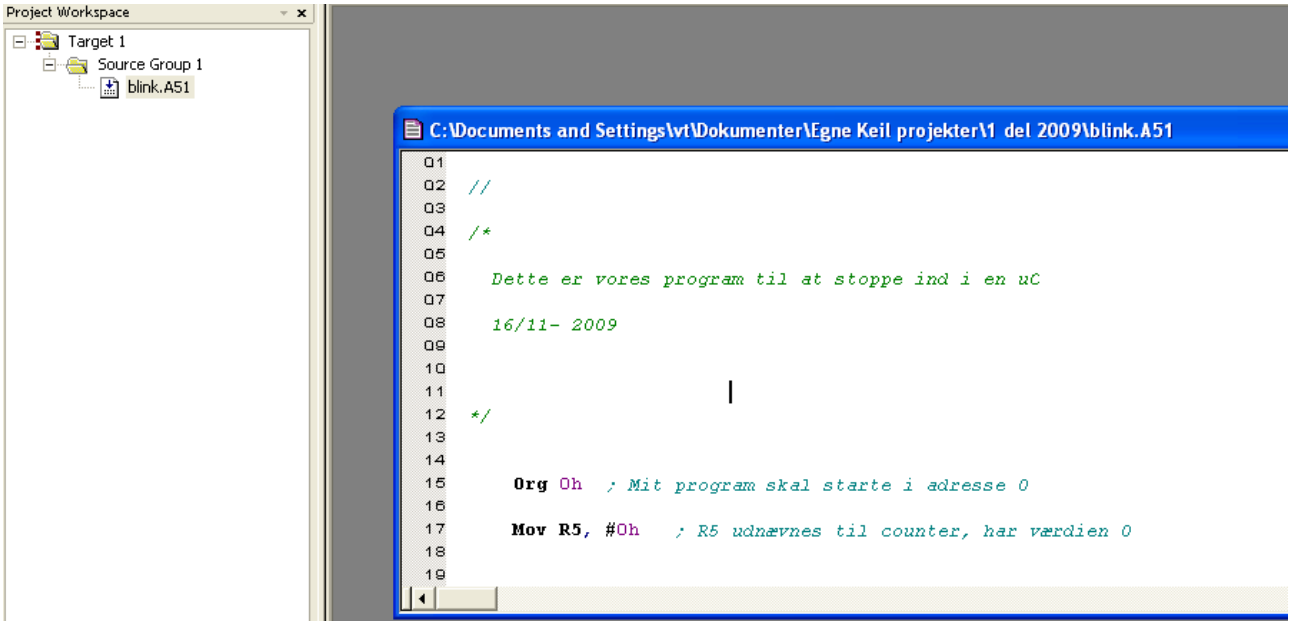
Projekt-fil-strukturen kan nu se ud som fx. dette. Kildetekstfilen hedder her "Blink.A51"

Hvis der Dobb. Klikkes på den, hopper cursoren over til højre til kildetekstfilen. – Klar til at skrive kilde-kode.



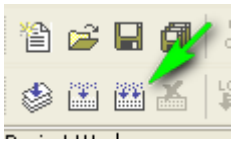
Ovenstående procedure kan også findes på min hjemmeside:

Design-vinduerne i uVision.



Skriv source code

Oversæt source code. Klick på



Hvis der ikke er syntax-fejl, skabes der en ".HEX" fil, der efterfølgende skal brændes over i microcontrolleren !!

Programmet skaber en række filer. Sørg for de bliver gemt i hver deres projekt-bibliotek, så de lettere kan ryddes ud igen !!

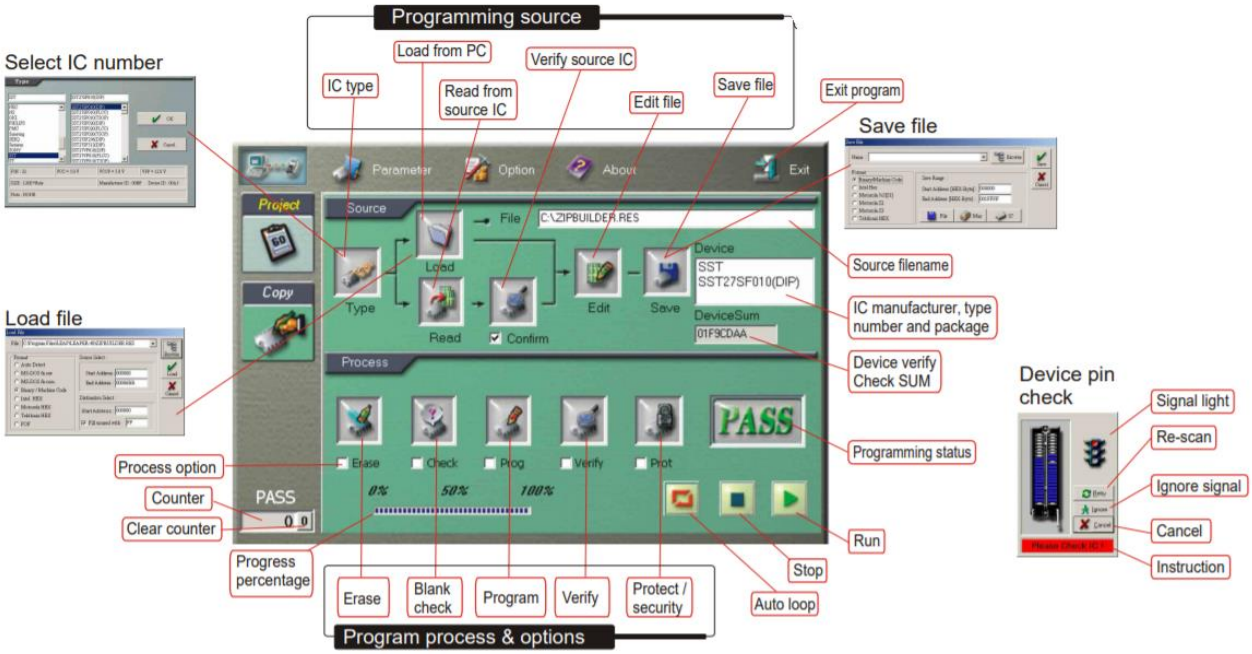
Fx:

uvproj	**Projektfilen**
Opt	**Options for projektet**
Lst	**Fil med kildetekst og oversat kode**
A51	**Selve kildeteksten**
Hex	**Den kode, der skal brændes over i uC-en**

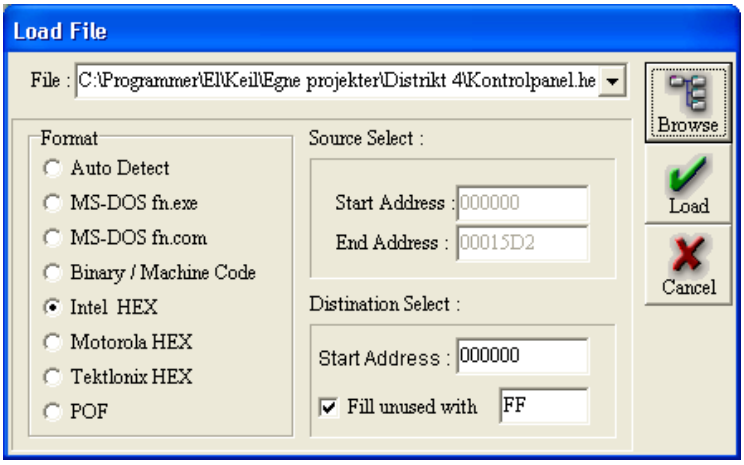
Osv.

Brænderprogrammet Leaper48

Vælg korrekt Device (type select). Hver IC skal programmeres på sin egen måde, dvs. der er skrevet et hav af drivere.



Browse for at finde og loade HEX-filen.



Run Program

Test uC i application

/ Valle Thorø