C for uC kompendium

Denne compendium er et forsøg på at lære lidt om C for uC, med Keil syntax.

Startet d. 30/3-2010

/ valle

Kildetekst:

I Keil skrives en kildetekst som det sker i asembler. Her skal kildetekst-filen dog have extension ” .c”

C har den fordel, at der kan udføres komplekse beregninger, at interrupts er lettere at styre osv.

Men man kommer ikke uden om at man skal kende sfr’s til opsætning af timere osv.

Operatorer:

Operator	Forklaring	Eksempler

+		a = a + b;
		a += b;
		
		a = a + 1; // Increment med 1
		a++

-		a = a - b; // a = a - b
		a -= b;
		
		a = a - 1; // Decrement
		a--; // som do.

*	Gange	a = a * b; // a = a gange b
		a *=b; // som do.

/	divider	a = a / b; // a = a divideret med b.
		a /= b; // som do.

%		a = a % b; // a = restdelen ved division
		a %=b; // Restdelen ved division.

!	Not, på bit niveau	P1_0 = !P1_0 if (a!=b) y = 0; // != Not equal,
~	Not, på byte niveau	P1 = ~P1

&	And	P1 = P1 & 0xF0 //P1 og F0h and’es.
		
		a = a & b; // Bitvis and funktion
		a &=b; // som do.

&&	” Og hvis samtidig ”	if ((x == 5) && (y == 7))
		DoSomething();
		
		if ((P3_4 == 1) && (P1_1 == 1) && (P1_0 == 1) &&(!koert))

|	Or	P1 = P1 | 0x81
>>	Shift right	
<<	Shift left	P1 = 0x01 P1 = (P1<<8) Nu er P1 = 10000000

= =	Compare	if ((P1 = =0) & (a <= 128))
		{ Kode her
		}

<	Mindre end	
>	Større end	
<=	Mindre end eller lig med	
>=	Større end eller lig med	
!=	Not equal	
^	exor	r = a^b
sqrt		at2 = sqrt(at1);
Rand();	Random, Kræver include af stdlib.h	LED1 = rand(); y = rand(); // Y er unsigned char !!
nop ();	ASM Nop	_nop_ (); // kræver include <intrins.h>

Datatyper:

Data Type	Bits	Bytes	Value Range
Bit	1		0 to 1
Signed char	8	1	-126 to +127
Unsigned char	8	1	0 to 256
Enum	16	2	-32768 to +32767
Signed short	16	2	-32768 to +32767
Unsigned short	16	2	0 to 65535
Signed int	16	2	-32768 to +32767
Unsigned int	16	2	0 to 65535
Signed long	32	4	-2147483648 to +2147483647

Unsigned long	32	4	0 to 4294967295
Float	32	4	+/-1.7549E+38 to +/-3.402823E+38
Sbit	1		0 to 1
Sfr	8	1	0 to 256
Sfr16	16	2	0 to 65535

For example, the declaration: sfr P0 = 0x80, declares the variable p0 and assigns it the special function address of 0x80. This is the address of PORT 0 on the 8051.

The C51 compiler automatically converts between data types when the result implies a different data type. For example, a bit variable used in an integer assignment is converted to an integer.

Bemærk, at I C-sproget, skal alle linjer afsluttes med semicolon, undtagen de, der slutter med parenteser, '{' eller '}'.

Source code organisering:

Følgende afsnit gennemgår organiseringen af en C-source kode fil.

// Headers

Her inkluderes biblioteker dvs. .h filer. Headerfiler kan bruges til at deklarere SFR-navne, Konstanter, Matematiske funktioner Skal der bruges matematiske operationer, skal man inkludere matematik-biblioteket. I bilag, se liste over biblioteker:	#include <Reg2051.h> #include <Math.h> // til Matematikfunktioner #include <stdlib.h> // til random funktion #include <intrins.h> // til asm nop #include <stdio.h> Herefter kan udføres denne beregning. ***a =(c*cos(b))+sin(b);***

Deklarering af variable

Erklæring af globale variable. Bruges normalt i uC, med mindre man løber tør for RAM.
Bruger mere RAM end lokale variable.

Variable skal erklæres før de kan bruges. Der	unsigned char a,b,c;

skal angives, hvilken type, de er, dvs. hvilke data, de kan indeholde. Og dermed også hvor meget de fylder i RAM	signed char d;
	a = 100;
	b = 200;
	c = a - b;
	d = a - b;
	
	unsigned char x= 5; // 0 til 255 Global !

Erklæring af et array, eller gruppe, kaldet display, med 10 elementer.
 	
De har nummer 0 til 9.
	char display[10];
	
	display[0] = 100;
	display[3] = 60;
	display[1] = display[0] - display[3];
	
	int tal_array[6]={0x08, 0x14, 0x3A, 0x47, 0xff, 0xB4};
	// Array starter fra nummer 0 til 5
	
	char byte_array[5]={'a', 0x02, 'b', 'e', 0x34 }; // C kan desværre ikke håndtere binære tal.

Erklæring af konstanter ??? I ROM. Som do, men med prefix "code".	***code unsigned char message[500];***

Definering af konstanter:	**#define led_on_time 184** #define hoj 1 // Navnet hoj er lig 1 #define lav 0 #define true 1 #define false 0

Definering af pin-navne:
 	
I stedet for navnet LED1 menes pin P1_0
 	
Det kan gøres – åbenbart med define eller sbit.
	#define LED1 P1_0
	#define LCD_RS P1_0 // Register select
	#define LCD_EN P1_1 // Enable
	#define LCD_D4 P1_2 // Data bits
	#define LCD_D5 P1_3 // Data bits
	#define LCD_D6 P1_4 // Data bits
	#define LCD_D7 P1_5 // Data bits
	
	sbit kontakt = P3_0; // variablen kontakt er forbundet til P3 bit 0.
	sbit On = P3_2;
	sbit button = P3_4;

Prototypes:

Erklæring af prototypes. Ellers kan der ikke kaldes funktioner længere nede i et program.
Her en liste over de følgende sub-programmer.
Det ser tilsyneladende ud til, at et kald til en anden sub nedad i programmet ikke kan ske, med mindre, der her er anført en liste over sub-program-navne, afsluttet med semikolon.
	/***
	* Prototype(s) *
	***/
	void LCD_delay(unsigned char ms);
	void LCD_enable();
	void LCD_command(unsigned char command);
	void LCD_putc(unsigned char ascii);
	void LCD_puts(unsigned char *lcd_string);
	void LCD_init();
	void Setup_seriel(void);
	
	// void er vist nok ikke nødvendig !!!!!

Functions body

Her placeres alle funktioner, der hører til programmet. (eller sub-programs)

Også interrupt funktioner.

Initialisering

Sub-program, der kun køres ved power_on. .
Fx opsætning af konstanter, opsætning af timere, seriel del, interrupts osv.

Main-Infinit loop.

En uC skal altid køre. Derfor er main-programmet en løkke, der altid skal køre. **main(){** *...* *main-funktionens kode* *...* **}**	**main(){** **while(1){ // infinite loop** **delay(30000);** **P1_0 = 0;** **delay(30000);** **P1_0 = 1;** **}** **}**

Løkkestrukturer:

IF:

if (*expression***) {** *…* *code to be executed* *…* **}**	**if ((P1 == 0) & (a <= 128)){** *…* *code to be executed* *…* **}**

IF Else

	if (*expression_1***) {** *…* *code block 1* *…* **}else if(***expression_2***) {** *…* *code block 2* *…* **}else if(***expression_3***) {** *…* *code block 3* *…* **}else{** *…* *code block 4* *…* **}**

For Loops

for(*start***;***condition***;***step***){** *…* *code block* *…* **}**	**For(i=0;i<10;i++){** **P0 = i;** **}**

While

	while(i < 10){ **P0 = i;** **i = i +1;** **}** **while(1){** **delay(30000);** **P1_0 = 0;** **delay(30000);** **P1_0 = 1;** **}**

Switch:

Ej færdig:

```
char a;

char With_switch()
{
switch(a)
{
case '0': return 0;
case '1': return 1;
case 'A': return 2;
case 'B': return 3;
default:  return 255;
}
}

char With_if()
{
if(a=='0') return 0;
else if(a=='1') return 1;
else if(a=='A') return 2;
else if(a=='B') return 3;
else return 255;
}
```

In Test Case 4, we still have 11 possible function calls. However, the indices for the functions are non-uniformly spaced. For the switch-based code, we simply fill in the requisite values as shown below:

```
char switch_fn(unsigned char i)
{
    char res;

    switch(i)
    {
      case 0x30:
        res = fna();
        break;
```

```
      case 0x35:
        res = fnb();
        break;

      case 0x36:
        res = fnc();
        break;

      case 0x39:
        res = fnd();
        break;

      default:
        res = 0;
        break;
    }

    return res;
}
```

Funktioner:

Programstumper, der kan bruges mange gange i et program.

Function_name(*parameter_1, Parameter_2, Parameter_3***){**
...
function body
...
return *value (optional)*
...
}

Funktion med navnet delay. Den har en parameter kaldet y. Dvs. der skal et tal med, når funktionen kaldes, unsigned int. I denne funktion gemmes tallet i variablen y. Der oprettes en int variabel, i, der starter på 0, og så længe i er mindre end y, udføres funktionens body, ig i forhøjes med 1. Funktionen kaldes fx med	Funktion uden return-værdi: **delay(unsigned int y){** **unsigned int i;** **for(i=0;i<y;i++){** **;** **}** **}**

delay(30000); Dette giver ca. 1 sek delay ved 12 MHz.	

Funktion med retur variabel

En funktion kan beregne, og returnere Kaldes med **angle = deg_to_rad(102,18);** Variablen angle får resultatet af funktions-beregningen.	**deg_to_rad(float deg){** **float rad;** **rad = (deg * 3.14)/180;** **retrun rad;** **}**

Timeren:

```
#include<reg51.h>

sbit pulse = P3^0;
void main(void)
{ while(1)
 { TMOD = 0b00010000; //Timer 1, mode 1 (16-bit)
  TL1=0x00;     // load TL with 0x00
  TH1=0xDC;     //load TH with 0xDC
  TR1=1;      //start Timer1

  while(!TF1){;}    //wait for timer1 overflow

  TR1=0;      //stop timer1

  pulse = ~pulse;  //complement P3.0

  TF1=0;      //clear timer1 flag
 }
}
```

Interrupts:

Hvordan er det lige syntaxen er, med interrupts for prioritet, er det nummer ?? altså at 4 er seriel ??? hvad så med at øge prioriteten ??

Ekstern:

Opsætning af extern interrupt: IT0/IT1 = 1 (External interrupt caused by a falling edge signal on P3.2/P3.3) IT0/IT1 = 0 (External interrupt caused by a low level signal on P3.2/P3.3)	**EA = 1;** **EX0 = 1;** **EX1 = 1;** **IT0 = 1;** **IT1 = 1;** Evt. opsætning i funktion: **setup_interrupts ()** // Function to setup the External interrupt 0 in the required mode **{** **EA = 1;** **EX0 = 1;** **IT0 = 1;** **}** Setup kaldes med **setup_interrupts ();**
Funktionen filter er en interrupt-funktion, nummer 0	**filter () interrupt 0** //The function the be executed when external interrupt occurs **{** **counter = 0;** //Reset the counter to 0 **}**

example:
CODE:

```
void External_Int0() interrupt 0{
//code
}
```

CODE:

```
void isr_name (void) interrupt 2 {
// Interrupt routine code
}
```

```
// Interrupt

   #include<reg51.h>

   sbit LED = P1^7;

   void turn_on(void) interrupt 0          // Extern int 0
   {
    LED=1;
   }

   void turn_off(void) interrupt 2         // Extern int 1
   {
    LED=0;
   }

   void main(void)
   {
    LED=0;          //Turn LED off
    IT0=0;         //set external interrupt 0 edge-triggered
    IT1=1;         //set external interrupt 1 edge-triggered
    EX0=1;          //enable external interrupt 0 interrupt
    EX1=1;          //enable external interrupt 1 interrupt
    EA=1;          //enable global interrupt
    while(1)       //loop forever
    {;}
   }
```

Tabel over interrupt numre

Interrupt nummer	Beskrivelse	adresse	
0	External Int 0	03h	
1	Timer / counter 0	0Bh	
2	External Int 1	13h	
3	Timer / counter 1	1Bh	
4	Seriel port	23h	

Timer:

setup_interrupts (); //setup the External interrupt

?????????

// The interrupt of the Timer 0 points to the interrupt priority number 1. For Timer 0, the format of the ISR must be:

```
 void your_ISR_name_here(void) interrupt 1
{
 //your routine here
}
```

The interrupt of the Timer 1 points to the interrupt priority number 3. For Timer 1, the format of the ISR must be:
*/

```
void your_ISR_name_here(void) interrupt 3
{
 //your routine here
}
```

```
#include<reg51.h>

sbit pulse = P3^0;

void toggle_pin(void) interrupt 3
{
 pulse = ~pulse;  //complement P3.0
}

void main(void)
{
  TMOD = 0b00010000; //Timer 1, mode 1 (16-bit)
  TL1=0x00;    // load TL with 0x00
  TH1=0xDC;    //load TH with 0xDC

  ET1=1;     //enable Timer 1 interrupt
  EA=1;      //enable global interrupt
  TR1=1;     //start Timer1

  while(1)   //loop forever
```

```
    {
  }
```

Seriel

Ej færdig:

```
SCON = 0x50;            /* mode 1, 8-bit uart, enable receiver */
TMOD = 0x20;            /* timer 1, mode 2, 8-bit reload */
TH1  = 0xF3;            /* reload value for 2400 baud */
TR1  = 1;               /* timer 1 run */
TI   = 1;               /* set TI to send first char of uart */
```

```
/* Send a character to the serial port */
void sndchr(x)
unsigned int x;
{
SM1 = 0;                    ??          /* clear the tx buffer full flag */
SBUF = x;
while (SM1){}
```

```
#include<reg51.h>

void my_uart_isr(void) interrupt 4
{ if(TI)        //check if interrupt is caused by Transmit Flag
   TI=0;        //clear TI
  else          //go here if interrupt is caused by Receive Flag
   {
    P1=SBUF;
    RI=0;       //clear RI
   }
}

void main(void)
{
 TMOD = 0x20;    //Timer 1, mode 2 (auto-reload)
 TH1 = 0xFD;     //load TH with -3 or FDh
 SCON = 0x50;    //UART mode 1, receive enabled
 TR1 = 1;        //start Timer1
 RI = 0;         //clear RI
```

```
  ES=1;          //enable UART interrupt
  EA=1;          //enable global interrupt
  while(1)       //loop forever
  {;}
 }
```

// Unlike the timer interrupts, the TI or RI must be cleared inside the ISR.

Kode eksempler:

```
#include <REGX52.h>
#include <math.h>

unsigned long time, ON_time; //Global Variables

void main(){
   P1_3 = 1;     //Set up P1_3 as input pin
   ON_time = 100000;
   while(1){
       if (time < ON_time){
          time++;    // start or continue counting
          P1_0 = 0;     //Turn ON the LED
       }else{
          P1_0 = 1;    // Turn OFF the LED
       }
       if (P1_3 == 0){     // if the switch is pressed,
          time = 0;    // reset 'time' to 0
       }
   }
}
```

Bilag:

REG51.H
Header file for 8051.

```
/*  BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;

sbit ET0  = 0xA9;
sbit EX0  = 0xA8;
    /*  IP   */
sbit PS   = 0xBC;
sbit PT1  = 0xBB;
sbit PX1  = 0xBA;
sbit PT0  = 0xB9;
sbit PX0  = 0xB8;
/*  P3  */
sbit RD   = 0xB7;
sbit WR   = 0xB6;
```

```
            sbit T1   = 0xB5;
            sbit T0   = 0xB4;
            sbit INT1 = 0xB3;

                                    /*  SCON */
sbit SM0  = 0x9F;
sbit SM1  = 0x9E;
sbit SM2  = 0x9D;
sbit REN  = 0x9C;
sbit TB8  = 0x9B;
sbit RB8  = 0x9A;
sbit TI   = 0x99;
sbit RI   = 0x98;
                                  /*  BIT Register */
/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;
                                    /*  TCON */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit INT0 = 0xB2;
sbit TXD  = 0xB1;
sbit RXD  = 0xB0;
```

Headerfiler i C (Søg: "C Standard Library")

– assert.h Enforcing assertions when functions execute
– ctype.h Classifying characters
– errno.h Testing error codes reported by library functions
– float.h Testing floating-point type properties
– iso646.h Using Amendment 1—iso646.h standard header
– limits.h Testing integer type properties
– locale.h Adapting to different cultural conventions
– math.h Computing common mathematical functions
– setjmp.h Executing non-local goto statements
– signal.h Controlling various exceptional conditions
– stdarg.h Accessing a varying number of arguments
– stdbool.h Adds support for the bool data type in C.
– stddef.h Defining several useful types and macros
– stdio.h Performing input and output
– stdlib.h Performing a variety of operations
– string.h Manipulating several kinds of strings
– time.h Converting between various time and date formats
– wchar.h Support for wide characters
– wctype.h Classifying wide characters