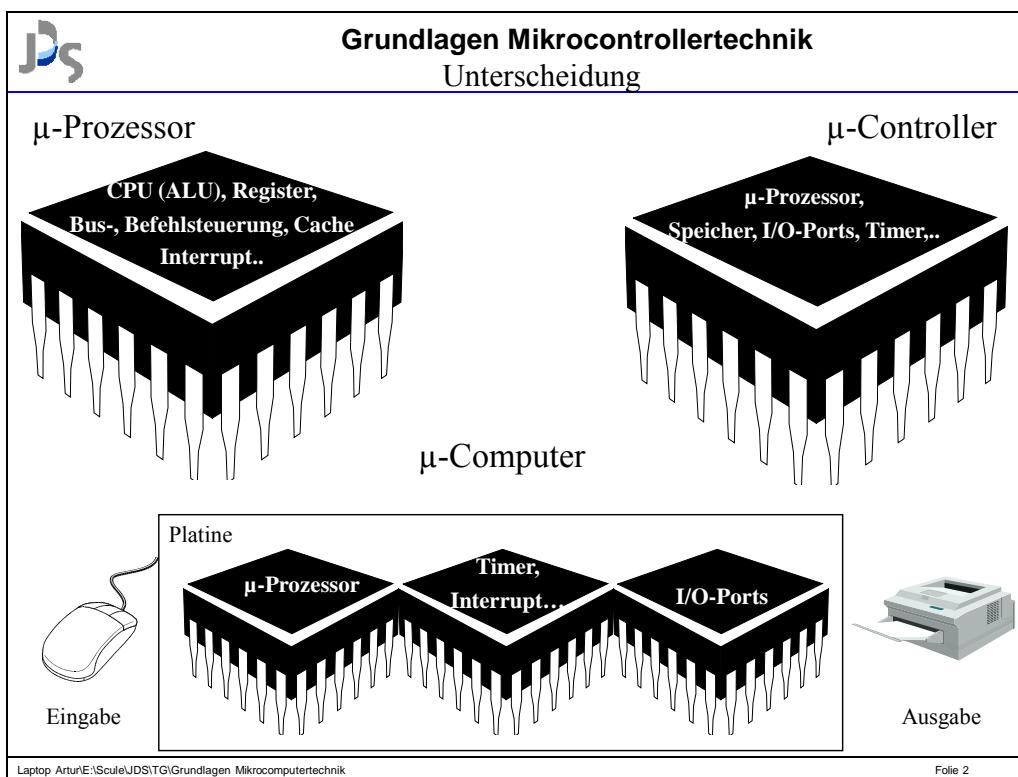
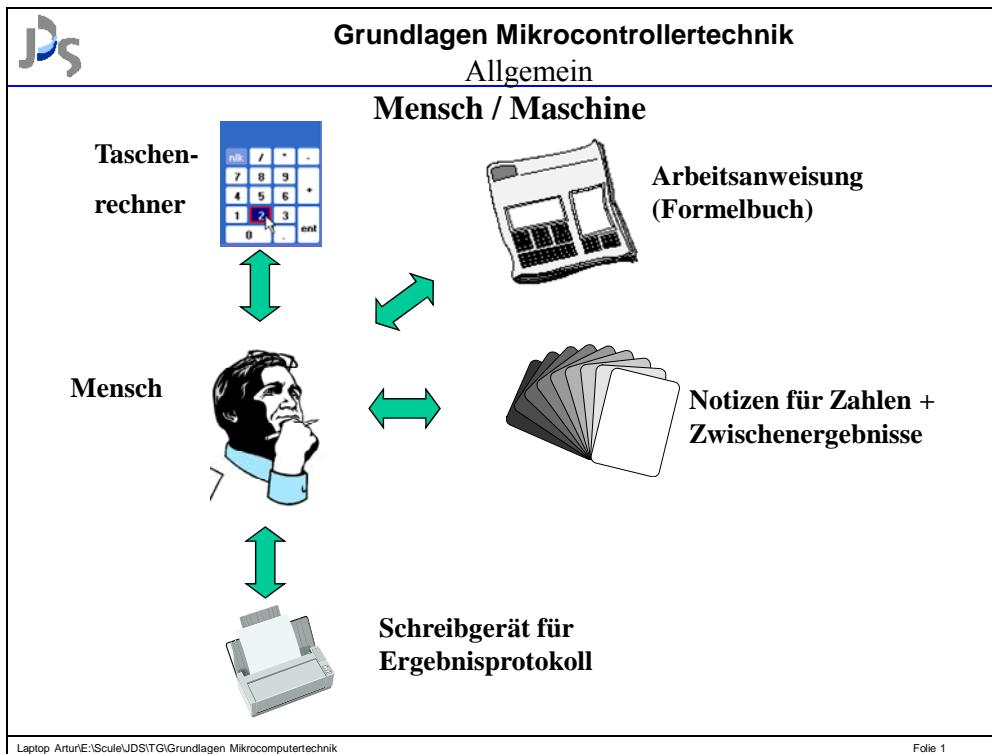
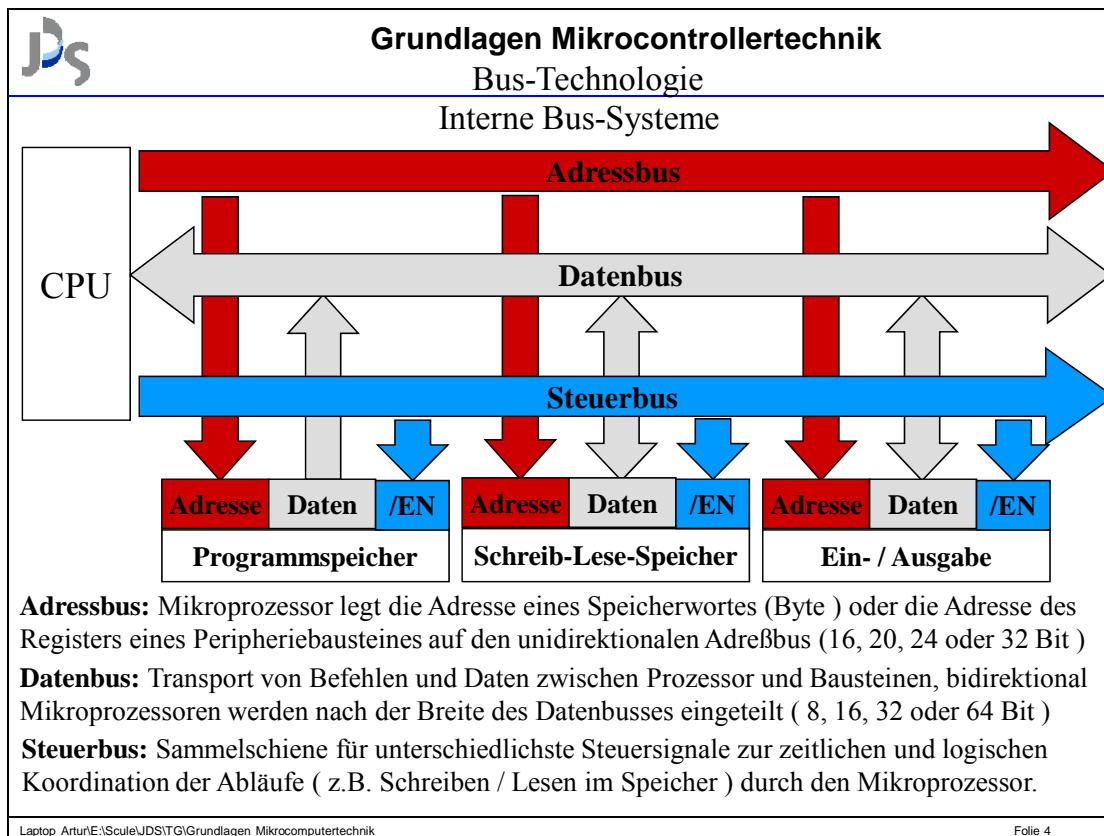




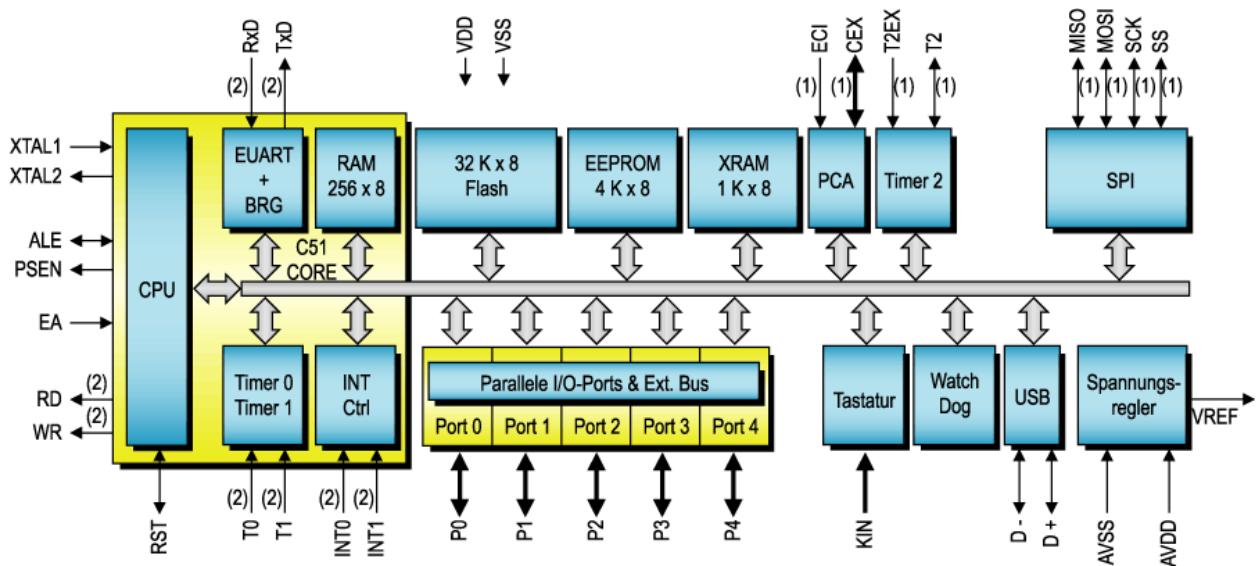
1. Einführung

1.1. Einführung





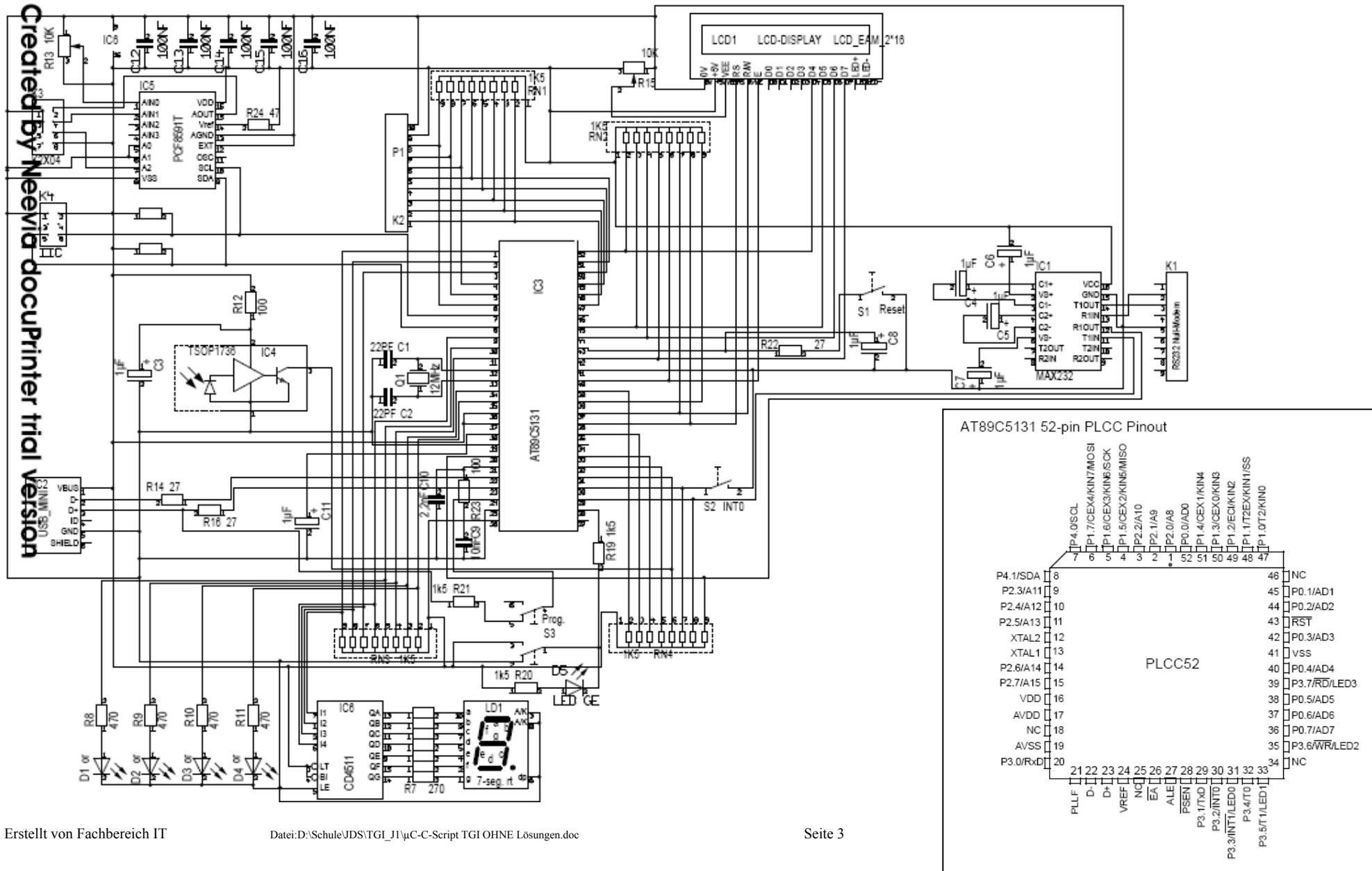
Blockschaltbild von at89C5131



μ C-C-Script TG-I IT, AT89C5131

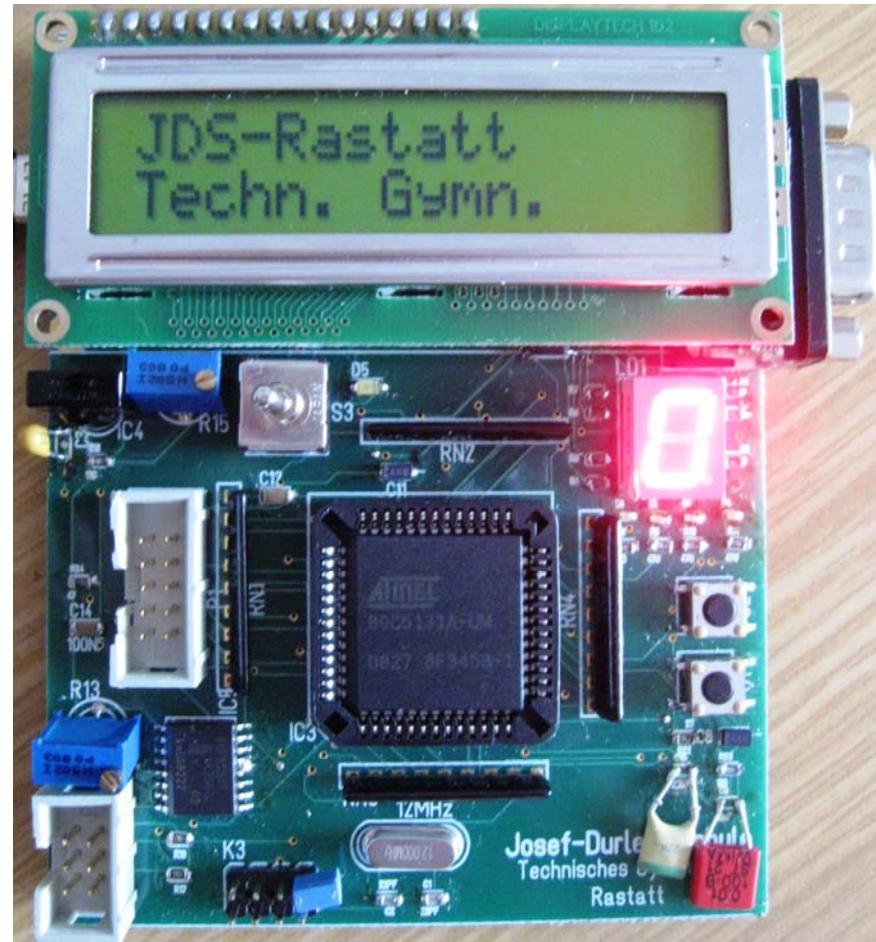
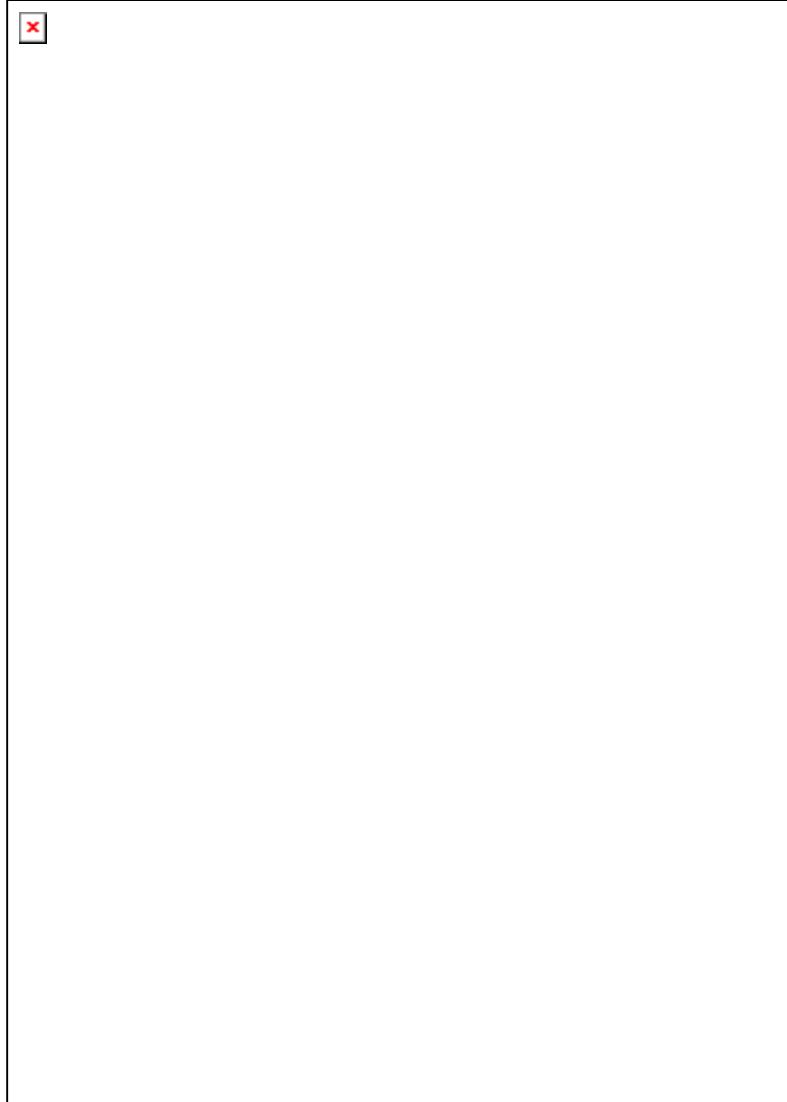


1.2. Aufbau der Projektplatine JDS-Rastatt, TG: Schaltplan



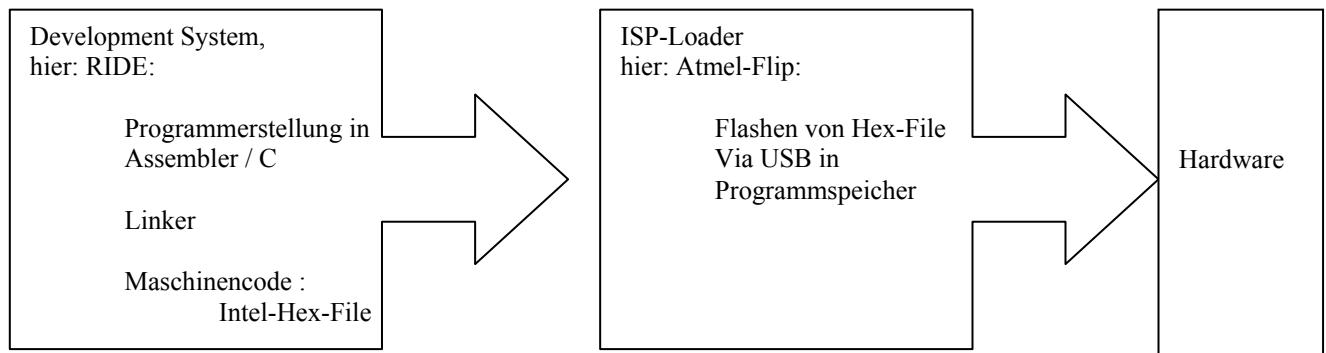


1.3. Aufbau der Projektplatine JDS-Rastatt, TG: Bestückungsplan





2. Entwicklungsumgebung – Programm erstellen – Programmierung Baustein



2.1. Entwicklungsumgebung RIDE

Für jedes Programm, das später in einen µ-Controller heruntergeladen werden soll, muss ein Projekt erstellt werden. Das Projekt kann mehrere Assembler und / oder C- Dateien enthalten, die später zusammengefügt (gelinkt) werden. Das gelinkte Programm erhält den Namen des Projekts.

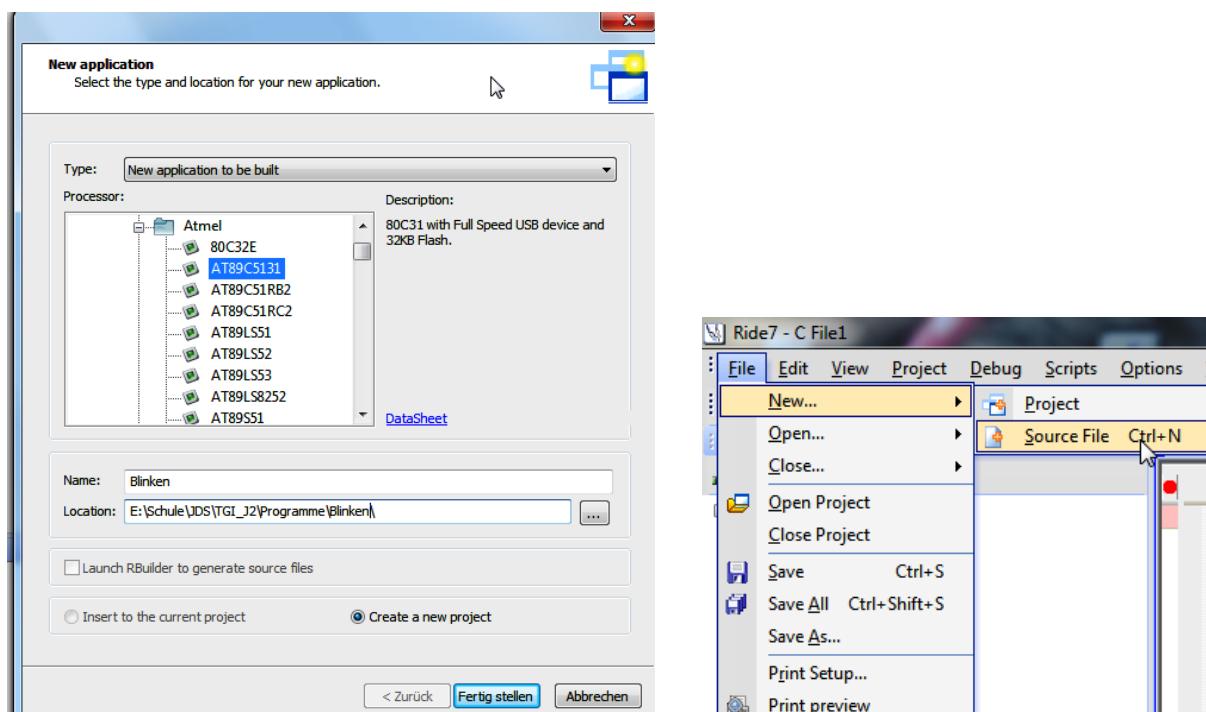
!! Der Pfad des Projektes sowie der Datei darf keine Sonderzeichen bzw. Leerzeichen enthalten und darf nicht länger als 62 Zeichen besitzen !!

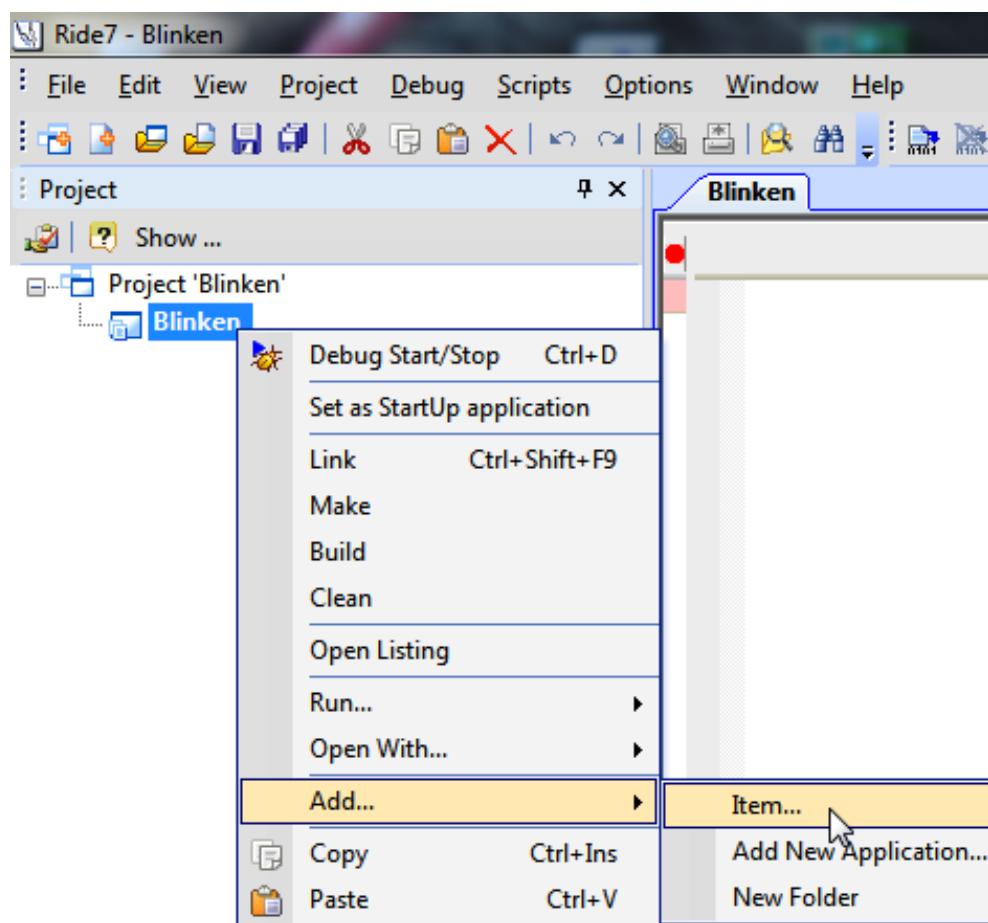
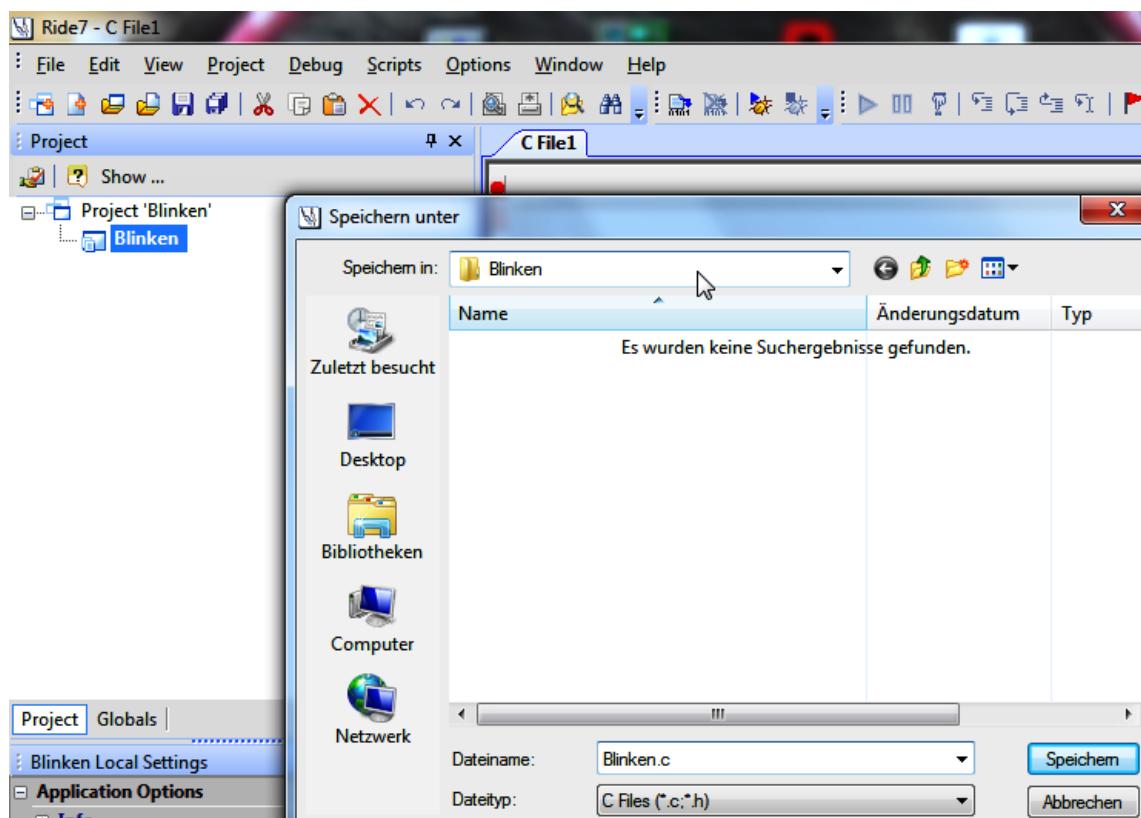
Installation RIDE 7 (Win 7, Vista, XP...)

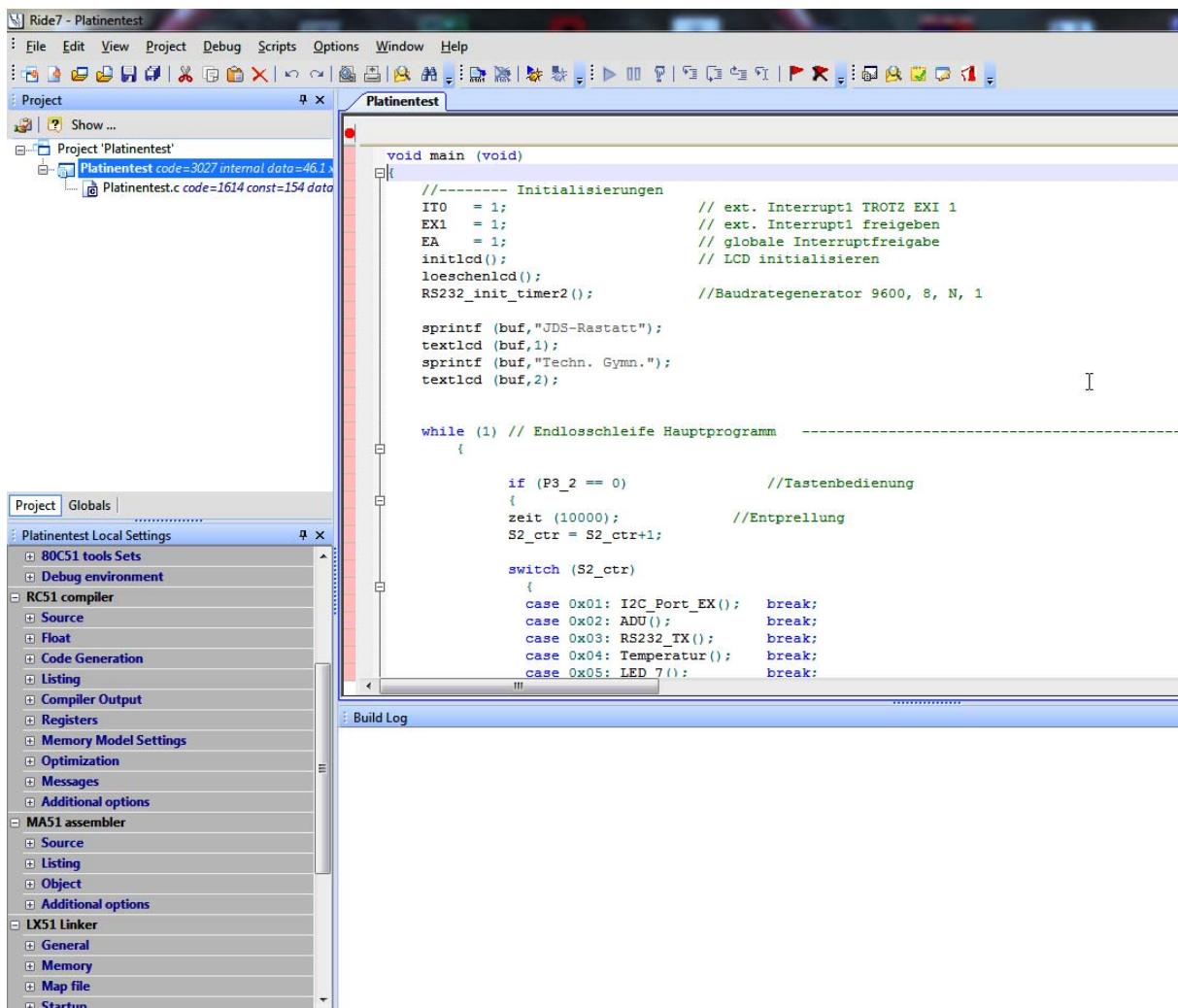
Installieren Ride7_7.30.10.0169.exe Installieren RKit-51_6.05.10.0125_eval
<http://www.schueler-projekte.jdsr.de/eRoadster/Rechts/Secret.UR/TG/IDE%20Mikrocontroller%208051.zip>

Programm starten:

- Neues Projekt IM EIGENEN ORDNER
- Neue Quelldatei
- Einbinden der Quelldatei







```
void main (void)
{
    //----- Initialisierungen
    ITO = 1;                                // ext. Interrupt1 TROTZ EXI 1
    EX1 = 1;                                // ext. Interrupt1 freigeben
    EA = 1;                                 // globale Interruptfreigabe
    initLCD();                               // LCD initialisieren
    loeschenLCD();
    RS232_init_timer2();                     //Baudrategenerator 9600, 8, N, 1

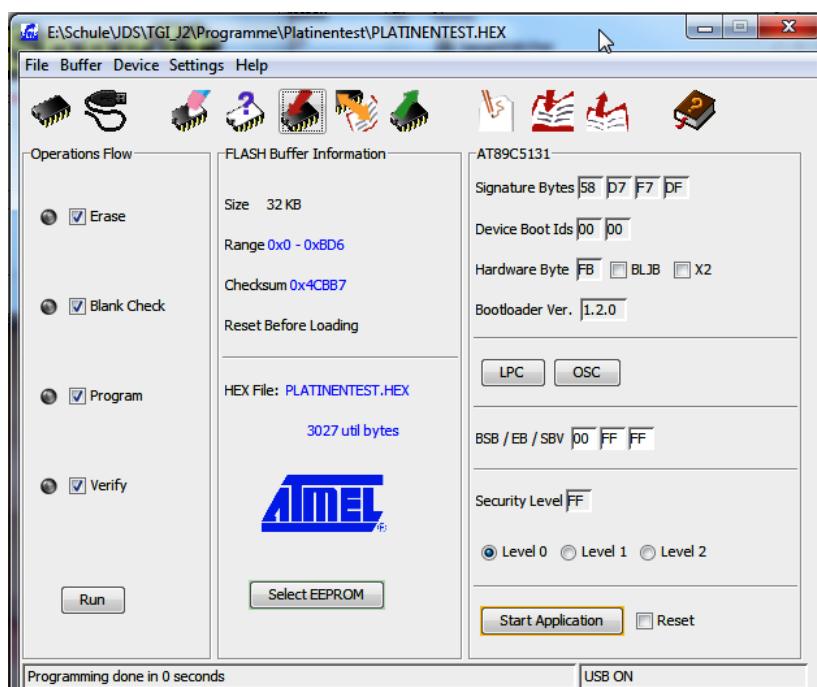
    sprintf (buf,"JDS-Rastatt");
    textLCD (buf,1);
    sprintf (buf,"Techn. Gymn.");
    textLCD (buf,2);

    while (1) // Endlosschleife Hauptprogramm -----
    {
        if (P3_2 == 0)                      //Tastenbedienung
        {
            zeit (10000);                  //Entprellung
            S2_ctr = S2_ctr+1;

            switch (S2_ctr)
            {
                case 0x01: I2C_Port_EX();   break;
                case 0x02: ADU();         break;
                case 0x03: RS232_TX();    break;
                case 0x04: Temperatur();  break;
                case 0x05: LED_7();       break;
            }
        }
    }
}
```

Installieren Programmer: Flip von Atmel

www.atmel.com





3. Auszug Formelsammlung TG:

Die Programmiersprache C (ANSI-C)

1. Datentypen, Variable, Konstante, Operatoren und Ausdrücke

Datentypen

Datentyp	Größe	Wertebereich
bit	1 Bit	0 oder 1
signed char	1 Byte	-128 bis +127
unsigned char	1 Byte	0 bis 255
signed int	2 Byte	-32768 bis +32767
unsigned int	2 Byte	0 bis 65535
signed long	4 Byte	-2147483648 bis +2147483647
unsigned long	4 Byte	0 bis 4294967295
float	4 Byte	$\pm 1,176E-38$ bis $\pm 3,40E+38$
pointer	1-3 Byte	Adresse einer Variablen
FILE		Dateizeiger

Operatoren und ihre Priorität

Mathematische Operatoren		Priorität	Verhältnis und logische Operatoren
++	Inkrement	Höchste	
--	dekrement		
-	monadisches Minus (Vorzeichen)		> >= < <=
*	Mal		== !=
/	Div		&&
%	mod		
+	Plus	niedrigste	
-	minus		
Bitweise Operatoren			
&	UND		Beispiele
	ODER		X = 10: Y = ++X → Y = 11 ; Y = X++ → Y = 10
^	EXOR		Y = -X → Y = 9 ; Y = X-- → Y = 10
~	Einerkomplement		Y = Y >> 1 schiebe Y um 1 nach rechts
<<	schieben nach links ;		
>> - nach rechts		



2. Aufbau eines C-Programms

C bietet zwei Möglichkeiten zur Kommentareingabe :

- a) **//** Kommentar für eine Zeile
- b) **/*** Kommentar für einen Block von einer oder mehreren Zeilen ***/**

```
{      Anfang eines zusammengehörigen Befehlsblocks      (begin)
}      Ende eines zusammengehörigen Befehlsblocks (end)
```

// INCLUDE Files: → Compileranweisung über zusätzliche Quellcodes mit Funktionen und Deklarationen

```
#include <reg51xx.h>      // Registerdeklaration zum 5051xx
#include <math.h>          // Einbinden mathematischer Funktionen
```

// Konstantendeklaration → Ablage im Programmspeicher → Wert im Programm nicht änderbar

```
#define ANZAHL 10          // ANZAHL entspricht 10
#define TRUE 1               // TRUE entspricht 1
```

// Deklaration globaler Variablen → Ablage im Datenspeicher → Wert im Programm änderbar

```
int i = 8, j = 3, k;           /* Zählvariablen i , j und k mit den Anfangswerten 8 für i und 3 für j */
char TASTE;                  /* 1 Byte große Variable von 0-255
signed long 4BYTE;           /* 4 Byte große Variable von 0 – 232
char code *text = "Hallo";   /* Textstring mit 5 Bytes (+ 0 als Stringende) im Programmspeicher*/
char ldata schieb;            /* 1 Byte-Variable im bitadressierbaren Speicherbereich
```

// Deklaration von Funktionen

```
Typ Funktion_1(Typ Parameter1, Typ Parameter2 )
{
    // Als Typ kann jeder Datentyp stehen. (void ⇒ keine Typzuweisung)
    // Begin von Funktion_1
    // lokale Variablen Deklaration;
    // Befehlsfolge;
}
```

```
Typ Funktion_2()
{
    // Begin von Funktion_2
    // lokale Variablen Deklaration;
    // Befehlsfolge;
}
```

// Hauptprogramm - auch Hauptfunktion main()

```
main( )
{
    // Begin des Hauptprogramms
    // lokale Variablen Deklaration;
    // Befehlsfolgen;
}
```



3 Befehle zur Steuerung des Programmflusses

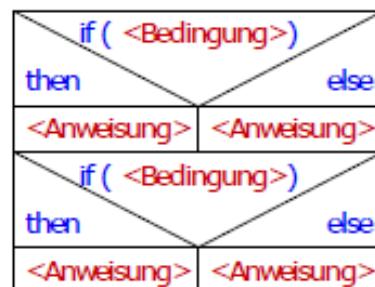
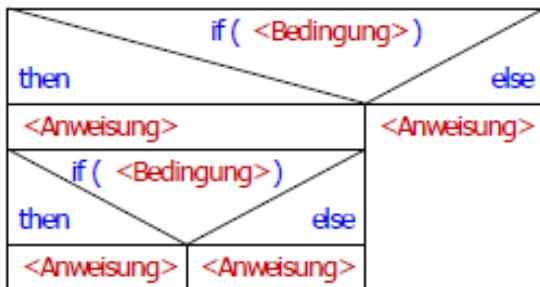
3.1 If , else

```
if (Bedingung)
{
    Befehlsfolge für wahre Bedingung ;
}
else
{
    Befehlsfolge für falsche Bedingung;
}
```



Anmerkungen:

- Die Befehlsfolgen selbst können wiederum If-Anweisungen sein oder verschachtelte if-Anweisungen
- Die else- Anweisung ist nicht zwingend notwendig



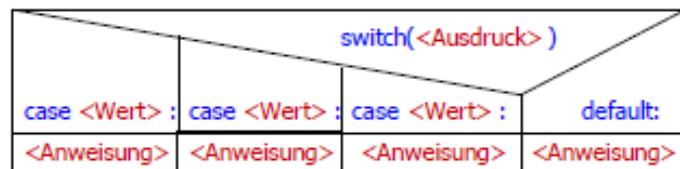
3.2 switch

```
switch (Ausdruck)
{
    case Konstante 1:
        Befehlsfolge 1;
        break;

    case Konstante 2:
        Befehlsfolge 2;
        break;

    case Konstante X:
        Befehlsfolge X;
        break;

    default:
        {
            Befehlsfolge;
        }
}
```



Anmerkungen:

- Die Befehlsfolgen selbst können wiederum If-Anweisungen oder switch-Anweisungen sein.
- Die default-Anweisung ist nicht zwingend notwendig.
- Der break-Befehl bricht die switch-Anweisung ab.



3.2 Schleifen

3.2.1 Die for-Schleife

```
for ( Initialisierung ; Bedingung ; Veränderung )
{
    Befehlsfolge;
}
```



Anmerkungen:

- geeignet für Schleifen, bei denen die Anzahl der Durchläufe bekannt ist.
- Der Körper der Schleife kann auch leer sein (for (; ;)) jedoch die Semikolon müssen bleiben.
- Die Initialisierung legt die Startwerte der Variablen fest. Es können dabei auch mehrere Variablen mit Komma getrennt initialisiert werden.
- Ist die Bedingung erfüllt (TRUE) wird die Befehlsfolge bearbeitet
- Nach der Befehlsfolge bestimmt die Veränderung, wie die Variablen verändert werden.

Beispiel:

```
for ( i = 0, j = 8 , z = 0 ; i < j ; i++ , j - = 2 )
{
    z = i + j;
}                                // z = 6
```

3.2.2 Die while-Schleife (Kopfgesteuert)

```
while ( Bedingung )
{
    Befehlsfolgen;
}
```



Anmerkungen:

- Ist die Bedingung nicht erfüllt, also FALSE, dann wird die Befehlsfolge nicht bearbeitet
- Die Befehlsfolgen werden solange wiederholt, solange die Bedingung erfüllt bzw. WAHR ist
- Endlosschleife mit while(1) oder while(TRUE)
- Controller-Programme werden normalerweise mit einem Hardwarereset abgebrochen. Daher fangen die Programme für den µC meistens mit while(1) { an

3.2.3 Die do-while-Schleife (Fußgesteuert)

```
do {
    Befehlsfolgen;
} while ( Bedingung );
```



Anmerkungen:

- Die Befehlsfolge wird mindestens einmal bearbeitet, auch wenn die Bedingung nicht erfüllt ist.
- Die Befehlsfolgen werden solange wiederholt, wie die Bedingung erfüllt bzw. WAHR ist
- Endlosschleife mit while(1) oder while(TRUE)

4. Der break-Befehl

In einer Schleife beendet der break-Befehl diese, die Programmsteuerung geht direkt an die auf die Schleife folgenden Befehle über.





4. Algorithmus erstellen: Struktogramm: Symbolik

Strukturierte Programmentwicklung

Zuweisung

Variable \leftarrow derAusdruck

Variable := derAusdruck

Sequenz

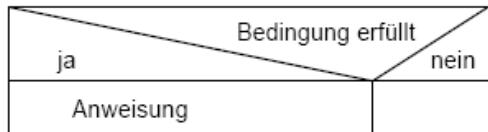
Anweisung 1

Anweisung 2

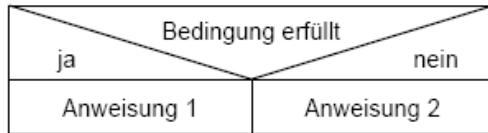
Anweisung 3

Auswahl

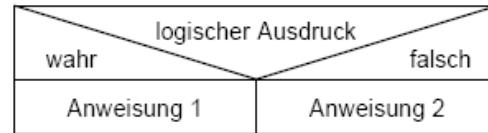
einseitige Auswahl



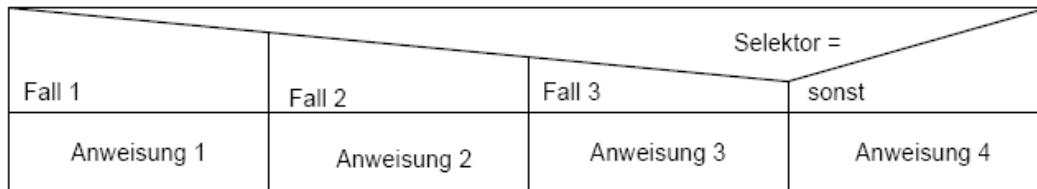
zweiseitige Auswahl



zweiseitige Auswahl

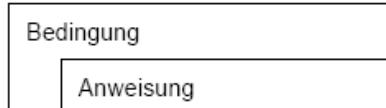


Mehrfachauswahl

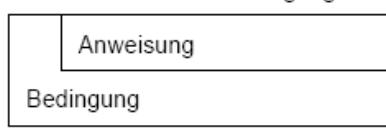


Wiederholung (Iteration)

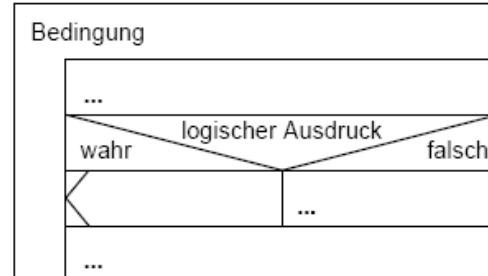
Schleife mit Eintrittsbedingung



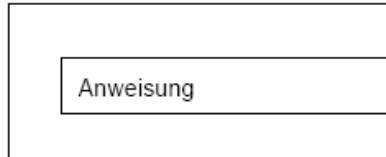
Schleife mit Austrittsbedingung



Schleife mit Abbruchmöglichkeit



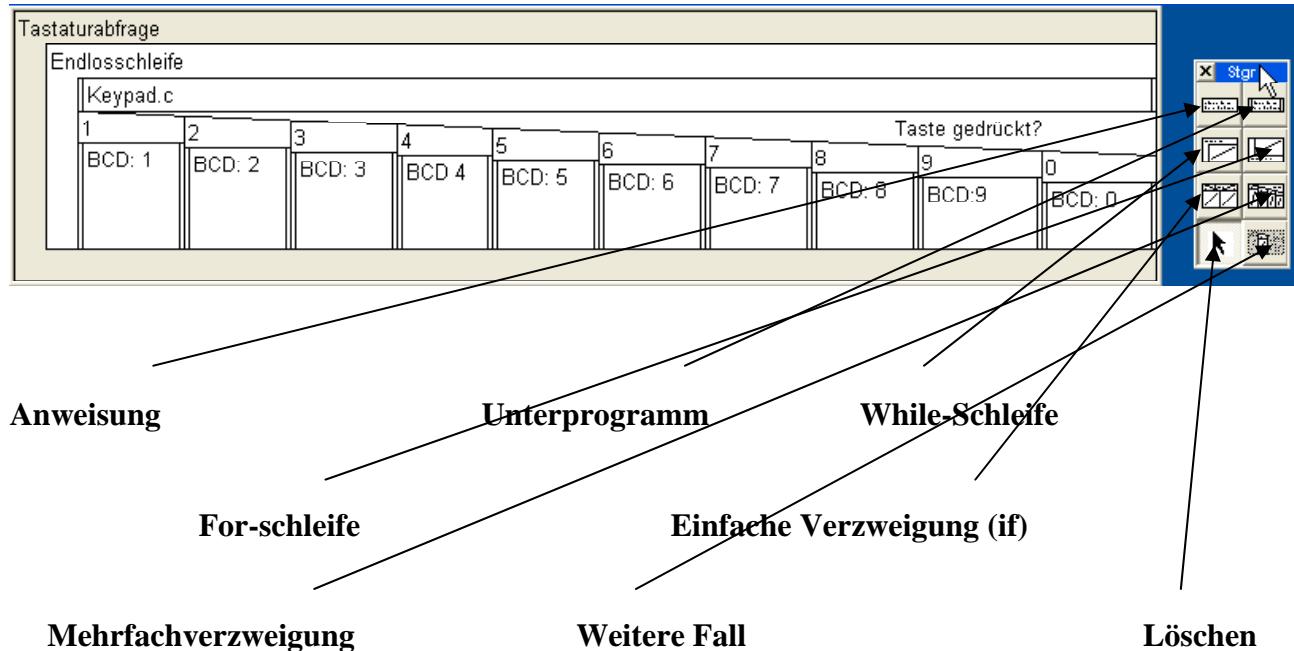
Schleife ohne Bedingungsprüfung (Endlosschleife)





Zum Entwickeln eines Algorithmus eignet sich für Hochsprachen besser das Struktogramm. Hierzu können wir ein freies Tool nutzen, um dies zu Visualisieren: den Struktogramm

Hier ein Beispiel für ein Struktogramm:



http://www.schueler-projekte.jdsr.de/eRoadster/Rechts/Secret_UR/TG/Struktogrammer.zip



5. Erste Schritte: Programmanalyse

AUFGABE: Erstellen Sie folgendes Programm und beantworten Sie die zugehörigen Fragen:
(wenn nötig mittels Debugger)

```
#include <at89c5131.h>           //Initialisierung
sfr at P2 Ausgabe;
sbit at P3_2 UP;

unsigned int i;                  //Variablendeclaration

void main (void) //----- Hauptprogramm -----
{
    Ausgabe = 0;
    while(1)
    {
        if (!UP)
        {
            Ausgabe++;
            if (Ausgabe == 10)
            {
                Ausgabe = 0;
            }
        }
        for (i=0xffff;i!=0;i--);
    } // Ende while(1)
} // Ende main
```

- 1.1 Beschreiben Sie die Funktion des Programms in Worten.
- 1.2 Ergänzen sie die Kommentare.
- 1.3 Erstellen Sie ein Struktogramm.
- 1.4 Welche Änderung ist notwendig, damit die LEDs „schneller zählen“?
- 1.5 Geben Sie die Programmteile für folgende Änderung an:
Wenn P3_2 gedrückt ist (1), soll aufwärts, ansonsten abwärts gezählt werden.

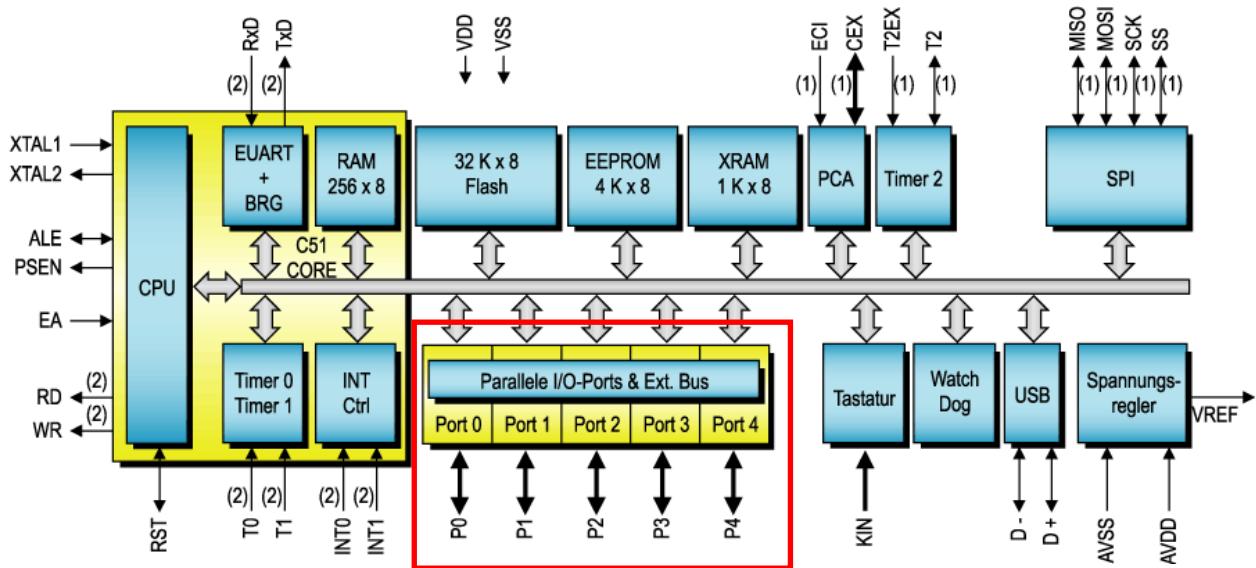
Erste Schritte: Operatoren

Beschreiben Sie die Wirkungen der folgenden Anweisungen.

- 1.6 $a = b \mid 0x0F;$
- 1.7 $\text{if } ((\text{taster} == 0) \&\& (\text{a} == 3)) \text{ b} = 7;$
- 1.8 $c = c >> 2;$
- 1.9 $\text{while } ((b \& 0x0F) == 0) \{a++;}$



6. Ausgaben Port:



6.1. Ausgabe Bit: LED an

Aufgabe: Erstellen Sie ein funktionsfähiges C- Programm (für RIDE inkl. Kommentaren) welches die LED D1 BITWEISE einschaltet:

Optional: Erweitern Sie das Programm so, dass alle 4 LEDs leuchten

6.2. Ausgabe Byte: LED an

Aufgabe: Erstellen Sie ein funktionsfähiges C- Programm (für RIDE inkl. Kommentaren) welches die LED D1 BYTEWEISE einschaltet:

Optional: Erweitern Sie das Programm so, dass alle 4 LEDs leuchten



6.3. Wechselblinker (Ausgabe Bit / Byte...)

Aufgabe: Erstellen Sie ein Struktogramm sowie ein funktionsfähiges C- Programm (für RIDE inkl. Kommentaren) welches die beiden LED's D2 und D3 BITWEISE nacheinander aufblinken lassen...

Optional: Ändern Sie das Programm so, dass die Ausgabe Byteweise erfolgt
Ändern Sie das Programm so, dass D1 und D4 ebenso wechselseitig blinken...



7. Debugger zur Fehlersuche bzw. Simulation

Grundeinstellung zum Start des Simulators: Options->Debug->Virtual Machine

Erstmalig:

- Oszillatoreinstellungen 12MHz
- Speichergröße OK...

oder



Debug -> Start

Fenster wählen

oder mit View -> Code

Disassemblerter Code + Hochsprache

oder mit View -> Main Registers

oder mit View -> Hardware

 Tile vertical

alle aktiven Fenster anordnen

**Aufbau des Debug-Bildschirms**

Symbolleiste mit Debug-Befehlen

Portsimulation

Programmcode in C oder Assembler

Programmcode mit zugehörigem Assemblercode

Blaue Balken: dieser Befehl wird als nächster ausgeführt.

Debuggen des Programmcodes (C oder Assembler) und des Disassembelten Codes möglich.

Prinzip: auf das zu debuggende Fenster klicken, dann Einzelschritt (Step) oder Automatik (Go)

Einzelschritt (Step)

In Unterprogramm oder Funktion springen:



Unterprogramm oder Funktion überspringen

**Automatik (Go)**

Fensterinhalte (Ports, Watch..) während des Programmablaufs aktualisieren:

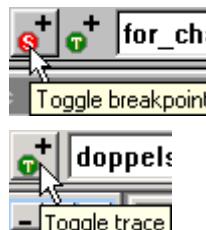


Geschwindigkeit einstellen:



Programmstart:

Haltepunkte setzen:



Tracepunkte setzen:



(an diesen Stellen soll mit Trace beobachtet werden)

Variablen, SFR etc beobachten

Watch-Fenster sichtbar mit View -> Watch

Watches	Value
p2.0	FALSE
p1	100 (0x64)

Rechter Mausklick im Watch-Fenster -> Add -> Variablenname eingeben

Rechter Mausklick auf Variable -> Evaluate -> Wert ändern -> Modify 2x klicken

Debugger beenden

Entweder durch Klick auf Debugger-Symbol

oder mit Debug -> Terminate



8. Clock-Controll = Zeiten vom at89c5131

Aus Datasheet at89c5131.pdf:

The AT89C5131 clock controller is based on an on-chip oscillator feeding an on-chip Phase Lock Loop (PLL). All the internal clocks to the peripherals and CPU core are generated by this controller.

The AT89C5131 X1 and X2 pins are the input and the output of a single-stage on-chip inverter (see Figure 5) that can be configured with off-chip components as a Pierce oscillator (see Figure 6). Value of capacitors and crystal characteristics are detailed in the section "DC Characteristics".

The X1 pin can also be used as input for an external 48 MHz clock.

The clock controller outputs three different clocks as shown in Figure 5:

- a clock for the CPU core
- a clock for the peripherals which is used to generate the Timers, PCA, WD, and Port sampling clocks
- a clock for the USB controller

These clocks are enabled or disabled depending on the power reduction mode as detailed in Section "Power Management", page 146.

Two clock sources are available for CPU:

- Crystal oscillator on X1 and X2 pins: Up to 32 MHz
- External 48 MHz clock on X1 pin

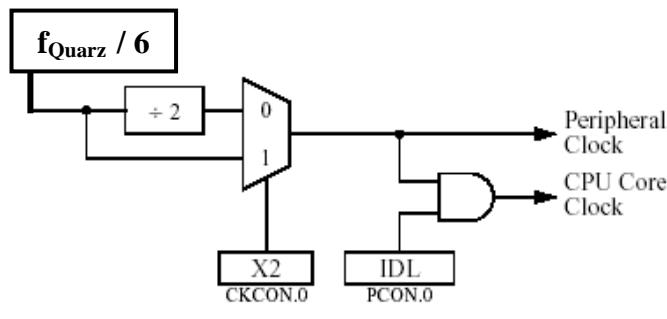


Table 28. CKCON0 (S:8Fh)
Clock Control Register 0

Bit Number	Bit Mnemonic	Description
7	-	Reserved The value read from this bit is always 0. Do not set this bit.
6	WDX2	Watchdog Clock This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect. Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
5	PCA2X	Programmable Counter Array Clock This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect. Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
4	SIX2	Enhanced UART Clock (Mode 0 and 2) This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect. Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
3	T2X2	Timer2 Clock This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect. Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
2	T1X2	Timer1 Clock This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect. Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
1	TOX2	Timer0 Clock This control bit is validated when the CPU clock X2 is set. When X2 is low, this bit has no effect. Clear to select 6 clock periods per peripheral clock cycle. Set to select 12 clock periods per peripheral clock cycle.
0	X2	System Clock Control bit Clear to select 12 clock periods per machine cycle (STD mode, $F_{CPU} = F_{PER} = F_{OSC}/2$). Set to select 6 clock periods per machine cycle (X2 mode, $F_{CPU} = F_{PER} = F_{OSC}$).

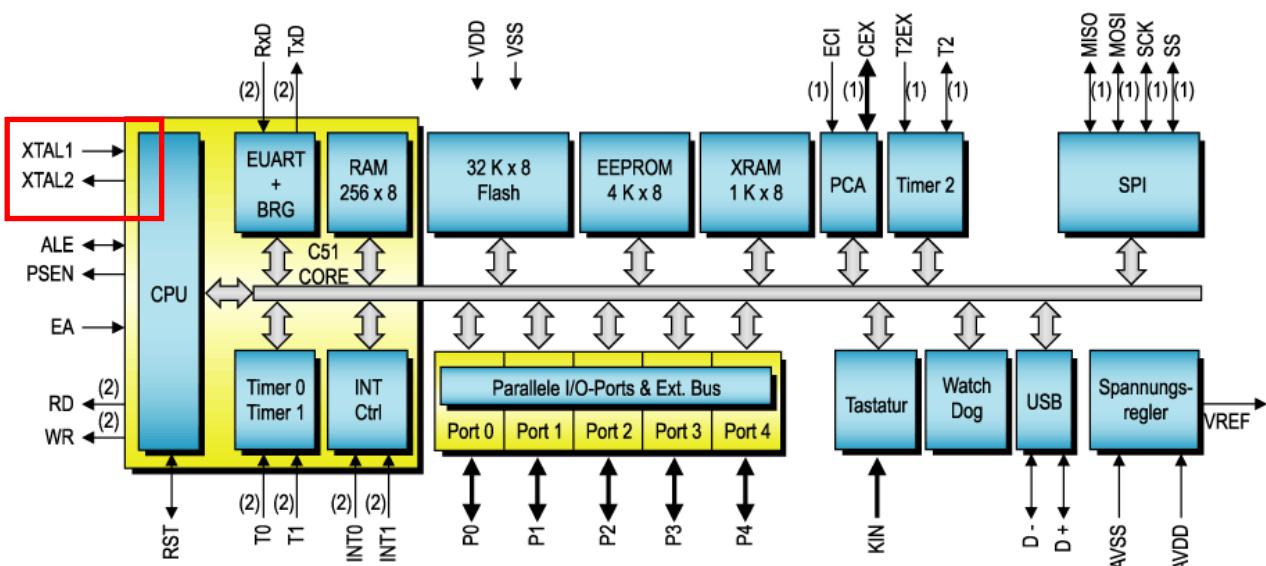
Reset Value = 0000 0000b

Im JDS-Board mit dem AT89C5131 befindet sich ein **12 MHz-Quarz**.

Resetzustand:

Nach einem Reset wird der Quarztakt durch _____ dividiert, da CKCON0 = 0 ist.

Der CPU-Core-Takt und der Peripherie-Takt ist 1 MHz, somit ergibt sich für einen Maschinenzyklus die Periodendauer von _____.





8.1. Wechselblinker im Sekundentakt

Aufgabe: Erstellen Sie ein Struktogramm sowie ein funktionsfähiges C- Programm (für RIDE inkl. Kommentaren) welches die beiden LED's D2 und D3 Byteweise nacheinander im Sekundentakt aufblinken lassen...

Benutzen Sie hierfür eine Funktion *Zeit* als FOR-Schleife, welche im Hauptprogramm mittels Werteübergabe (max. 16 Bit) aufgerufen wird.

Ermitteln Sie mit dem Debugger so genau wie möglich den Schleifenwert für 500ms.

Struktogramm:

Welchen Wert ermitteln Sie mit dem Debugger? _____

Optional: Ermitteln Sie im Schaltplan die Messpunkte D2, D3, GND um den „Schleifenwert“ mittels einem DS-Oszilloskop zu messen... Time: 200ms, CH1: 200mV, Cursor...



Welchen Wert ermitteln Sie mittels Messung (nur Labor!)? _____



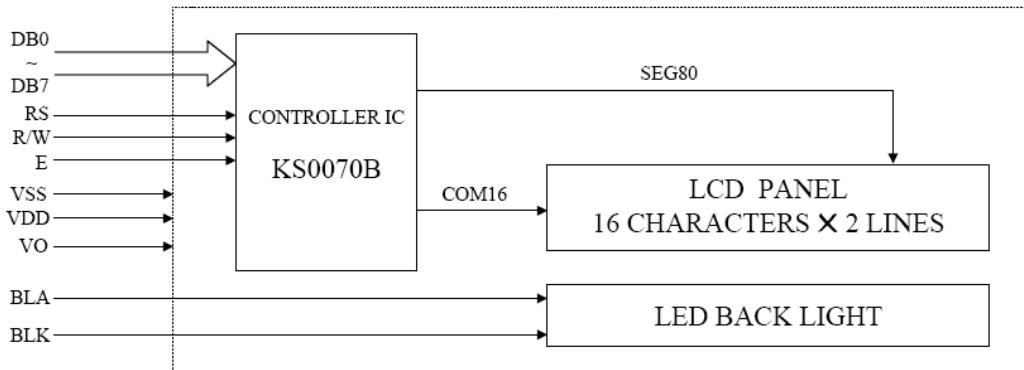
9. LCD-Ausgabe

9.1. LCD, Auszug Datenblatt

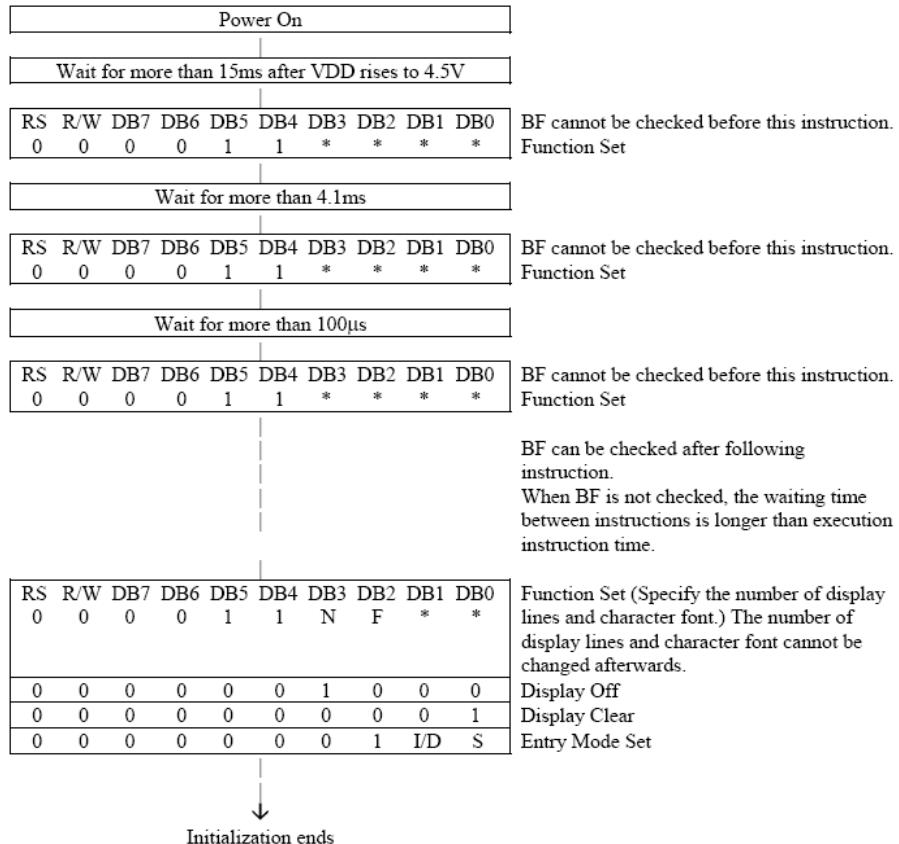
Das LC-Display zur Ausgabe besitzt einen eigenen µ-Controller inkl. RAM-Speicher. Deshalb wird dieses betrachtet, wie wenn wir einen „externen Speicher“ beschreiben oder auslesen wollen.

■ BLOCK DIAGRAM

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
VSS	VDD	VO	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	BLA	BLK



◆ Initializing by Instruction



DISPLAY DATA RAM ADDRESS MAP

Characters	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
First line	00H	01H	02H	03H	04H	05H	06H	07H	08H	09H	0AH	0BH	0CH	0DH	0EH	0FH
Second line	40H	41H	42H	43H	44H	45H	46H	47H	48H	49H	4AH	4BH	4CH	4DH	4EH	4FH



Die zugehörigen Funktionen um das LCD anzusteuern befinden sich in der LCD.c bzw. LCD.h Datei. DIESE BEIDEN DATEIEN MÜSSEN IMMER IN DEN PROJEKTORDNER KOPIERT WERDEN!

```
extern void initlcd (void);                                // Initialisierung
extern void loeschenlcd (void);                            // LCD löschen
extern void textlcd (unsigned char *text, unsigned char zeile); // Textausgabe in Zeile 1 bis 4
extern void definierelcdsymbol (unsigned char pixelprozeile [8],unsigned char adr);

                                // Definition von max 7 eigenen Zeichen Adr 1 bis 7
extern void LCDbefehl (unsigned char befehl);           // Ausgabe von Befehlen extern
void charlcd (unsigned char zeichen);                   // Ausgabe eines Zeichens an die
aktuelle Cursorposition
extern void cursorpos (unsigned char position);        // Setzen der Cursorposition
```

Als weitere Möglichkeit zur Ausgabe gibt uns der C-Compiler mit der Standard IO-Funktion printf. ACHTUNG -> Speicherfresser! Hier ein Auszug aus dem Datenblatt:

7.2.5 sprintf and printf

Printf function invokes sprintf.

The usual formatting rules are observed.

To construct the formatted element, the sprintf function requires a stack buffer. The buffer size is set by SIZEBUFPRTNTF NUMBER. The number is initialised at 25 in the *RC51S.LIB* and *RC51T.LIB* (SMALL and TINY models) and at 50 in the *RC51L.LIB* (LARGE and HUGE models). It is possible to change the value by re-declaring another value in one file only (see DEFINEJOKER control).

Example:

RC51 serie.c LARGE OE defnp=SIZEBUFPRTNTF/60

7.2.5.2 Using (s)printf with floating point variables

Sprintf and scanf are available in two forms:

- The standard form that accepts all formats specified by the ANSI standard. This is placed in the *RC51Fx.LIB* libraries.

- A simplified form (with no format for floating point numbers) placed in the *RC51x.LIB* libraries.

If floating-point numbers are not used, *RC51Fx.LIB* libraries are not linked with the application.

Note that to format floating-point numbers, basic operations (+ - * / etc.) between floating-point numbers are used.

As for the floating point math's facilities, a slight « adaptation » of ANSI standard has been used, in order to facilitate the implementation of the different formats.

For both printf and scanf, the I specifier indicates a double type (or a pointer to a double) and L indicates a long double variable (or a pointer to a long double variable).

Example:

```
printf ("%f,%lg,%10Lg",f1,db1,ldb1);
```

In the above example:

f1 is a float type variable,

db1 is a double type variable,

ldb1 is a long double type variable

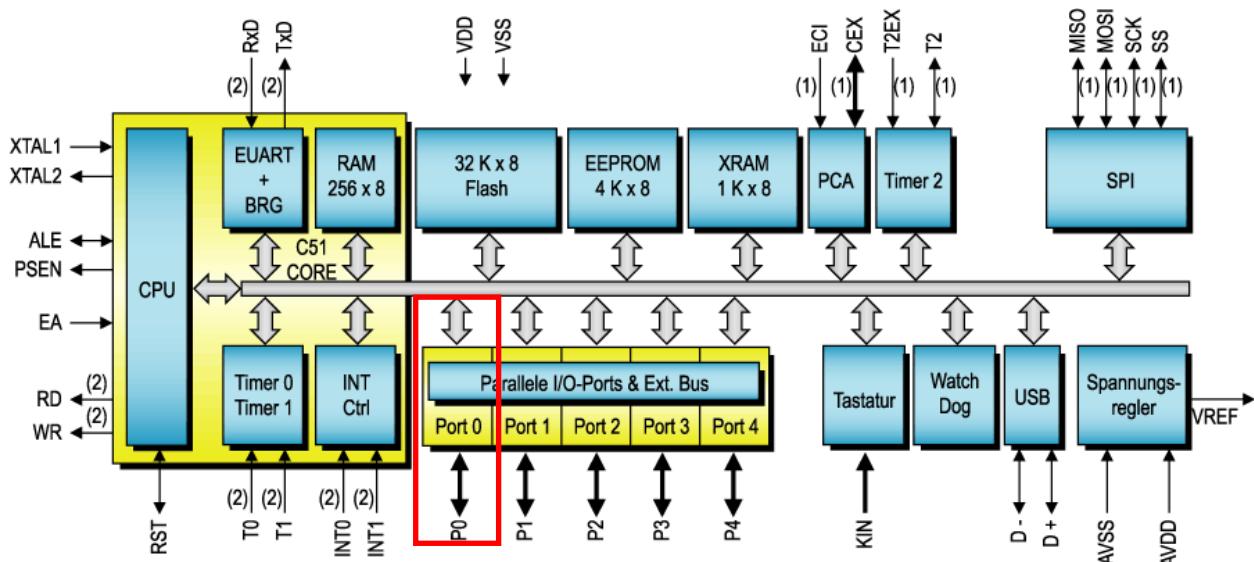
```
sprintf (buf,"TGI-J1, %2d", Klassenzahl);
```



9.2. LCD-Funktionen, lcd.c

In der Datei LCD.c sind alle notwendigen Funktionen zur Ansteuerung des LCD an einen bestimmten Portausgang (hier P0) hinterlegt. Deshalb muss diese Datei in jedes zukünftige Projekt kopiert und eingebunden werden, damit die Funktionen aufgerufen werden können! Des Weiteren werden Standardfunktionen der Entwicklungsumgebung benutzt weshalb die stdio.h im Programmheader eingebunden werden muss.

```
#include "lcd.c"                                // LCD-Anzeige
#include "stdio.h"                               // sprintf...
```



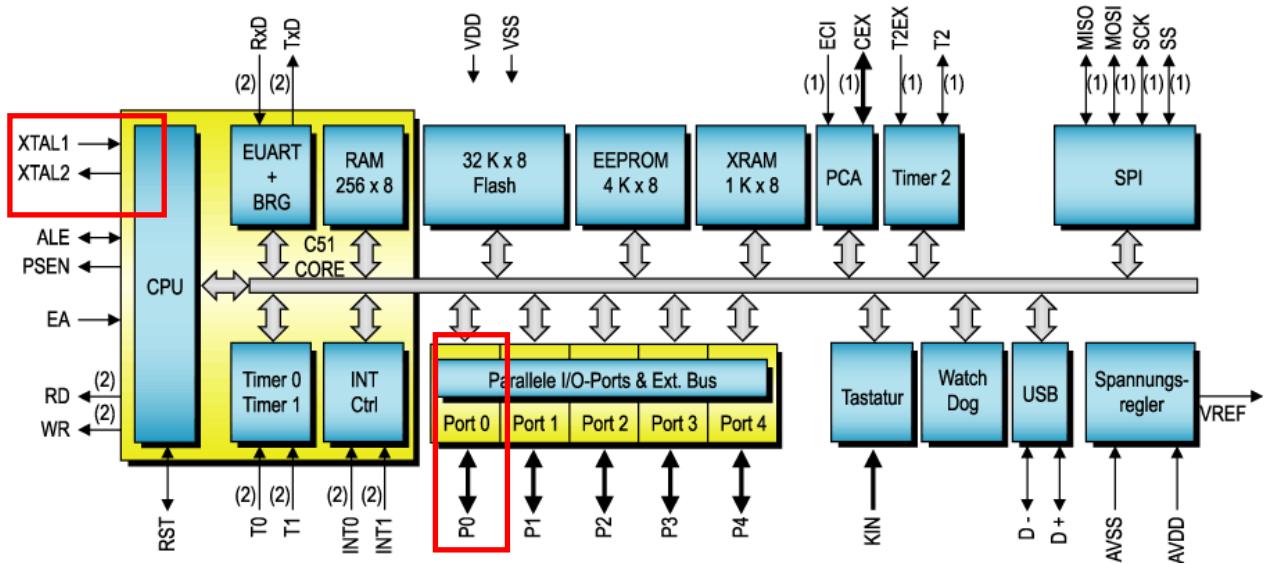
9.3. LCD-Ausgabe nach Zeit

Aufgabe: Erstellen Sie ein Struktogramm sowie ein funktionsfähiges C-Programm (für RIDE inkl. Kommentaren) welches zu Programmbeginn einmalig für 3 sec. Den Text *JDS-Rastatt, T-Gymn.* In Zeile 1 und 2 ausgibt und danach ständig den Klassennamen inkl. Schülerzahl als Variable (Zeile1) und Schülernamen (Zeile2) ausgibt.

Struktogramm:

Optional:

Erweitern Sie das Programm so, dass die Ausgabe nach drücken der Taste 1 erfolgt.

**Projektidee: Quarzuhr****P1: Programmierung einer einfachen Uhr mit Warteschleife und LCD-Ausgabe**

Erstellen Sie ein Struktogramm in Unterprogrammtechnik sowie mit RIDE ein C-Programm inkl. Kommentare mit dem in Zeile 1 *Uhr mit Delay* und die Uhrzeit im Format std:min:sek auf der LCD, Zeile 2 ständig sekundenweise ausgegeben wird. Dabei soll der Sekundentakt möglichst genau ermittelt werden und mit der LED d3 sichtbar sein. (Messung Labor?). Die Unterprogramme lauten:

Uhr(); Ausgabe_LCD(); Zeit();

Struktogramm: LCD_Uhr_Delay Main:



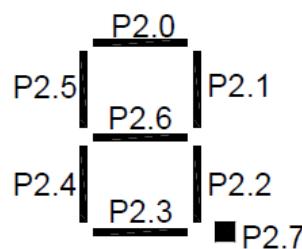
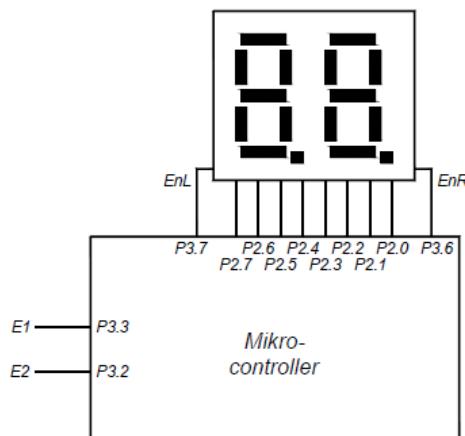
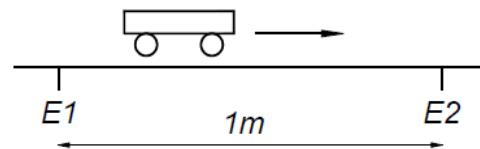
Struktogramm: Uhr:

Struktogramm: Ausgabe_LCD:



Ganzheitliche Übung: Zeitmessung

Die Zeit, die ein Wagen zum Durchlaufen einer Strecke benötigt, soll ermittelt werden. Dazu wurden 2 Sensoren E1 und E2, die bei Vorbeifahren des Wagens jeweils ein Signal (high) erzeugen, im Abstand von einem Meter installiert. Während der Wagen zwischen E1 und E2 fährt bleibt die Signallage unverändert. Bei Erreichen des E2 wird auch dieses Signal high. Die angezeigte Zeit kann mit einer vorgegebenen Zeitfunktion Zeit(98763) zwischen 1-34 sec. liegen.



Zuordnung der Segmente zu den Bits von Port 2

Am Port 2 des Mikrocontrollers ist eine zweistellige 7-Segment-Anzeige angeschlossen. Mit dem Signal EnR=1 (P 3.6) wird die rechte 7-Segment-Anzeige aktiviert, mit EnL=1 (P 3.7) die linke.

Erstellen Sie ein Struktogramm für die Funktion Main und Ergänzen das C-Programm so, dass die Aufgabe realisiert werden kann. Das Programm ist in Unterprogrammtechnik aufgebaut und hält folgende Funktionen:

Zaehl(); = Enthält den Zeitzähler

BCD(); = Enthält die Codierung der Anzeigen TIPP: Mehrfachverzweigung

Ausgabe(); = Enthält die Ansteuerung der Anzeige

Zeit(); = Warteschleife ungefähr Sekudentakt

Struktogramm Main:

**Programmauszug:**

```
#include "at89c5131.h"
#include "stdio.h"

sbit at P3_2 E2;
sbit at P3_3 E1;
sbit at P3_6 EnR;
sbit at P3_7 EnL;

unsigned char ZaehlerE, ZaehlerZ, ZahlE_7, ZahlZ_7; //Variablendeclaration

void zaehl (void)
{
}

void BCD (void)
{
```



{

```
void ausgabe (void)
{
```

{

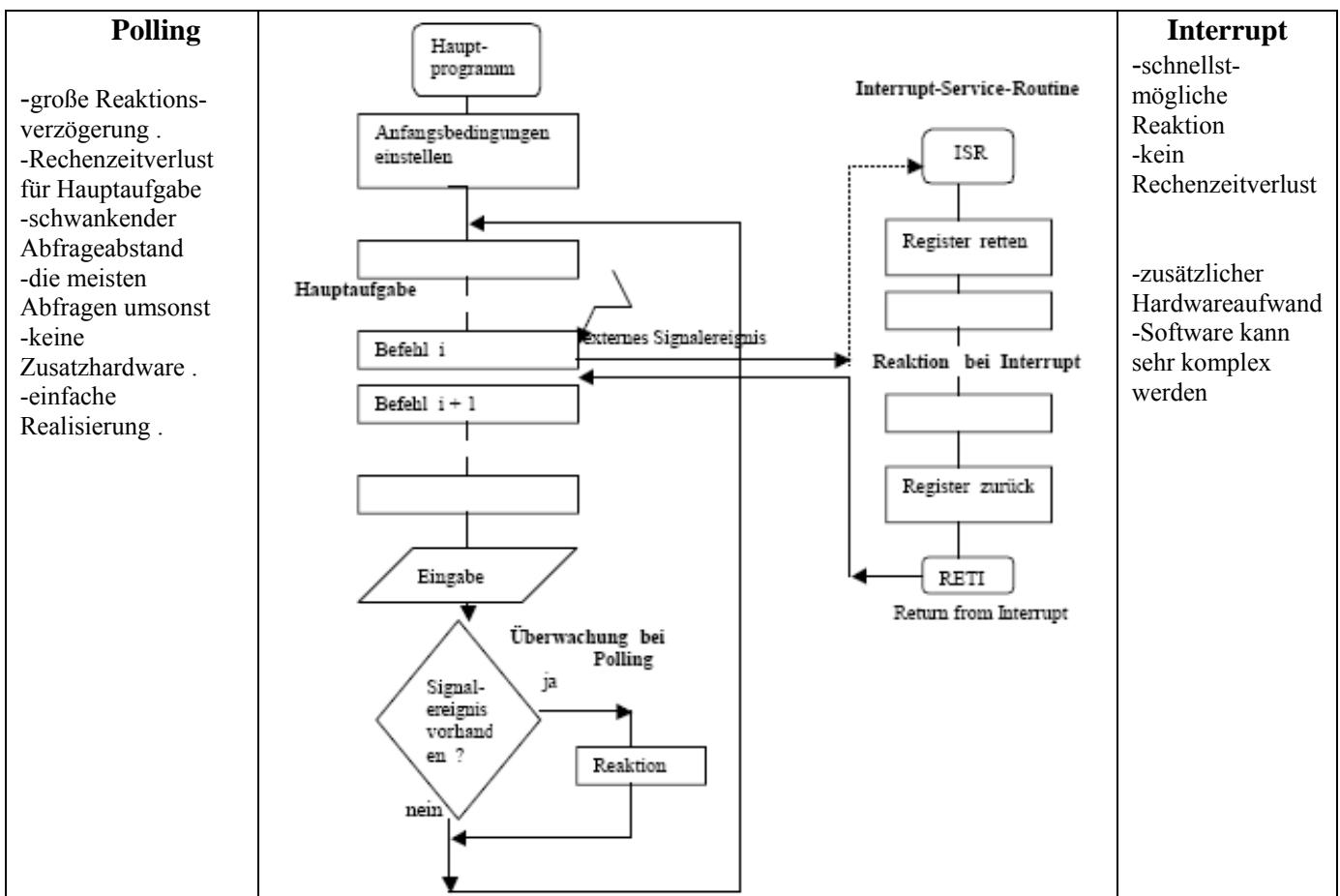
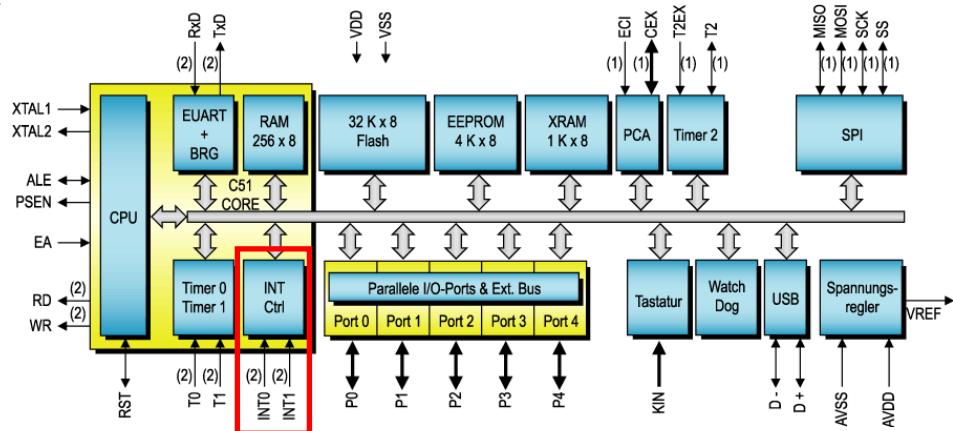
```
void main (void)
{
```

{

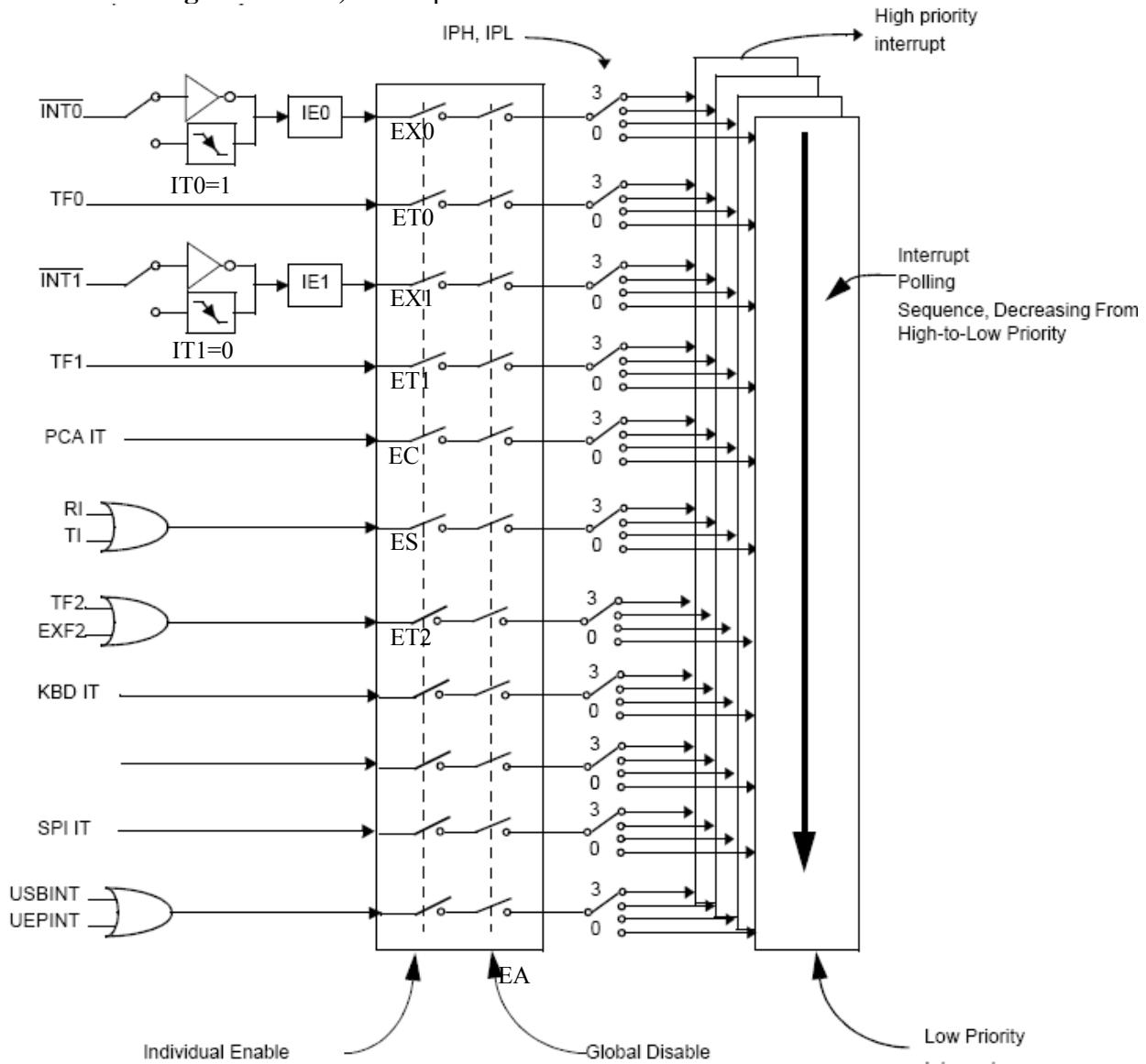
```
// Ende main
```



10. Interrupt



Nr.	Interrupt	Einsprung-adresse	Interrupt-Anforderungs-Bit
0	Externer Interrupt 0	0003h	IE0
1	Timer0 - Überlauf	000Bh	TF0
2	Externer Interrupt 1	0013h	IE1
3	Timer1 - Überlauf	001Bh	TF1
4	Serieller Port	0023h	TI oder RI
5	Timer2 – Überlauf / ext. Interrupt2	002Bh	TF2 oder EXF2
6	PCA	0033h	CF + CCFn (n=0-4)
7	Keyboard	003Bh	KBDIT
9	SPI	004Bh	SPIIT
10	USB	006Bh	USB

**Datenblattauszüge IR/Timer, Interrupt Sources:****Interrupt-Prioritäten:**

IPL0 - Interrupt Priority Register (B8h)

7	6	5	4	3	2	1	0
-	PPCL	PT2L	PSL	PT1L	PX1L	PT0L	PX0L

IPH0 - Interrupt Priority High Register (B7h)

7	6	5	4	3	2	1	0
-	PPCH	PT2H	PSH	PT1H	PX1H	PT0H	PX0H

Table 61. Priority Level Bit Values

IPH.x	IPL.x	Interrupt Level Priority
0	0	0 (Lowest)
0	1	1
1	0	2
1	1	3 (Highest)

**Übung: Interrupt Extern 0 = Taster S2, IR-Aktiv**

Erstellen Sie ein Struktogramm sowie ein C-Programm für RIDE, womit der externe Interrupt 0, Flankengesteuert, Priorität 0 (PIN 3.2) aktiviert wird und bei Betätigung des dort angeschlossenen Schalters S2 am LCD der Text *Interrupt aktiv* ausgegeben wird.

Welche SFR/Bits müssen gemäß Datenblattauszug konfiguriert werden, damit der IR nutzbar wird?

Optional: Schreiben Sie ein Programm welches die Klassenbezeichnung sowie die Anzahl der Schüler am LCD ausgibt. Die Anzahl der Schüler soll mit Betätigung der Taste S2 eingestellt werden.

Wie sieht das Struktogramm Main aus?

Wie sieht das Struktogramm ISR aus?



Programm: IR-Aktiv: Ergänzen Sie den Code an den fehlenden Stellen:

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                       // LCD-Anzeige
#include "stdio.h"                                     // sprintf...

//Konstanten- Variablendeklaration
unsigned char buf [16];                             // Text-Buffer für LCD
unsigned int j,z,t;

// ----- Funktionen -----
void zeit (unsigned int sec)
{
    for (z= sec; z != 0; z--);           // Zeitverzögerungsschleife 1000=??
}

//----- Interrupt-Service-Routine-----
void IR_Taster (void) interrupt 0      // Interrupt EX0
{
    EA = 0;                               // Interrupt deaktivieren
    sprintf (buf,"Interrupt aktiv");
    textlcd (buf,2);

    zeit(50000);
    EA = 1;                               // Interrupt wieder aktiv
}

//----- Hauptprogramm
void main (void)
{
    initlcd();                           // LCD initialisieren

    IT0 = 1;                            // ext. Interrupt0
    EX0 = 1;                            // ext. Interrupt0 freigeben
    EA = 1;                             // globale Interrupt Freigabe

    while (1) // Endlosschleife Hauptprogramm -----
    {
        sprintf (buf,"JDS-Rastatt");
        textlcd (buf,1);
        sprintf (buf," Techn. Gymn. ");
        textlcd (buf,2);
        zeit(20000);

    } // von while
} // von main
```

**Übung: Interrupt Extern 0 = Taster S2, Klassenbezeichnung**

Erstellen Sie ein Struktogramm sowie ein C-Programm für RIDE, womit der externe Interrupt 0, Flankengesteuert, Priorität 1 (PIN 3.2) aktiviert wird. Bei Betätigung des dort angeschlossenen Schalters S2 am LCD, Zeile 2 die Variable (*Schülerzahl, Hex-Darstellung*) nach dem Text *Schülerzahl:* ausgegeben werden. In der Zeile ist die Klassenbezeichnung auszugeben. Ist der IR nicht aktiv, so ist auf der Zeile2 der Text *weiter mi S2* auszugeben.

Welche SFR/Bits müssen gemäß Datenblattauszug konfiguriert werden, damit der IR nutzbar wird?

Wie sieht das Struktogramm Main aus?

Wie sieht das Struktogramm ISR aus?



Programm: Schülerzahl: Ergänzen Sie den fehlenden Code:

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

//Konstanten- Variablendeclaration
unsigned char buf [16];                                // Text-Buffer für LCD
unsigned int j,z,t;

// ----- Funktionen -----
void zeit (unsigned int sec)
{
    for (z= sec; z != 0; z--);                         // Zeitverzögerungsschleife 1000=??
}

//----- Interrupt-Service-Routine-----
void IR_Taster (void) interrupt 0                      // Interrupt EX0
{
    EA = 0;                                            // Interrupt deaktivieren
    sprintf (buf,"Schülerzahl:" %2x, Schülerzahl);
    textlcd (buf,2);

    zeit(50000);
    EA = 1;                                            // Interrupt wieder aktiv
}

//----- Hauptprogramm
void main (void)
{
    initlcd();                                         // LCD initialisieren

    IT0  = 1;                                         // ext. Interrupt0
    EX0  = 1;                                         // ext. Interrupt0 freigeben
    EA   = 1;                                         // globale Interrupt Freigabe

    while (1) // Endlosschleife Hauptprogramm -----
    {
        sprintf (buf,"TGI J1");
        textlcd (buf,1);
        sprintf (buf," weiter mit S2 ");
        textlcd (buf,2);
        zeit(20000);

    } // von while
} // von main
```

**Übung: externer Interrupt: Zähler 1-9**

Erstellen Sie folgendes Struktogramm sowie ein C-Programm für RIDE:

Bei Beginn des Programms wird der Text „JDS-Rastatt“ in Zeile 1 und „Techn. Gymn.“ in Zeile 2 ausgegeben.

Durch Auslösen des externen Interrupt 0 (Taster S2, Flankengesteuert, Priorität 2) wird jeweils am LCD (Zeile 2) und an der 7 Segment-Anzeige ein Zähler von 1-9 realisiert. Nach Erreichen der Zahl 9 soll der Zähler von neuem beginnen.

Struktogramm main:

Struktogramm ISR:

**P2: Programmierung einer Uhr mit Warteschleife, LCD-Ausgabe, Externer IR**

Ergänzen Sie die Übung P1 *Projektaufgabe Uhr* wie folgt:

Übung P2: Interrupt Extern 0 = Taster S2, IR-aktivieren

Erstellen Sie ein Struktogramm sowie ein C-Programm für RIDE inkl. Kommentare, womit der externe Interrupt 0, Flankengesteuert, Priorität 1 aktiviert wird und bei Betätigung des dort angeschlossenen Schalters S2 einmalig am LCD der Text *Interrupt aktiv* ausgegeben wird.

Wie sieht das Struktogramm Main aus?

Wie sieht das Struktogramm ISR aus?

**Programm P2: IR aktivieren:** Ergänzen Sie den Code an den fehlenden Stellen:

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

//Konstanten- Variablendeclaration
sfr at P2 LED;                                         //Sekundenblinker
unsigned char buf [16];                                 // Text-Buffer für LCD
unsigned long z;
unsigned char sec, min, std ;

//----- Interrupt-Service-Routine-----

// ----- Funktionen -----
void zeit (unsigned long sec)
{
...
}

void Uhr (void)
{
...
}

void Ausgabe_LCD (void)
{
...
}

//----- Hauptprogramm
void main (void)
{
    //IR aktivieren

initlcd();                                              // LCD initialisieren
sprintf (buf,"Uhr mit Delay");
textlcd (buf,1);           //Ausgabe erste Zeile
while (1) // Endlosschleife Hauptprogramm -----
{
    Uhr();
    Ausgabe_LCD();
    LED = 0x2f;
    zeit (99827);        //Schleife 1 sec!? Hardware
    LED = 0x0f;
}
}
```

**P2: Programmierung einer Uhr mit Warteschleife, LCD-Ausgabe, Externer IR**

Ergänzen Sie die Übung P1 *Projektaufgabe Uhr* wie folgt:

Übung P2: Interrupt Extern 0 = Taster S2, Ausgabe Anwender

Erstellen Sie ein Struktogramm sowie ein C-Programm für RIDE inkl. Kommentare, womit der externe Interrupt 0, Flankengesteuert, Priorität 3 aktiviert wird und durch Betätigung des dort angeschlossenen Schalters S2 beim

1x drücken: am LCD den Text *Std stellen* ausgibt.

Wie sieht das Struktogramm Main aus?

Wie sieht das Struktogramm ISR aus?



Programm P2: Ausgabe Anwender: Ergänzen Sie den Code an den fehlenden Stellen:

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                       // LCD-Anzeige
#include "stdio.h"                                      // sprintf...

//Konstanten- Variablendeklaration
sfr at P2 LED;                                         //Sekundenblinker
unsigned char buf[16];                                  // Text-Buffer fuer LCD
unsigned long z;
unsigned char sec, min, std;

//----- Interrupt-Service-Routine-----


// ----- Funktionen -----
void zeit (unsigned long sec)
{
...
}

void Uhr (void)
{
...
}

void Ausgabe_LCD (void)
{
...
}

void main (void)                                       //----- Hauptprogramm
{
    initlcd();                                         // LCD initialisieren
    sprintf (buf,"Uhr mit Delay");
    textlcd (buf,1);                                 //Ausgabe erste Zeile
    while (1) // Endlosschleife Hauptprogramm -----
    {
        Uhr();
        Ausgabe_LCD();
        LED = 0x2f;
        zeit (99827);                               //Schleife 1 sec!? Hardware
        LED = 0x0f;
    }
}
```

**P2: Programmierung einer Uhr mit Warteschleife, LCD-Ausgabe, Externer IR**

Ergänzen Sie die Übung P1 *Projektaufgabe Uhr* wie folgt:

Übung P2: Interrupt Extern 0 = Taster S2, Anwender Uhr stellen

Erstellen Sie ein Struktogramm sowie ein C-Programm für RIDE inkl. Kommentare, womit der externe Interrupt 0, Flankengesteuert, Priorität 3 aktiviert wird und durch Betätigung des dort angeschlossenen Schalters S2 beim

1x drücken: am LCD den Text *Std stellen* ausgibt.

2x drücken: am LCD den Text *Min stellen* ausgibt.

3x drücken: am LCD den Text *sec stellen* ausgibt.

Wie sieht das Struktogramm Main aus?

Wie sieht das Struktogramm ISR aus?



Programm P2: Anwender Uhr stellen: Ergänzen Sie den Code an den fehlenden Stellen:

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                       // LCD-Anzeige
#include "stdio.h"                                      // sprintf...

//Konstanten- Variablendeclaration
sfr at P2 LED;                                         //Sekundenblinker
unsigned char buf [16];                                 // Text-Buffer für LCD
unsigned long z;
unsigned char sec, min, std ;
//----- Interrupt-Service-Routine-----


// ----- Funktionen -----
void zeit (unsigned long sec)
{
...
}

void Uhr (void)
{
...
}

void Ausgabe_LCD (void)
{
...
}

void main (void)                                     //----- Hauptprogramm
{
    //IR aktivieren

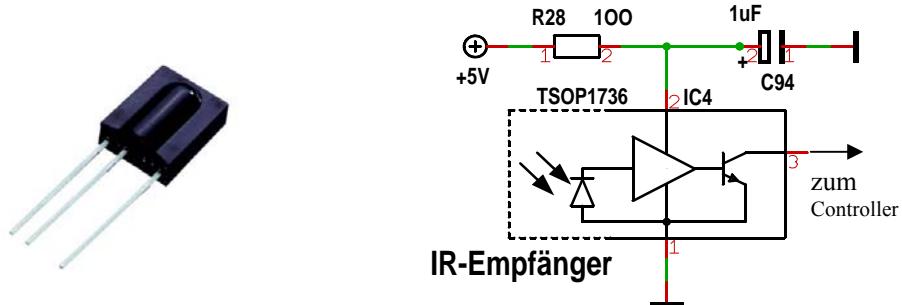
initlcd();                                              // LCD initialisieren
sprintf (buf,"Uhr mit Delay");
textlcd (buf,1);           //Ausgabe erste Zeile
while (1) // Endlosschleife Hauptprogramm -----
{
    Uhr();
    Ausgabe_LCD();
    LED = 0x2f;
    zeit (99827);        //Schleife 1 sec!? Hardware
    LED = 0x0f;
}
}
```



10.1. IR-Fernbedienung mit IR-EX1:

10.1.1. Empfang von IR-Fernbedienung...

Eine IR-Fernbedienung, zur Steuerung und Parametrierung eines Mikrocontrollers, stellt eine komfortable Schnittstelle zwischen Benutzer und Steuerung dar. Gleichzeitig reduziert sich der Hardwareaufwand, da anstatt einer Tastatur nur ein IR-Empfänger vorgesehen werden muss. Um eine sichere Datenübertragung zu ermöglichen, wird nicht einfach Licht AN/AUS übertragen, sondern es wird ein moduliertes und codiertes Signal gesendet.



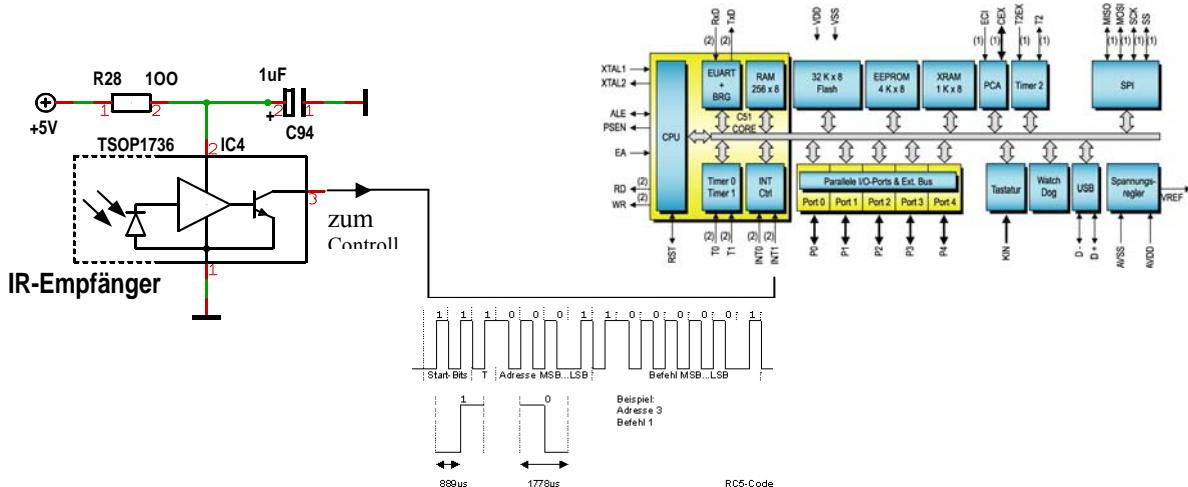
Leider gibt es bei den Signalen/Protokollen eine Vielzahl von unterschiedlichen Formaten. Die Zustände 0 und 1 sind dabei entweder als Bitwechsel oder durch das Impuls/Pausenverhältnis definiert. Um dies zu ermitteln, messen wir diesen Code mit einem DSO (Digitales-Speicher-Oszilloskop).

10.1.2. Universal IR-Fernbedienung Bsp.: hama; Einstellung Philips RC5-Code...



- Taste Setup 3 sec. bis LED dauerhaft leuchtet
- Quellentaste drücken: TV
- Code aus Liste eingeben: 0200
- Taste SHIFT kurz und dann OK zum speichern
- SHIFT Taste erneut kurz drücken

10.1.3. Technologieschema der seriellen Datenübertragung





10.1.4. Bsp.: RC5-Code

Das Protokoll ermöglicht das Adressieren von 32 Geräte mit je 64 Befehlen. So kann man problemlos den Fernseher einschalten, ohne dass gleichzeitig alle anderen RC5-Geräte mit eingeschaltet werden.

Der **Aufbau des Codes** besteht aus:

2 Startbits

1 Togglebit (wechselt bei jeder Übertragung, um das Festhalten einer Taste zu erkennen)

5 Adressbits

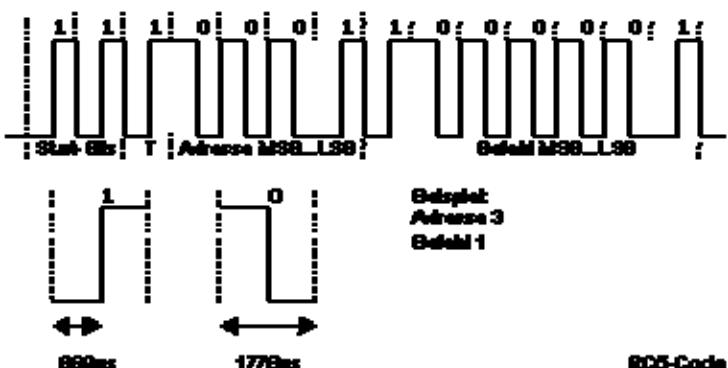
6 Befehlsbits

Der RC5-Code verwendet ein biphasenkodiertes Format, d.h. ein Bitzustand wird durch einen Bitwechsel definiert:

Manchester Code:

Wechsel 0 → 1 = logische 1

Wechsel 1 → 0 = logische 0.



Die Gesamtlänge eines so kodierten Bits beträgt 1,778ms.

Die Gesamtzeit beträgt daher ($14 \times 1,778\text{ms} = 24,889\text{ms}$) pro Übertragung.

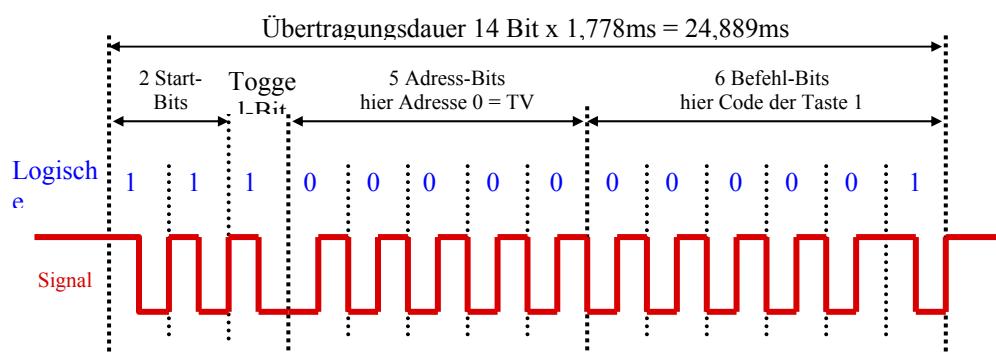
Zwischen zwei Übertragungen ist eine Pause von mindestens 114ms.

Reale Oszilloskopbilder RC5-Code

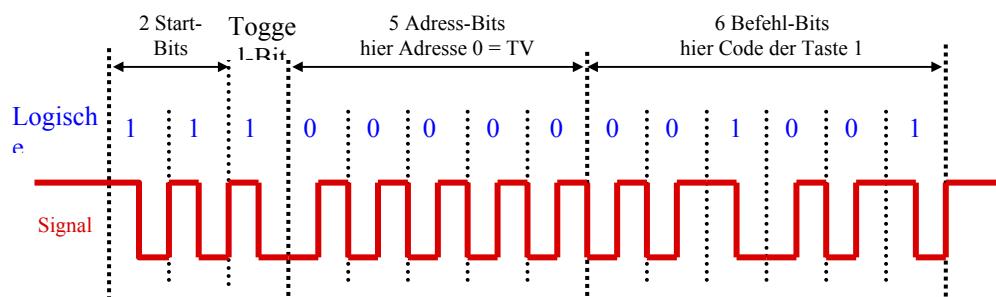
Das Signal am IR-Empfänger ist invertiert, denn hier bedeutet



An der Fernbedienung: Taste 1 gedrückt, Einstellung TV



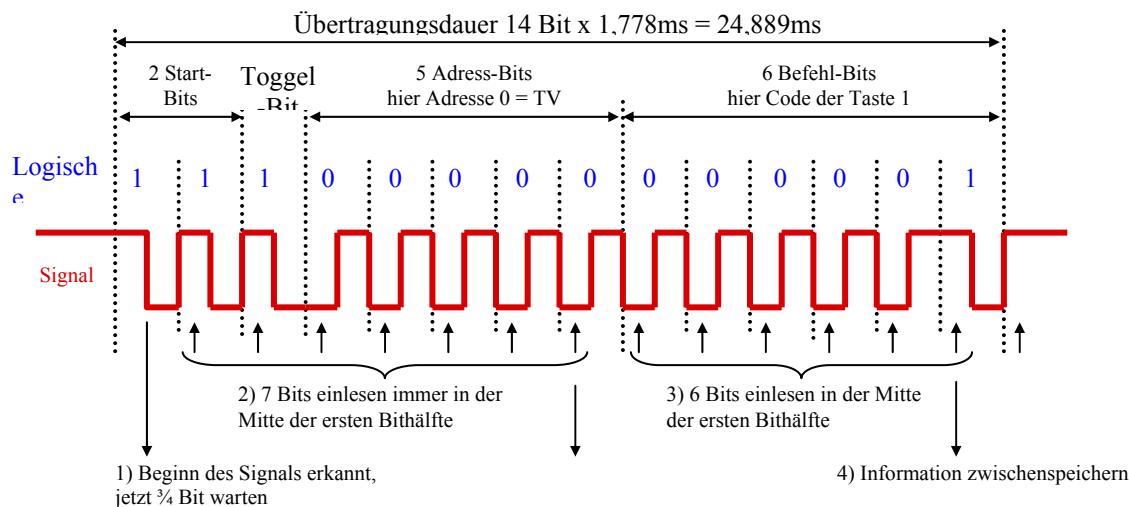
Taste ? _____ ? gedrückt



**Aufgabe: IR-Fernbedienung mit IR-EX1:**

Erstellen Sie folgendes Struktogramm sowie ein C-Programm in RIDE inkl. Kommentaren, welches bei Beginn des Programms den Text „Digitaluhr“ in Zeile 1 und „Uhr stellen.“ in Zeile 2 ausgibt.

Durch Auslösen des externen Interrupt 1 (IR-Fernbedienung, Flankengesteuert, Priorität 3) wird am LCD (Zeile 2) der Text „FB-Code: xx“ ausgegeben

**Struktogramm Main:**

IF_IR_EX1 = FB-Code Main
EX1 = 1
EA = 1
IT1 = 0 !! Obwohl Datenblatt = 1 zeigt!
IP= 3
initLCD
LCD: Digitaluhr
LCD: Uhr stellen
Endlos
LCD-Text: "FB-Code:", aktueller Variablenwert

Struktogramm ISR:

IF_IR_EX1 = FB-Code ISR
EX1 = 0 (IR-aus: Da Flankengesteuert wird bei 14-Bit-Signal jeweils IR ausgelöst!)
Variable löschen: RC5Code = 0;
7 3/4-Bit überspringen = zeit(1710)
For-Schleife 6x für die letzten 6 Bits
Schieben um für neues Bit Platz machen: RC5Code <<1
BIT (IRInput) einlesen -> mit Byte Variable RC5Code verordnen (nichts überschrieben wird) und zu-
Hilfsbit für Zeitmessung löschen: oszi = 0
Wartezeit halbe Bitlänge = 0,87ms -> zeit (108)
Hilfsbit für Zeitmessung setzen: oszi = 1
Wartezeit halbe Bitlänge = 0,87ms -> zeit (108)
IR wieder aktivieren EX=1

**Ergänzen Sie den fehlenden Code:**

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

#define IRinput P3_3;                                    // Infrarot-Empfänger
sbit at P1_1 oszi;                                     // Hilfsbit für Oszi zur Zeitmessung
unsigned char buf[16];                                  // Text-Buffer für LCD
unsigned char RC5Code, z, T, t1,j;

void zeit (unsigned long sec)
{
    for (z= sec; z != 0; z--);      // Zeitverzögerungsschleife
}

//IRService-Routine RC 5 Code: Original-----
```

```
void IReinlesen (void) interrupt 2          // Interrupt EX1
{

}

void main (void)
{
    initlcd();                                         // LCD initialisieren
    sprintf (buf," Digitaluhr");
    textlcd (buf,1);
    sprintf (buf,"Uhr stellen");
    textlcd (buf,2);

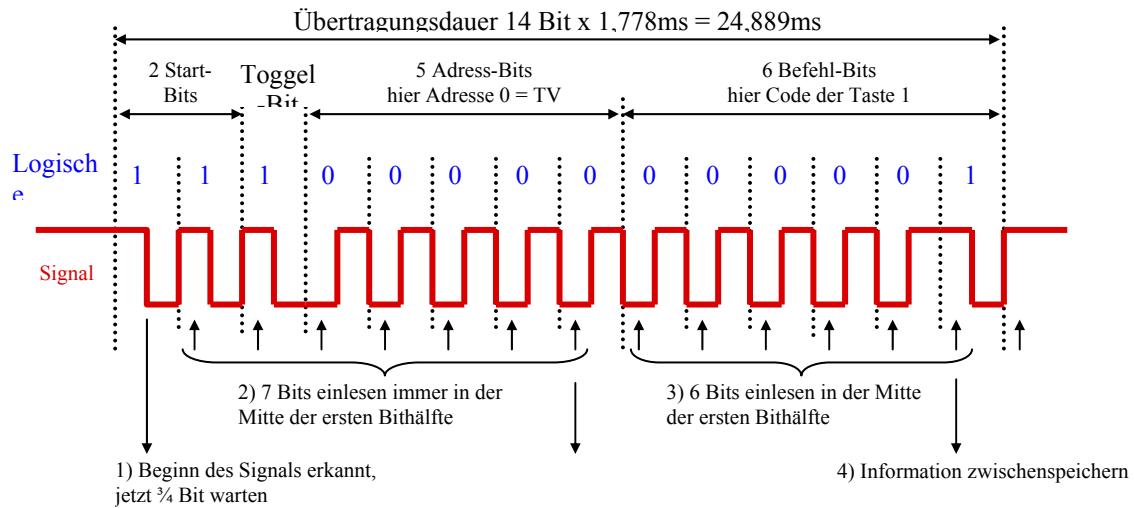
    while(1)                                         // Endlosschleife
    {

    } // von while
}// von main
```

**Aufgabe: IR-Fernbedienung mit IR-EX1:**

Erstellen Sie folgendes Struktogramm sowie ein C-Programm in RIDE inkl. Kommentaren, welches bei Beginn des Programms den Text „Digitaluhr“ in Zeile 1 und „Uhr stellen.“ in Zeile 2 ausgibt.

Durch Auslösen des externen Interrupt 1 (IR-Fernbedienung, Flankengesteuert, Priorität 3) wird am LCD (Zeile 2) der Text „FB-Code: xx“ sowie der zugehörige Zahlenwert in Hex ausgegeben und immer aktualisiert:

**Struktogramm Main:****Struktogramm ISR:**

**Ergänzen Sie den fehlenden Code:**

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

#define IRinput P3_3;                                     // Infrarot-Empfänger
sbit at P1_1 oszi;                                      // Hilfsbit für Oszi zur Zeitmessung
unsigned char buf[16];                                    // Text-Buffer für LCD
unsigned char RC5Code, z, T, t1,j;

void zeit (unsigned long sec)
{
    for (z= sec; z != 0; z--);      // Zeitverzögerungsschleife
}

//IRService-Routine RC 5 Code: Original-----
```

```
void IReinlesen (void) interrupt 2          // Interrupt EX1
{

}

void main (void)
{
    initlcd();                                         // LCD initialisieren
    sprintf (buf," Digitaluhr");
    textlcd (buf,1);
    sprintf (buf,"Uhr stellen");
    textlcd (buf,2);

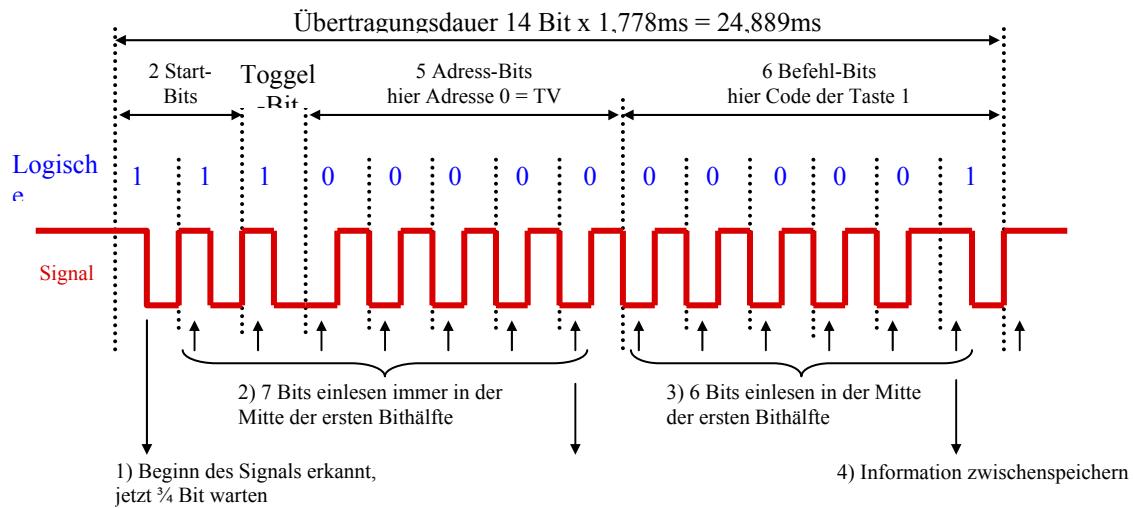
    while(1)                                         // Endlosschleife
    {

    } // von while
}// von main
```

**Aufgabe: IR-Fernbedienung mit IR-EX1:**

Erstellen Sie folgendes Struktogramm sowie ein C-Programm in RIDE inkl. Kommentaren, welches bei Beginn des Programms den Text „Digitaluhr“ in Zeile 1 und „Uhr stellen.“ in Zeile 2 ausgibt.

Durch Auslösen des externen Interrupt 1 (IR-Fernbedienung, Flankengesteuert, Priorität 3) wird am LCD (Zeile 1) der Text „FB-Adresse: xx“ sowie der zugehörige Zahlenwert in Hex ausgegeben und aktualisiert. In Zeile 2 wird der Text „FB-Befehl: xx“ sowie der zugehörige Zahlenwert in Hex ausgegeben und aktualisiert.

**Struktogramm Main:****Struktogramm ISR:**

**Ergänzen Sie den fehlenden Code:**

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

#define IRinput P3_3;                                     // Infrarot-Empfänger
sbit at P1_1 oszi;                                      // Hilfsbit für Oszi zur Zeitmessung
unsigned char buf[16];                                    // Text-Buffer für LCD
unsigned char RC5Code, z, T, t1,j;

void zeit (unsigned long sec)                           //IRService-Routine RC 5 Code: Original-----
{
    for (z= sec; z != 0; z--);      // Zeitverzögerungsschleife
}

void IReinlesen (void) interrupt 2          // Interrupt EX1
{
}

void main (void)
{
    initlcd();                                         // LCD initialisieren
    sprintf(buf," Digitaluhr");
    textlcd (buf,1);
    sprintf(buf,"Uhr stellen");
    textlcd (buf,2);

    while(1)                                         // Endlosschleife
    {

    } // von while
}// von main
```



Programm P3: Uhr stellen mittels FB und S2...

Der Anwender der Uhr möchte zu Programmbeginn seine Uhr stellen...

Als Eingabemöglichkeit können Sie „nur“ die IR-FB, Tasten 0-9 sowie den „Platinentaster“ S2 nutzen.
Entwickeln Sie einen Menülaufplan (ähnlich einem Programmalaufplan) wo erkenntlich wird, wie sie diese Aufgabe lösen.

TIPP: Benutzen Sie die beiden IR-EX zur Eingabe.

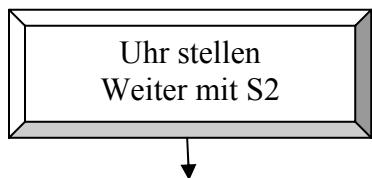
Als Hilfe ist das Struktogramm von MAIN abgebildet

Uhr stellen mit IR-FB + S2-IR Main
EX1 = 1
EX = 1
EA = 1
IT0 = 1
IT1 = 0 !! Obwohl Datenblatt = 1 zeigt!
IP= 3 EX0; IP=0 EX1
initLCD
LCD1: Uhr stellen
LCD2: weiter mit S2
Uhr stellen () !!!!
Endlos
Uhr ()
Zeit ~0,5 sec
LED an
LCD-Ausgabe() (Uhrzeit std:min:sec)
Zeit ~0,5sec
LED aus

Funktion Uhr_stellen():

Bsp. LCD-Menüführung:

Struktogramm Uhr stellen:





Ergänzen Sie den fehlenden Code:

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

#define IRinput P3_3;                                  // Infrarot-Empfänger
#define S2 P3_2;                                         // Taster S2
sbit at P1_1 oszi;                                     // Hilfsbit für Oszi zur Zeitmessung
sfr at P2 LED;

unsigned char buf [16], sec, std, min;                  // Text-Buffer für LCD
unsigned char RC5Code, T, t1,j, Sctr;
unsigned char secE, secZ, minE, minZ, stdE, stdZ ;
unsigned int z;

void ISR_S2 (void) interrupt 0           // Interrupt EX0
{
    ???
}

void zeit (unsigned int y)
{
}

void ISR_FB (void) interrupt 2          // Interrupt EX1
{
}

void Uhr_stellen (void)
{
    ???
}

void Uhr (void)
{
}

void Ausgabe_LCD (void)
{
}

void main (void)
{
    IT1 = 0;                // ext. Interrupt1 TROTZ EXI 1
    EX1 = 1;                // ext. Interrupt1 freigeben
    IT0 = 1;                // ext. Interrupt1 TROTZ EXI 0
    EX0 = 1;                // ext. Interrupt1 freigeben
    EA = 1;                 // globale Interruptfreigabe
    IPH0 = 0x01;             //Priorität = 3
    IPL0 = 0x00;
    initlcd();               // LCD initialisieren
    Uhr_stellen();

    while(1)                // Endlosschleife
    {
        Uhr();
        zeit (49000);         //Schleife 1 sec?? Hardware
        LED = 0x5F;
        Ausgabe_LCD();
        zeit (49000);         //Schleife 1 sec?? Hardware
        LED = 0xAF;
    } // von while
} // von main
```



Programm P3: Uhr stellen mittels FB und S2...

Der Anwender der Uhr möchte zu Programmbeginn seine Uhr stellen...

Als Eingabemöglichkeit können Sie „nur“ die IR-FB, Tasten 0-9 sowie den „Platinentaster“ S2 nutzen. Entwickeln Sie einen Menüablaufplan (ähnlich einem Programmablaufplan) wo erkenntlich wird, wie sie diese Aufgabe lösen.

TIPP: Benutzen Sie die beiden IR-EX zur Eingabe.

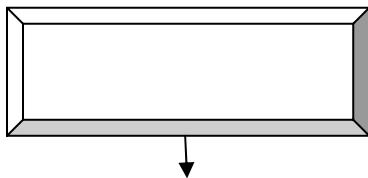
Als Hilfe ist das Struktogramm von MAIN abgebildet

Uhr stellen mit IR-FB + S2-IR Main
EX1 = 1
EX = 1
EA = 1
IT0 = 1
IT1 = 0 !! Obwohl Datenblatt = 1 zeigt!
IP= 3 EX0; IP=0 EX1
initLCD
LCD1: Uhr stellen
LCD2: weiter mit S2
Uhr stellen () !!!!
Endlos
Uhr ()
Zeit ~0,5 sec
LED an
LCD-Ausgabe() (Uhrzeit std:min:sec)
Zeit ~0,5sec
LED aus

Funktion Uhr_stellen():

Bsp. LCD-Menüführung:

Struktogramm Uhr stellen:



**Ergänzen Sie den fehlenden Code:**

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

#define IRinput P3_3;                                  // Infrarot-Empfänger
#define S2 P3_2;                                         // Taster S2
sbit at P1_1 oszi;                                     // Hilfsbit für Oszi zur Zeitmessung
sfr at P2 LED;

unsigned char buf [16], sec, std, min;                  // Text-Buffer für LCD
unsigned char RC5Code, T, t1,j, Sctr;
unsigned char secE, secZ, minE, minZ, stdE, stdZ ;
unsigned int z;

void ISR_S2 (void) interrupt 0           // Interrupt EX0
{
    {
    ????
}
void zeit (unsigned int y)
{
}

void ISR_FB (void) interrupt 2          // Interrupt EX1
{
}

void Uhr_stellen (void)
{
    {
    ???
}
void Uhr (void)
{
}

void Ausgabe_LCD (void)
{
}

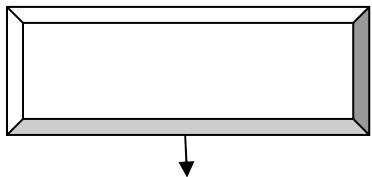
void main (void)
{
    IT1 = 0;                // ext. Interrupt1 TROTZ EXI 1
    EX1 = 1;                // ext. Interrupt1 freigeben
    IT0 = 1;                // ext. Interrupt1 TROTZ EXI 0
    EX0 = 1;                // ext. Interrupt1 freigeben
    EA = 1;                 // globale Interruptfreigabe
    IPH0 = 0x01;             //Priorität = 3
    IPL0 = 0x00;
    initlcd();              // LCD initialisieren
    Uhr_stellen();

    while(1)                // Endlosschleife
    {
        Uhr();
        zeit (49000);        //Schleife 1 sec?? Hardware
        LED = 0x5F;
        Ausgabe_LCD();
        zeit (49000);        //Schleife 1 sec?? Hardware
        LED = 0xAF;
    } // von while
} // von main
```

**Programm P3: Uhr stellen mittels FB und S2...**

Der Anwender der Uhr möchte zu Programmbeginn seine Uhr stellen...

Als Eingabemöglichkeit können Sie „nur“ die IR-FB, Tasten 0-9 sowie den „Platinentaster“ S2 nutzen.
Entwickeln Sie einen Menüablaufplan (ähnlich einem Programmablaufplan) wo erkenntlich wird, wie sie diese Aufgabe lösen.
TIPP: Benutzen Sie die beiden IR-EX zur Eingabe.

Funktion Uhr_stellen():**Bsp. LCD-Menüführung:****Struktogramm Uhr stellen:**

**Ergänzen Sie den fehlenden Code:**

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...

#define IRinput P3_3;                                  // Infrarot-Empfänger
#define S2 P3_2;                                         // Taster S2
sbit at P1_1 oszi;                                     // Hilfsbit für Oszi zur Zeitmessung
sfr at P2 LED;

unsigned char buf [16], sec, std, min;                  // Text-Buffer für LCD
unsigned char RC5Code, T, t1,j, Sctr;
unsigned char secE, secZ, minE, minZ, stdE, stdZ ;
unsigned int z;

void ISR_S2 (void) interrupt 0           // Interrupt EX0
{
    {
    ????
}
void zeit (unsigned int y)
{
}

void ISR_FB (void) interrupt 2          // Interrupt EX1
{
}

void Uhr_stellen (void)
{
    {
    ???
}
void Uhr (void)
{
}

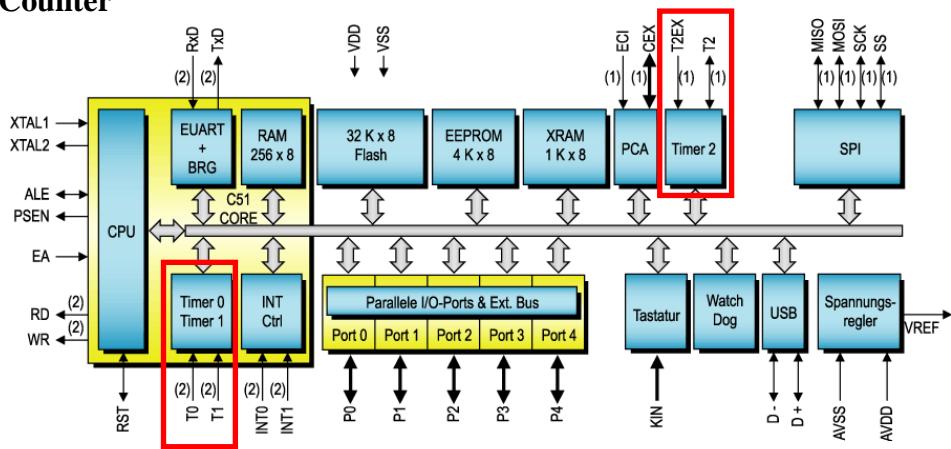
void Ausgabe_LCD (void)
{
}

void main (void)
{
    IT1 = 0;                // ext. Interrupt1 TROTZ EXI 1
    EX1 = 1;                // ext. Interrupt1 freigeben
    IT0 = 1;                // ext. Interrupt1 TROTZ EXI 0
    EX0 = 1;                // ext. Interrupt1 freigeben
    EA = 1;                 // globale Interruptfreigabe
    IPH0 = 0x01;             //Priorität = 3
    IPL0 = 0x00;
    initlcd();              // LCD initialisieren
    Uhr_stellen();

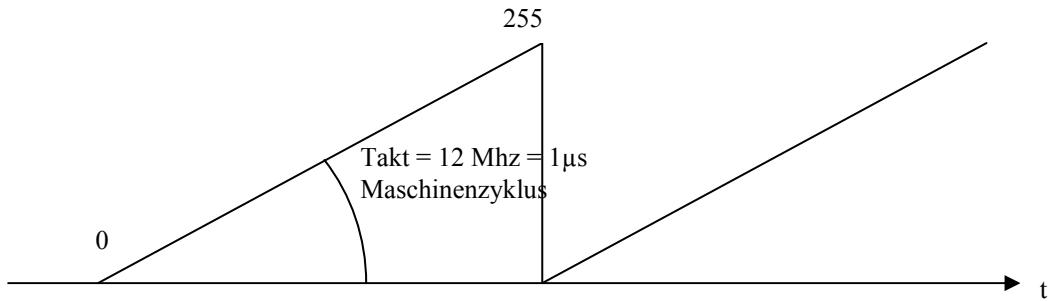
    while(1)                // Endlosschleife
    {
        Uhr();
        zeit (49000);         //Schleife 1 sec?? Hardware
        LED = 0x5F;
        Ausgabe_LCD();
        zeit (49000);         //Schleife 1 sec?? Hardware
        LED = 0xAF;
    } // von while
} // von main
```



11. Timer / Counter



11.1. Prinzip:



Aufgaben:

- Quarz wird auf 24 Mhz erhöht, Was ändert sich?: _____
- Zähler soll nur bis 130 zählen: Wie wirkt sich dies aus_____
- Es soll ein 16 Bit Timer verwendet werden. Was ändert sich? _____

11.2. Auszüge Datenblatt Timer/Counter 8051 T0, T1:

TMOD : Timermodus-Kontrollregister für Timer1 und Timer0							
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Kontrolle Timer 1				Kontrolle Timer 0			
TCON : Timer-Kontrollregister für Timer1 u. 0, ext Interrupt 1 u. 0							
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Kontrolle Timer 1 und 0				ext. Interrupt-Kontrolle			

**Register TMOD:**

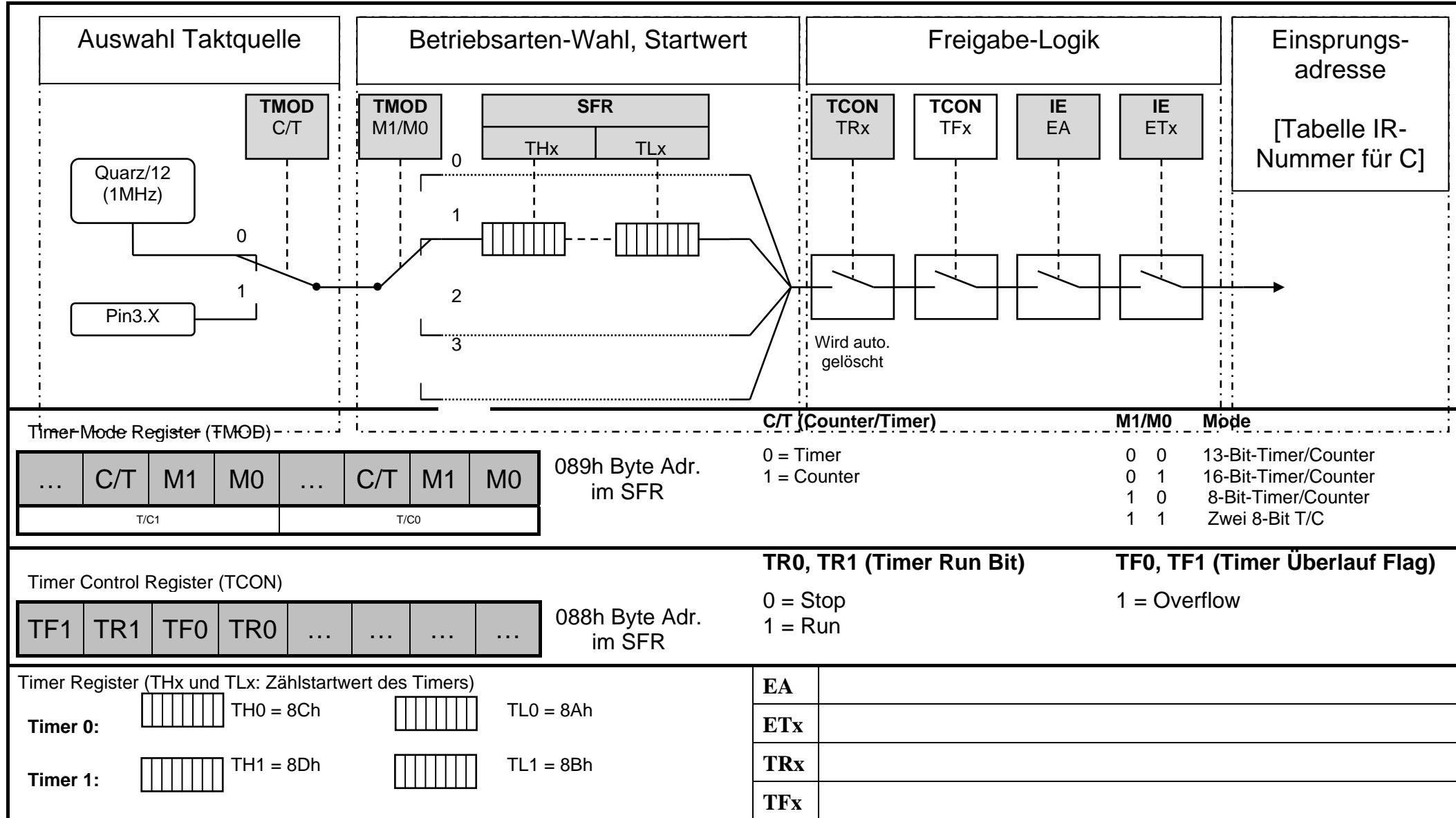
7	6	5	4	3	2	1	0
GATE1	C/T1#	M11	M01	GATE0	C/T0#	M10	M00
Bit Number	Bit Mnemonic	Description					
7	GATE1	Timer 1 Gating Control Bit Clear to enable Timer 1 whenever TR1 bit is set. Set to enable Timer 1 only while INT1# pin is high and TR1 bit is set.					
6	C/T1#	Timer 1 Counter/Timer Select Bit Clear for Timer operation: Timer 1 counts the divided-down system clock. Set for Counter operation: Timer 1 counts negative transitions on external pin T1.					
5	M11	Timer 1 Mode Select Bits <u>M11 M01</u> <u>Operating mode</u> 0 0 Mode 0: 8-bit Timer/Counter (TH1) with 5-bit prescaler (TL1). 0 1 Mode 1: 16-bit Timer/Counter. 1 0 Mode 2: 8-bit auto-reload Timer/Counter (TL1) ⁽¹⁾ 1 1 Mode 3: Timer 1 halted. Retains count					
4	M01						
3	GATE0	Timer 0 Gating Control Bit Clear to enable Timer 0 whenever TR0 bit is set. Set to enable Timer/Counter 0 only while INT0# pin is high and TR0 bit is set.					
2	C/T0#	Timer 0 Counter/Timer Select Bit Clear for Timer operation: Timer 0 counts the divided-down system clock. Set for Counter operation: Timer 0 counts negative transitions on external pin T0.					
1	M10	Timer 0 Mode Select Bit <u>M10 M00</u> <u>Operating mode</u> 0 0 Mode 0: 8-bit Timer/Counter (TH0) with 5-bit prescaler (TL0). 0 1 Mode 1: 16-bit Timer/Counter. 1 0 Mode 2: 8-bit auto-reload Timer/Counter (TL0) ⁽²⁾ 1 1 Mode 3: TL0 is an 8-bit Timer/Counter TH0 is an 8-bit Timer using Timer 1's TR0 and TF0 bits.					
0	M00						

Register TCON

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Bit Number	Bit Mnemonic	Description					
7	TF1	Timer 1 Overflow Flag Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 1 register overflows.					
6	TR1	Timer 1 Run Control Bit Clear to turn off Timer/Counter 1. Set to turn on Timer/Counter 1.					
5	TF0	Timer 0 Overflow Flag Cleared by hardware when processor vectors to interrupt routine. Set by hardware on Timer/Counter overflow, when Timer 0 register overflows.					
4	TR0	Timer 0 Run Control Bit Clear to turn off Timer/Counter 0. Set to turn on Timer/Counter 0.					
3	IE1	Interrupt 1 Edge Flag Cleared by hardware when interrupt is processed if edge-triggered (see IT1). Set by hardware when external interrupt is detected on INT1# pin.					
2	IT1	Interrupt 1 Type Control Bit Clear to select low level active (level triggered) for external interrupt 1 (INT1#). Set to select falling edge active (edge triggered) for external interrupt 1.					
1	IE0	Interrupt 0 Edge Flag Cleared by hardware when interrupt is processed if edge-triggered (see IT0). Set by hardware when external interrupt is detected on INT0# pin.					
0	IT0	Interrupt 0 Type Control Bit Clear to select low level active (level triggered) for external interrupt 0 (INT0#). Set to select falling edge active (edge triggered) for external interrupt 0.					

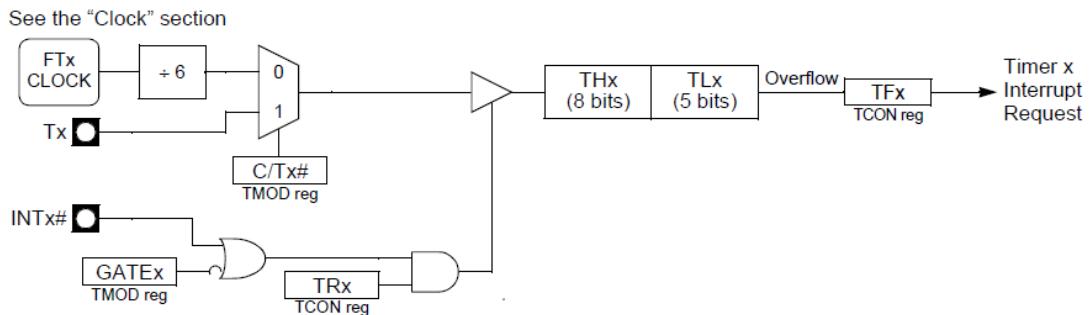


Zeichnen Sie alle „Schalterstellungen“ (= Bitwerte) ein für folgende Konfiguration:
T0: 8 Bit Timer, autoReloaded from TH0, Startwert 23_d ohne Freigabelogik mit Timer-IR-Freigabe und starten des Timers0



**Analyse Timer 0, Gruppe Modus0:****Zeichnen Sie die Konfiguration ein:****Mode 0 (13-bit Timer)**

Mode 0 configures Timer 0 as an 13-bit Timer which is set up as an 8-bit Timer (TH0 register) with a modulo 32 prescaler implemented with the lower five bits of TL0 register (see Figure 35). The upper three bits of TL0 register are indeterminate and should be ignored. Prescaler overflow increments TH0 register.

Figure 35. Timer/Counter x (x = 0 or 1) in Mode 0**TMOD : Timermodus-Kontrollregister für Timer1 und Timer0**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Kontrolle Timer 1				Kontrolle Timer 0			

TCON : Timer-Kontrollregister für Timer1 u. 0, ext Interrupt 1 u. 0

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Kontrolle Timer 1 und 0				ext. Interrupt-Kontrolle			

Programm:

```

void main (void)
{
...
TMOD = 0b00000000;           // Modus0 Timer
TCON = 0b00000000;           //ohne Interrupt
LED = 0x0F;
    while (1) // Endlosschleife Hauptprogramm -----
    {
        if (TF0 == 1)
        {
            LED = 0xFF;
            TF0 = 0;
        }
        else
            LED = 0x0F;
    }
    TR0 = 1;
    TR0 = 0;
    sprintf(buf,"T0_Mod0: %3d", TH0);
    textlcd(buf,2);
}
}

```

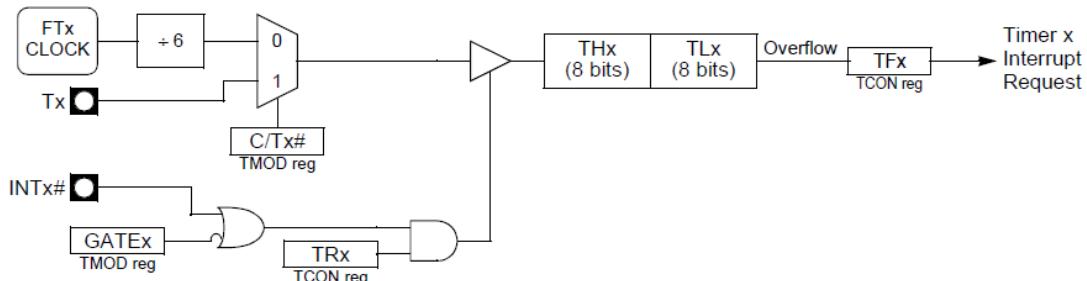
Beobachtung:

**Analyse Timer 0, Gruppe Modus1:****Zeichnen Sie die Konfiguration ein:****Mode 1 (16-bit Timer)**

Mode 1 configures Timer 0 as a 16-bit Timer with TH0 and TL0 registers connected in cascade (see Figure 36). The selected input increments TL0 register.

Figure 36. Timer/Counter x ($x = 0$ or 1) in Mode 1

See the "Clock" section

**TMOD : Timermodus-Kontrollregister für Timer1 und Timer0**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Kontrolle Timer 1				Kontrolle Timer 0			

TCON : Timer-Kontrollregister für Timer1 u. 0, ext Interrupt 1 u. 0

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Kontrolle Timer 1 und 0				ext. Interrupt-Kontrolle			

Programm:

```

void main (void)
{
...
    TMOD = 0b00000001;          // Modus1 Timer
    TCON = 0b00000000;          //ohne Interrupt
    LED = 0xF;
    while (1) // Endlosschleife Hauptprogramm -----
{
    if(TF0 == 1)
    {
        LED = 0xFF;
        TF0 = 0;
    }
    else
        LED = 0x0F;
    TR0 = 1;
    TR0 = 0;
    sprintf(buf,"T0_Mod1: %3d %3d", TH0, TL0);
    textlcd(buf,2);
}
}

```

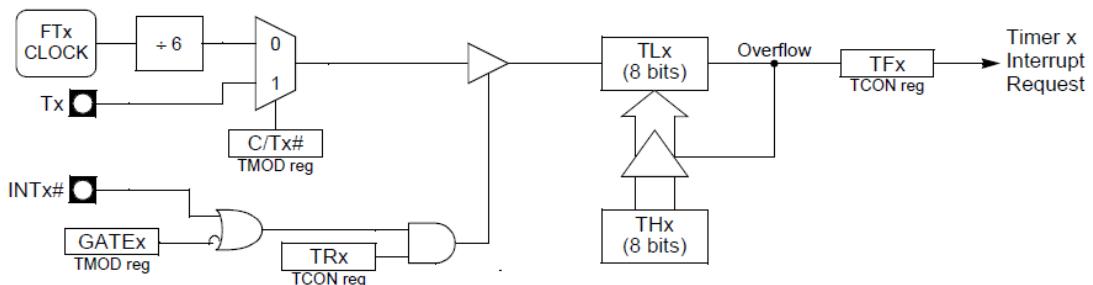
Beobachtung:

**Analyse Timer 0, Gruppe Modus2:****Zeichnen Sie die Konfiguration ein:**

Mode 2 (8-bit Timer with Auto-Reload) Mode 2 configures Timer 0 as an 8-bit Timer (TL0 register) that automatically reloads from TH0 register (see Figure 37). TL0 overflow sets TF0 flag in TCON register and reloads TL0 with the contents of TH0, which is preset by software. When the interrupt request is serviced, hardware clears TF0. The reload leaves TH0 unchanged. The next reload value may be changed at any time by writing it to TH0 register.

Figure 37. Timer/Counter x ($x = 0$ or 1) in Mode 2

See the "Clock" section

**TMOD : Timermodus-Kontrollregister für Timer1 und Timer0**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Kontrolle Timer 1				Kontrolle Timer 0			

TCON : Timer-Kontrollregister für Timer1 u. 0, ext Interrupt 1 u. 0

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Kontrolle Timer 1 und 0				ext. Interrupt-Kontrolle			

Programm:

```

void main (void)
{
    TMOD = 0b00000010;          // Modus2 Timer
    TCON = 0b00000000;          // ohne Interrupt
    TH0 = 50;
    LED = 0x0F;

    while (1) // Endlosschleife Hauptprogramm -----
    {
        if (TF0 == 1)
        {
            LED = 0xFF;
            TF0 = 0;
        }
        else
            LED = 0x0F;

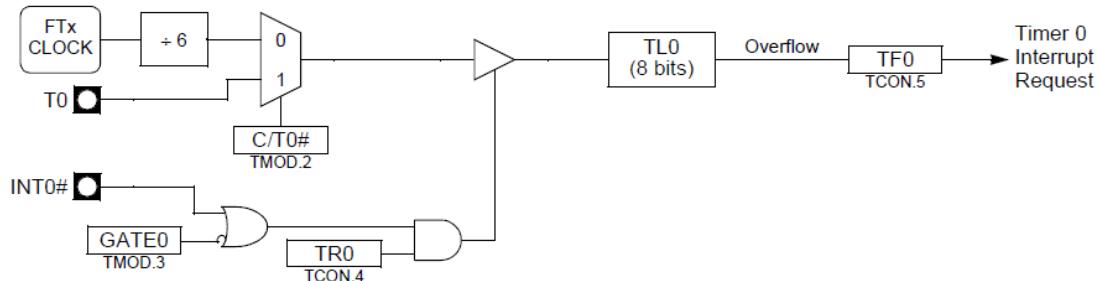
        TR0 = 1;
        TR0 = 0;
        sprintf(buf,"T0_Mod2: %3d %3d", TH0, TL0);
        textlcd(buf,2);
    }
}

```

Beobachtung:

**Analyse Timer 0, Gruppe Modus3:****Zeichnen Sie die Konfiguration ein:****Mode 3 (Two 8-bit Timers)**

Mode 3 configures Timer 0 such that registers TL0 and TH0 operate as separate 8-bit Timers (see Figure 38). This mode is provided for applications requiring an additional 8-bit Timer or Counter. TL0 uses the Timer 0 control bits C/T0# and GATE0 in TMOD register, and TR0 and TF0 in TCON register in the normal manner. TH0 is locked into a Timer function (counting $F_{PER}/6$) and takes over use of the Timer 1 interrupt (TF1) and run control (TR1) bits. Thus, operation of Timer 1 is restricted when Timer 0 is in mode 3.

Figure 38. Timer/Counter 0 in Mode 3: Two 8-bit Counters**TMOD : Timermodus-Kontrollregister für Timer1 und Timer0**

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Gate	C/T	M1	M0	Gate	C/T	M1	M0
Kontrolle Timer 1				Kontrolle Timer 0			

TCON : Timer-Kontrollregister für Timer1 u. 0, ext Interrupt 1 u. 0

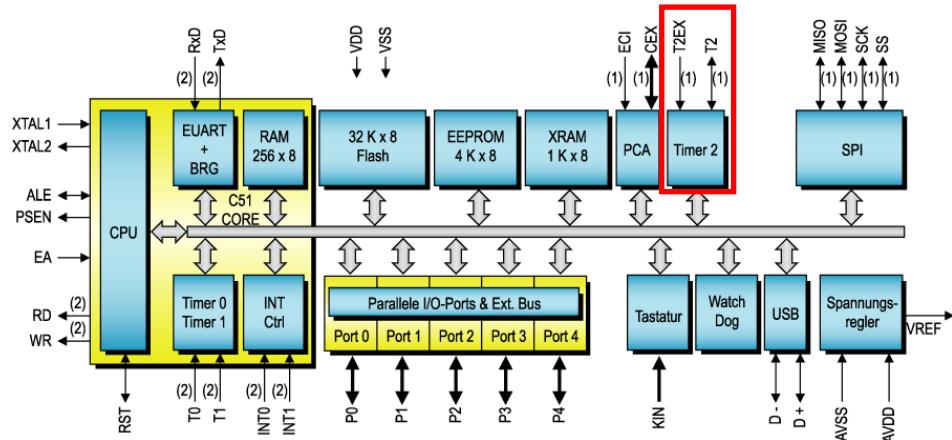
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Kontrolle Timer 1 und 0				ext. Interrupt-Kontrolle			

Programm:

```
void main (void)
{
    TMOD = 0b00000011;          // Modus3 Timer
    TCON = 0b00000000;          //ohne Interrupt
    LED = 0x0F;

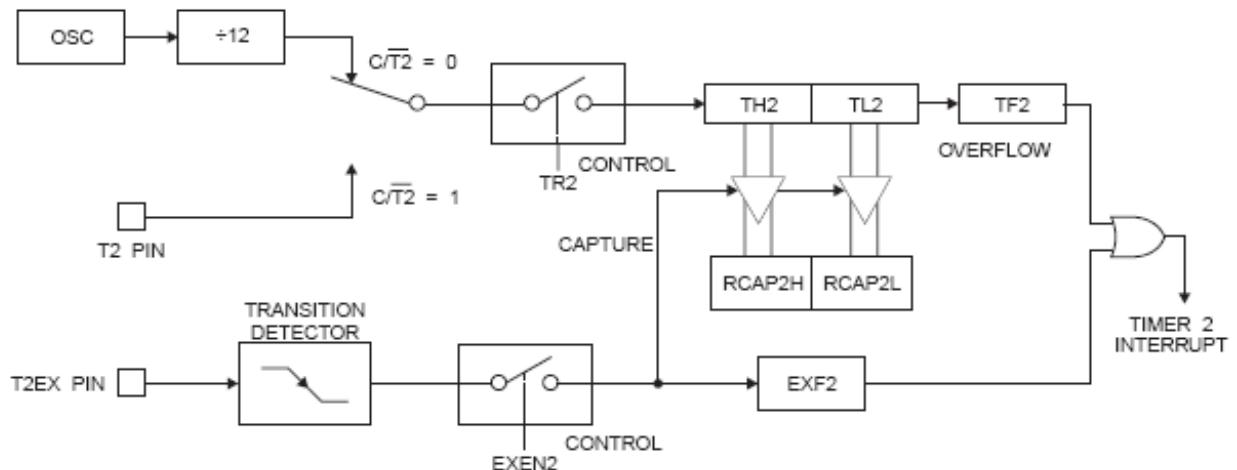
    while (1) // Endlosschleife Hauptprogramm -----
    {
        if (TF0 == 1)
        {
            LED = 0xFF;
            TF0 = 0;
        }
        else
            LED = 0x0F;
        TR0 = 1;
        TR0 = 0;
        sprintf(buf,"T0_Mod3: %3d %3d", TH0, TL0);
        textlcd (buf,2);
    }
}
```

Beobachtung:

**P4: Uhr mit Timer2 als Zeitgeber...**

Um eine exaktere Quarzzeit zu erhalten wollen wir mit dem Timer2 einen 50ms Zeitstempel realisieren:

Figure 1. Timer 2 in Capture Mode



Berechnung Startwerte:



T2CON - Timer 2 Control Register (C8h)

7	6	5	4	3	2	1	0	
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2#	CP/RL2#	
Bit Number	Bit Mnemonic	Description						
7	TF2	Timer 2 overflow Flag Must be cleared by software. Set by hardware on Timer 2 overflow, if RCLK = 0 and TCLK = 0.						
6	EXF2	Timer 2 External Flag Set when a capture or a reload is caused by a negative transition on T2EX pin if EXEN2 = 1. When set, causes the CPU to vector to Timer 2 interrupt routine when Timer 2 interrupt is enabled. Must be cleared by software. EXF2 doesn't cause an interrupt in Up/down counter mode (DCEN = 1).						
5	RCLK	Receive Clock bit Cleared to use Timer 1 overflow as receive clock for serial port in mode 1 or 3. Set to use Timer 2 overflow as receive clock for serial port in mode 1 or 3.						
4	TCLK	Transmit Clock bit Cleared to use Timer 1 overflow as transmit clock for serial port in mode 1 or 3. Set to use Timer 2 overflow as transmit clock for serial port in mode 1 or 3.						
3	EXEN2	Timer 2 External Enable bit Cleared to ignore events on T2EX pin for Timer 2 operation. Set to cause a capture or reload when a negative transition on T2EX pin is detected, if Timer 2 is not used to clock the serial port.						
2	TR2	Timer 2 Run control bit Cleared to turn off Timer 2. Set to turn on Timer 2.						
1	C/T2#	Timer/Counter 2 select bit Cleared for timer operation (input from internal clock system: $F_{CLK\ PERIPH}$). Set for counter operation (input from T2 input pin, falling edge trigger). Must be 0 for clock out mode.						
0	CP/RL2#	Timer 2 Capture/Reload bit If RCLK = 1 or TCLK = 1, CP/RL2# is ignored and timer is forced to Auto-reload on Timer 2 overflow. Cleared to Auto-reload on Timer 2 overflows or negative transitions on T2EX pin if EXEN2 = 1. Set to capture on negative transitions on T2EX pin if EXEN2 = 1.						

Reset Value = 0000 0000b

Bit addressable

**P4: Uhr mit Timer2 als Zeitgeber...**

Gegeben ist folgender Programmauszug, welcher einen Uhrentakt generiert mittels dem Timer2. Es soll alle 50ms ein Zeitstempel realisiert werden, welcher dann 20 mal gezählt eine Sekunde ergibt. Ergänzen Sie die fehlenden Werte im Hauptprogramm und kommentieren die Programmzeilen in der Endlosschleife. Schreiben Sie das Struktogramm für die MAIN.

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...
sfr at P2 LED;                                         //Sekundenblinker
unsigned char buf[16];                                  // Text-Buffer für LCD
unsigned long z;
unsigned char sec, min, std, TF_ctr ;
// ----- Funktionen -----
void Uhr (void)                                         //Bekannt aus vorigen Übungen
{
}

void Ausgabe_LCD (void)                                 //Bekannt aus vorigen Übungen
{
}

//----- Hauptprogramm
void main (void)
{
    //----- Initialisierungen
    initlcd();                                         // LCD initialisieren
    sprintf(buf,"Uhr mit T2");
    textlcd (buf,1);                                   //Ausgabe erste Zeile

    TH2 =      ;                                     // Startwert _____d
    TL2 =      ;                                     // für ca. 50ms
    TR2 =      ;                                     //Timer starten

    while (1) // Endlosschleife Hauptprogramm -----
    {
        Ausgabe_LCD();                               _____
        LED = 0x0f;                                 _____
        if (TF2 ==1)                                _____
            {
                TF2 = 0;                            _____
                TF_ctr++;                         _____
                if (TF_ctr == 20)                  _____
                    {
                        Uhr();                     _____
                        LED = 0xFF;                 _____
                        TF_ctr = 0;                _____
                    }
            }
    }
}
```

**P4: Uhr mit Timer2 als Zeitgeber...**

Gegeben ist folgender Programmauszug, welcher einen Uhrentakt generiert mittels dem Timer2. Es soll alle 50ms ein Zeitstempel realisiert werden, welcher dann 20 mal gezählt eine Sekunde ergibt. Ergänzen Sie die fehlenden Werte im Hauptprogramm und kommentieren die Programmzeilen in der Endlosschleife. Schreiben Sie das Struktogramm für die MAIN

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...
sfr at P2 LED;                                         //Sekundenblinker
unsigned char buf[16];                                  // Text-Buffer fuer LCD
unsigned long z;
unsigned char sec, min, std, TF_ctr ;
// ----- Funktionen -----
void Uhr (void)                                         //Bekannt aus vorigen Übungen
{
}

void Ausgabe_LCD (void)                                 //Bekannt aus vorigen Übungen
{
}

//----- Hauptprogramm
void main (void)
{
    //----- Initialisierungen
    initlcd();                                           // LCD initialisieren
    sprintf(buf,"Uhr mit T2");
    textlcd (buf,1);                                     //Ausgabe erste Zeile
}
```

while (1) // Endlosschleife Hauptprogramm -----

{
 Ausgabe_LCD();

LED = 0x0f;

if (TF2 ==1)

{
 TF2 = 0;

TF_ctr++;

if (TF_ctr == 20)

{
 Uhr();

LED = 0xFF;

TF_ctr = 0;

}

}

**P4: Uhr mit Timer2 als Zeitgeber...**

Gegeben ist folgender Programmauszug, welcher einen Uhrentakt generiert mittels dem Timer2. Es soll alle 50ms ein Zeitstempel realisiert werden, welcher dann 20 mal gezählt eine Sekunde ergibt. Ergänzen Sie die fehlenden Werte im Hauptprogramm und kommentieren die Programmzeilen in der Endlosschleife. Schreiben Sie das Struktogramm für die MAIN

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...
sfr at P2 LED;                                         //Sekundenblinker
unsigned char buf[16];                                  // Text-Buffer für LCD
unsigned long z;
unsigned char sec, min, std, TF_ctr ;
// ----- Funktionen -----
void Uhr (void)                                         //Bekannt aus vorigen Übungen
{
}

void Ausgabe_LCD (void)                                 //Bekannt aus vorigen Übungen
{
}

//----- Hauptprogramm
void main (void)
{
    //----- Initialisierungen
    initlcd();                                         // LCD initialisieren
    sprintf(buf,"Uhr mit T2");
    textlcd (buf,1);                                   //Ausgabe erste Zeile

    while (1) // Endlosschleife Hauptprogramm -----
    {
        Ausgabe_LCD();
        LED = 0x0f;
    }
}
```



Übungen Timer / Counter: Timer2 als Stoppuhr IR-Taste Start/Stopp

Es soll mittels dem µC-Board und RIDE eine Stoppuhr realisiert werden, welche mit einem Zeitstempel des Timers 2 mit Interrupt von 50ms genau zählen soll. Die Taste S₂ (= IREX0) soll die Stoppuhr starten und stoppen; ausgelöst durch einen IR. Deshalb ist die IR-Priorität des Tasters auf 3 zu stellen. Die Ausgabe im LCD erfolgt in Zeile 1 zu Programmstart = *Stoppuhr*. Wenn die „Uhr“ läuft, soll in Zeile 1 *Stoppuhr run* stehen und auf Zeile 2 die Uhrzeit in 50ms Schritten. Bei Stop durch S2 soll die aktuell gestoppte Zeit sowie der Text *STOPP* in Zeile 1 dargestellt werden. Erstellen Sie die Struktogramme sowie ergänzen Sie den fehlenden Programmcode.

Struktogramm Main:

Struktogramm ISR_Stopp_Start_EX0:

ISR_Timer2

**Programmauszug:**

```
#include "at89c5131.h"          // fuer Atmel AT89C5131
#include "lcd.c"                 // LCD-Anzeige
#include "stdio.h"               // sprintf...
//Konstanten- Variablendeklaration
sfr at P2 LED;                //Sekundenblinker
unsigned char buf[16];          // Text-Buffer fuer LCD
unsigned char msec50, IREX0ctr;

void ISR_Stopp_Start_EX0 (void) interrupt 0
{

}

void ISR_Timer2 (void) interrupt 5
{

}

void Ausgabe_LCD (void)
{

}

void main (void)
{
    initlcd();                  // LCD initialisieren
    sprintf(buf,"Stoppuhr ");
    textlcd (buf,1);           //Ausgabe erste Zeile

    while (1) // Endlosschleife Hauptprogramm-----
    {

    }
}
```



Übungen Timer / Counter: Countdownzähler für Raketenstart

Es soll mittels dem µC-Board und RIDE ein Zähler für den Countdown zu einem Raketenstart realisiert werden. Hierzu wird mit einem Zeitstempel des Timers 2 mit Interrupt von 50ms genau zählen soll. Die Taste S₂ (= IREX0) soll der Countdown starten, stoppen bzw. unterbrochen werden; ausgelöst durch einen IR. Deshalb ist die IR-Priorität des Tasters auf 2 zu stellen. Die Ausgabe im LCD erfolgt in Zeile 1 zu Programmstart = Counter. Wenn die „Uhr“ läuft, soll in Zeile 1 Counter *run* stehen und auf Zeile 2 der Zählstand = *Noch ddd sec.* Bei Stop durch S₂ soll der aktuell gestoppte Zählerstand sowie der Text *Counter Stopp* in Zeile 1 dargestellt werden. Erreicht der Zählerstand 0, ist auf beiden Zeilen der Text *FIRE* auszugeben. Erstellen Sie die Struktogramme und ergänzen Sie den fehlenden Programmcode.

Struktogramm Main:

Struktogramm ISR_Stopp_Start:

ISR_Timer2

**Programmauszug:**

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...
sfr at P2 LED;                                         // Sekundenblinker
unsigned char buf[16];                                  // Text-Buffer fuer LCD
unsigned long z;
unsigned char IREX_ctr, ctr,TF_ctr ;

void ISR_Start_Stopp (void) interrupt 0
{

}

void ISR_Timer2 (void) interrupt 5
{

}

void Ausgabe_LCD (void)
{
    sprintf(buf, "Noch: %3d sec", ctr); //d = Dezimahl. x = Hex...
    textlcd (buf,2);
}

void main (void)
{
    initlcd();                                         // LCD initialisieren
    sprintf(buf,"Counter   ");
    textlcd (buf,1);        //Ausgabe erste Zeile

    while (1) // Endlosschleife Hauptprogramm -----
    {

    }
}
```



Übungen Timer / Counter: elektronischer Würfel 1-6

Es soll mittels dem µC-Board und RIDE ein Zähler als elektronischer Würfel realisiert werden. Hierzu wird mit einem Zeitstempel des Timers 2 mit Interrupt von 50ms gezählt. Die Taste S₂ (= IREX0) soll den Würfel starten bzw. stoppen; ausgelöst durch einen IR. Deshalb ist die IR-Priorität des Tasters auf 3 zu stellen. Die Ausgabe im LCD erfolgt in Zeile 1 zu Programmstart = *Wuerfel*. Wenn der Würfel „rollt“, soll in Zeile 1 *Wuerfel run* stehen und auf Zeile 2 der Zählstand = *Zahl d.* Bei Stopp durch S₂ soll der aktuell gestoppte Zählerstand sowie der Text *Wuerfelzahl* in Zeile 1 dargestellt werden. Erstellen Sie die Struktogramme und ergänzen Sie den fehlenden Programmcode.

Struktogramm Main:

Struktogramm ISR_Stopp_Start:

ISR_Timer2

**Programmauszug:**

```
#include "at89c5131.h"          // fuer Atmel AT89C5131
#include "lcd.c"                 // LCD-Anzeige
#include "stdio.h"                // sprintf...
unsigned char buf[16];           // Text-Buffer für LCD
unsigned long z;
unsigned char IREX_ctr, ctr,TF_ctr ;

void ISR_Start_Stopp (void) interrupt 0
{

}

void ISR_Timer2 (void) interrupt 5
{

}

void Ausgabe_LCD (void)
{
    sprintf(buf, "Zahl: %1d    ", ctr); //d = Dezimahl. x = Hex...
    textlcd(buf,2);
}

//----- Hauptprogramm
void main (void)
{

    while (1) // Endlosschleife Hauptprogramm -----
    {

    }
}
```

**P5: Programmierung einer einfachen Uhr:**

- Unterprogrammtechnik: Uhr_stellen(), Uhr(), Ausgabe_LCD()
- Stellen der Uhr nach Reset mittels IR-FB und IREX0 stdZ, stdE, minZ, minE, sekZ, sekE
- IR Priorität: EX0 =3, EX1 = 2 IRT2 = 1
- Sekudentakt mittels Timer2, 50ms Zeitstempel inkl. IR
- LED-Blinken im Sekudentakt

Struktogramm ISR_IR_FB:**Struktogramm Main:****Struktogramm Uhr_stellen:****Struktogramm Uhr:**



Programmstruktur P5 Digitaluhr

```
#include "at89c5131.h"                                // fuer Atmel AT89C5131
#include "lcd.c"                                         // LCD-Anzeige
#include "stdio.h"                                       // sprintf...
//Konstanten- Variablendeclaration
sfr at P2 LED;                                         //Sekundenblinker
#define IRinput P3_3;                                    // Infrarot-Empfänger
sbit at P1_1 oszi;                                      // Hilfsbit für Oszi zur Zeitmessung
unsigned char buf [16], sec, std, min;                  // Text-Buffer für LCD
unsigned char RC5Code, T, t1,j, Sctr, TF_ctr;
unsigned char secE, secZ, minE, minZ, stdE, stdZ ;
unsigned long z;

// ----- Funktionen -----
void ISR_S2 (void) interrupt 0      // Interrupt EX0
{
    ...
}

void zeit (unsigned int y)
{
    for (z= y; z != 0; z--);      // Zeitverzögerungsschleife 1000=??
}

//IRService-Routine RC 5 Code: Original-----
void ISR_IR_FB (void) interrupt 2      // Interrupt EX1
{
    ...
    ... einfügen
}

void ISR_Timer2 (void) interrupt 5
{
    ...
}

void Uhr_stellen (void)
{
    ...
}

void Uhr (void)
{
    ...
}

void Ausgabe_LCD (void)
{
    sprintf(buf, "Zeit: %2d:%2d:%2d", std, min, sec); //d = Dezimahl. x = Hex...
    textlcd (buf,2);
}

//----- Hauptprogramm
void main (void)
{
    while (1) // Endlosschleife Hauptprogramm -----
    {
        ...
    }
}
```



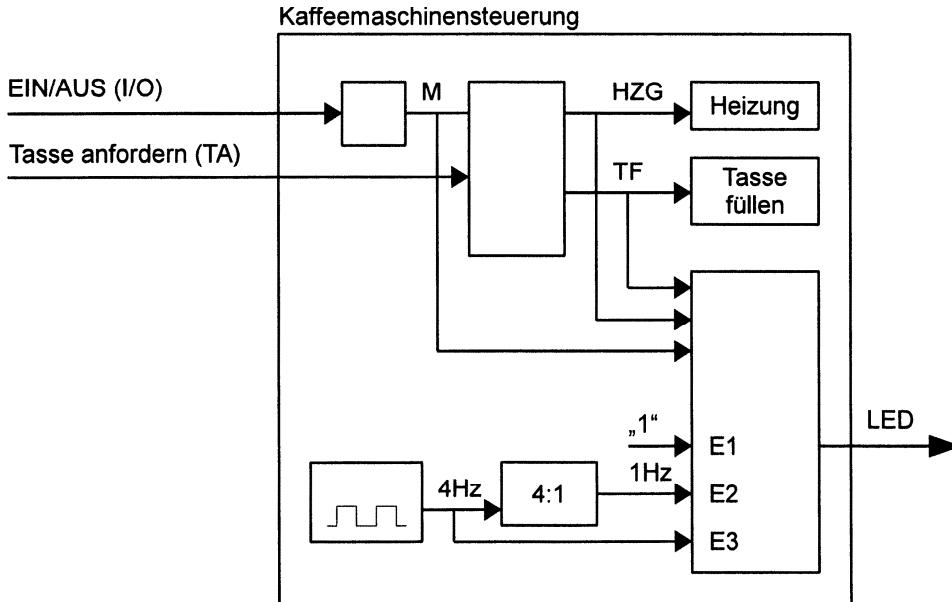
Übungen Mikrocontrollertechnik, ganzheitliche Aufgaben

Hauptprüfung 2011/2012	Berufliches Gymnasium (TG)	
1.5.2	Informationstechnik (Hardware)	
	Teil: 1 (Pflichtteil)	Aufgabe: 2 (3Seiten)

Kaffeeautomat

Im Folgenden soll das Prinzip eines einfachen Kaffeeautomaten betrachtet werden.

Blockschaltbild:



2.1 Das Bedienfeld besteht aus folgenden Komponenten:

Taster I/O : Schaltet die Maschine **EIN** (Internes Signal **M = 1**) und **AUS** (**M = 0**)

Taster TA : Tassenfüllung anfordern \Rightarrow wird mit Kaffee für 10 Sekunden befüllt (**TF = 1**)

Anzeige **LED**: Die Leuchtdiode leuchtet

- **dauernd**, wenn die Maschine **bereit** ist , ...
- ... **blinkt schnell** (4 Hz), wenn Wasser **geheizt** ...
- ... **langsam** (1 Hz) wenn eine Tasse **befüllt** wird,

2.1.1 Entwerfen (zeichnen) Sie eine Schaltung, mit deren Hilfe der Taster I/O die Maschine ein- und ausschalten kann (das Signal **M** soll bei jeder Tastenbetätigung abwechselnd auf High („1“) bzw. Low („0“) gehen). 2

2.1.2 Begründen Sie, warum der Taster I/O entprellt werden muss. 1

2.2 Blinkschaltung

Abhängig von den oben beschriebenen Bedingungen soll die LED mit 1 Hz bzw. 4 Hz blinken oder dauernd leuchten.

Es steht ein **Taktsignal** mit einer Frequenz von **4 Hz** zur Verfügung (s. Blockschaltbild).

2.2.1 Aus dem vorhandenen 4 Hz - Signal soll eine Frequenz von **1 Hz** erzeugt werden. 3
 Beschreiben Sie einen Lösungsvorschlag für ein entsprechendes synchrones Schaltwerk (4:1). Stellvertretend können Sie auch eine entsprechende Schaltung zeichnen.



2.2.2 Entwickeln Sie eine Schaltung, welche die drei Eingänge (**E1... E3**) in Abhängigkeit von den Steuerleitungen **M**, **HZG** und **TF** auf den Ausgang **LED** schaltet oder ihn auf „0“ legt.

2.3 Zur Zeitmessung für den Heizvorgang wird ein Zähler verwendet, der aus dem **1 Hz** Signal die **30 s** erzeugt.

2.3.1 Wie viele Flipflops werden für den Zähler minimal benötigt (Begründung angeben)?

2.3.2 Geben Sie die Kopfzeilen sowie die erste und die letzte Zeile der codierten Zustandsfolgetabelle des Zählers aus 2.3.1 an.

2.3.3 Beschreiben Sie mit wenigen Stichworten, welche zusätzliche Information aus der codierten Zustandsübergangstabelle (im Vergleich zum Zustandsdiagramm) gewonnen werden können.

2.4 Alternativ soll die Funktion des Kaffeautomaten mit einem **Mikrocontroller** realisiert werden.

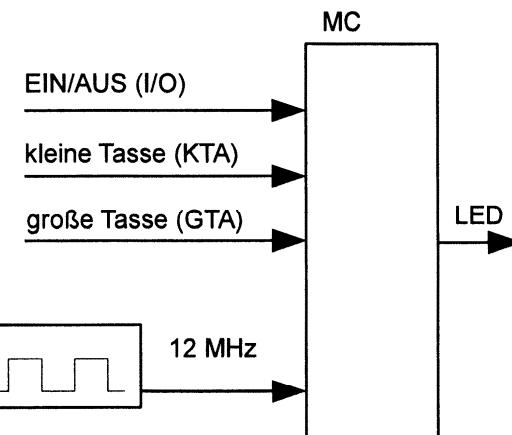
Es kann zwischen **zwei Tassengrößen** gewählt werden:

I/O: Taster

KTA: kleine Tasse und

GTA: große Tasse (Taster)

Die Taktfrequenz von 12MHz wurde für einen MC8051 gewählt, passen Sie diese Frequenz entsprechend an, wenn Sie einen anderen MC-Typ verwenden.



Geben Sie eine Liste der Pinbelegung für Ihren MC an!

- Mit dem Taster **<I/O>** schaltet man die Maschine EIN und AUS.
- Der Heizvorgang wird begonnen, wenn eingeschaltet wurde.
- Mit den Tastern **KTA** bzw. **GTA** wird ein Füllvorgang gestartet; das ist nur möglich, wenn der Heizvorgang abgeschlossen ist (**HZG = 0**).
- Die Kaffeemaschinensteuerung steuert die LED an und sorgt außerdem dafür, dass die Heizung (**HZG**) nach dem Einschalten und nach einem Füllvorgang für **30 Sekunden** auf High („1“) geht; solange kann kein weiterer Füllvorgang gestartet werden.



2.4.1 Zunächst soll die Automatensteuerung ohne Interrupt realisiert werden. Es stehen fertige Unterprogramme zur Verfügung: 4

heizung: HZG geht für 30 s auf „1“ / LED blinkt mit 4 Hz

kleine_tasse: TF geht für 10 s auf „1“ / LED blinkt mit 1 Hz

grosse_tasse: TF geht für 20 s auf „1“ / LED blinkt mit 1 Hz

Zeichnen Sie einen Programmablaufplan (für Assembler-Programm) oder ein Struktogramm (für C-Programm) des Hauptprogramms und kommentieren Sie Ihren Lösungsvorschlag sinnvoll.

2.5 Das Blinken der LED mit 4 Hz soll nun mittels **Timer** in einer **ISR** realisiert werden, so dass das UP <**heizung**> nur noch die Aufgabe hat, den Timer mit allen notwendigen Parametern für die Blinkfrequenz und die Heizdauer zu starten.

2.5.1 Bestimmen Sie eine geeignete Betriebsart für den Timer und schreiben Sie die Initialisierung als UP in Assembler oder als Funktion in „C“. Ergänzen Sie Ihr Listing durch aussagekräftige Kommentare. 3

2.5.2 Schreiben Sie das Unterprogramm bzw. die Funktion <**heizung**> 1

2.5.3 Zeichnen Sie einen PAP oder ein Struktogramm für die Timer_ISR. 3

2.5.4 In analoger Weise zum Unterprogramm <**heizung**> könnte man auch die beiden Unterprogramme <**kleine_tasse**> und <**grosse_tasse**> derart schreiben, dass auch hier Blinkfrequenz und Zeitdauer durch den gleichen Timer realisiert werden. Schreiben Sie eines der beiden UP und kommentieren Sie Ihre Befehlszeilen. 2

Welche zusätzlichen Ergänzungen müssten Sie in der ISR zum Timer durchführen? Eine verbale Beschreibung ist ausreichend.

2.6 Auf den **Taster I/O** soll nun mittels externem **Interrupt** reagiert werden.

2.6.1 Bestimmen Sie einen sinnvollen Portanschluss für den Taster I/O und schreiben Sie die Initialisierungszeilen als Unterprogramm (ISR) für den externen Interrupt. Ergänzen Sie Ihr Listing mit sinnvollen Kommentaren 2

2.6.2 Nun soll die ISR für den externen Interrupt **zusätzlich** dafür sorgen, dass dann, wenn im laufenden Betrieb **AUS**-geschaltet wird, alle Aktivitäten **gestoppt** und auf den **Ausgangszustand** zurück gegangen wird. Erstellen Sie einen PAP oder ein Struktogramm für diese ISR oder schreiben Sie ein kommentiertes Listing. 2



Aufgabe Abitur 2010/2011

Diebstahlsicherung für Audiogeräte im Auto

Audiogeräte im PKW müssen beim Einbau oder beim Batteriewechsel durch die Eingabe von vier Ziffern (0 bis 9) freigegeben werden. Die Freigabe erfolgt nur, wenn die eingegebenen Ziffern mit den vom Hersteller im Gerät gespeicherten Codeziffern (siehe Tabelle 1) übereinstimmen. Solange die Spannungsversorgung im Standby-Betrieb nicht unterbrochen wird, bleibt die einmalige Freigabe erhalten.

Informationslogik

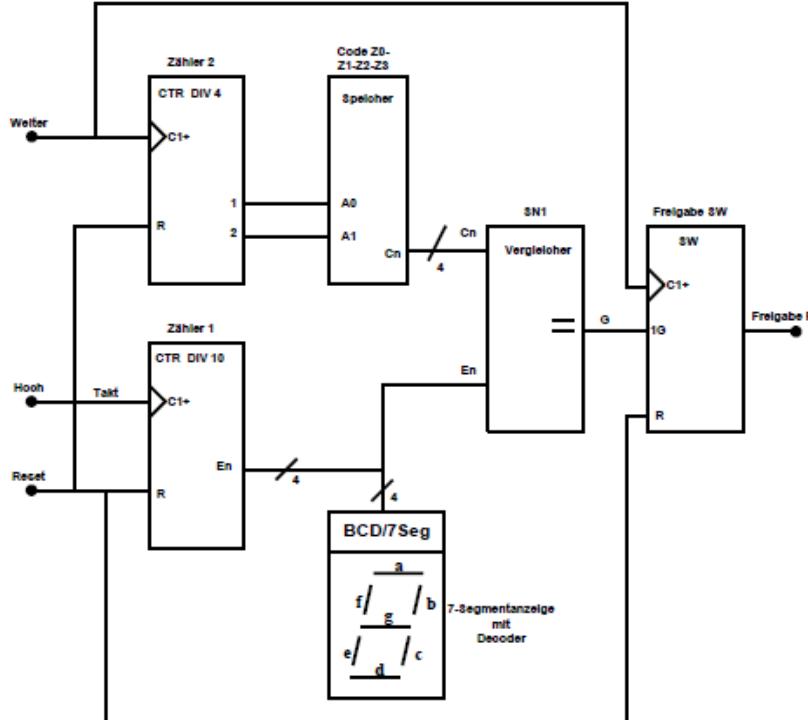
Der Besitzer erzeugt seine Ziffern mit dem Taster **Hoch** am dualen **Zähler 1**. Das eingestellte Zeichen E und die Codezahl C werden mit dem Vergleicher **SN 1** überprüft. Mit dem Taster **Weiter** wird die Auswertung des Vergleichs im Freigabe-Schaltwerk (SW) ausgelöst.

Außerdem wird mit dem Taster **Weiter** durch den **Zähler 2** die Adresse der nächsten Codeziffer erzeugt und vom Speicher **Code** mit den Adressleitungen **A0** und **A1** angefordert (s. Tabelle 1).

Ein Neustart der Codeeingabe erfolgt mit dem Taster Reset. Alle Taster sind prellfrei. Beim Start stehen die Zähler auf 0.

Die Anzeige der Eingabeziffern erfolgt über die 7Seg – Anzeige mit vorgeschaltetem Decoder.

Tabelle 1: Codespeicher						
A1	A0	C3	C2	C1	C0	
0	0	0	1	1	0	Z0
0	1	1	0	0	0	Z1
1	0	0	0	0	1	Z2
1	1	0	1	0	1	Z3

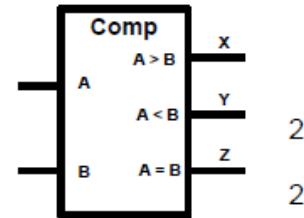


Blockschaltbild der Anlage

**Schaltnetz 1: 4-Bit-Vergleicher**

Das Schaltnetz SN1 vergleicht das eingegebene Binärzeichen E (E3, E2, E1, E0) mit dem gespeicherten Binärzeichen C (C3, C2, C1, C0). Bei E = C wird der Ausgang G = 1. Im ersten Schritt sollen Sie einen 1-Bit-Vergleicher untersuchen.

- 1.1.1 Geben Sie die Funktionstabelle des 1-Bit-Vergleichers an.
- 1.1.2 Verwenden Sie den nebenstehenden Baustein, um einen 4-Bit-Vergleicher zu entwerfen, der die beiden Eingangszahlen auf **Gleichheit** überprüft.

2
2**Speicher**

- 1.2.1 Geben Sie die Codeziffern Z0, Z1, Z2 und Z3 des Herstellers dezimal an.
- 1.2.2 Nennen Sie zwei Eigenschaften eines Speicherbausteins, der für die Speicherung des Codes geeignet ist.

1
2**Schaltwerk: Freigabe**

Die Freigabe erfolgt mit einem synchronen Schaltwerk, das vom Taster Weiter getaktet wird und dessen Folgezustand vom Eingang G abhängt. Wurden alle 4 Zeichen der Reihe nach korrekt eingegeben (G=1) erfolgt die Freigabe (F=1). Wurde ein Zeichen falsch eingegeben (G = 0), beginnt das Schaltwerk wieder im Startzustand. Der Reset zum Neustart der Codeeingabe erfolgt taktunabhängig.

- 1.3.1 Zeichnen Sie das Zustandsdiagramm für das Freigabe-Schaltwerk.
- 1.3.2 Bestimmen Sie die codierte Zustandsübergangstabelle für das Freigabe-Schaltwerk.
- 1.3.3 Ermitteln Sie die Schaltfunktion für die Freigabe F.

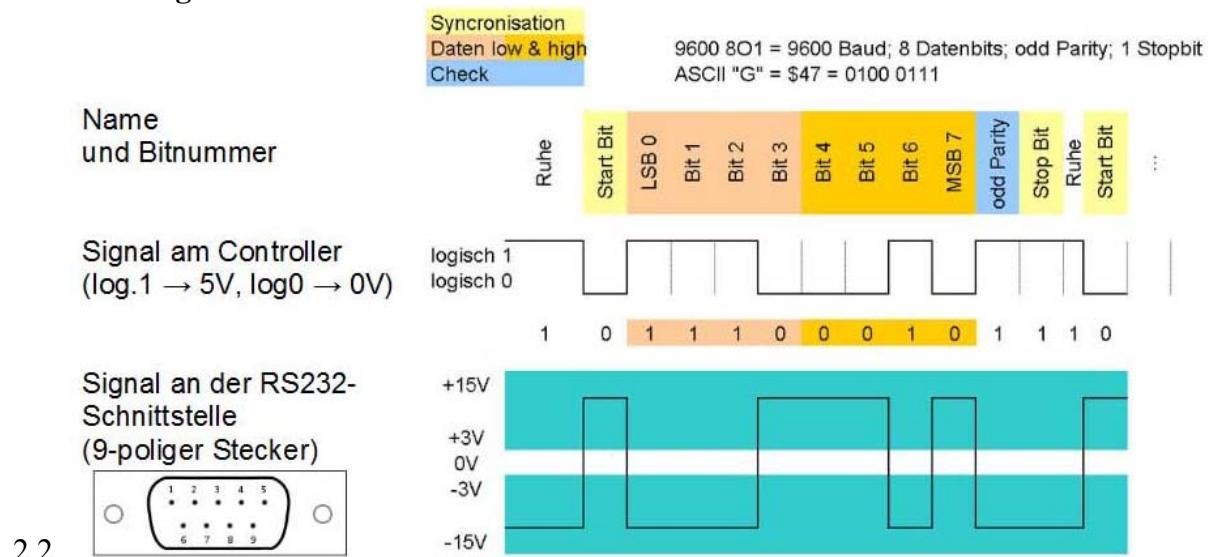
3
3
1**Mikrocontroller**

Die Eingabe der Zeichen und der Vergleich mit den 4 Codeziffern des Herstellers erfolgt nun mit einem Mikrocontroller. Die Taster „Weiter“ und „Hoch“ werden hierbei auf externe Interrupts gelegt. Der vierstellige Code ist im Programmspeicher in einer Tabelle „codesp“ abgelegt. Für die folgenden Aufgaben ist es Ihnen freigestellt, ob Sie den geforderten Programmcode in C oder Assembler schreiben.

- 1.4.1 Legen Sie die genauen Portbezeichnungen für Ihren Mikrocontroller fest.
- 1.4.2 Schreiben Sie das Hauptprogramm in C oder Assembler mit folgenden Aufgaben:
 - Initialisierung der externen Interrupts
 - Angabe der Tabelle „codesp“ nach Tabelle 1
 - Deklaration und Initialisierung der notwendigen Variablen
 - Kommentieren Sie das Programm aussagekräftig
- 1.4.3 Schreiben Sie die Interrupt-Service-Routine zum Einstellen der Zeichen (0-9) mit dem Taster „Hoch“ in Assembler oder C. Der eingestellte Wert ist für eine BCD-codierte 7-Segmentanzeige an einem freien Port auszugeben.
Hinweis: Nach der Ziffer 9 folgt wieder die Ziffer 0.
- 1.4.4 Entwickeln Sie einen PAP oder ein Struktogramm für die Interrupt-Serviceroutine für den Taster „Weiter“. Die Aufgabe der Serviceroutine ist es die eingegebenen Ziffern mit den vorgegebenen Ziffern im Codespeicher zu vergleichen. Wurden alle 4 Ziffern korrekt eingegeben, ist der Freigabe-Pin auf 1 zu setzen und die ISR zu deaktivieren. Bei falscher Eingabe ist ein neuer Versuch nur nach einem Reset möglich.
- 1.4.5 Schreiben Sie die Interrupt-Serviceroutine für den Taster „Weiter“ in Assembler oder C.

1
5
3
4
3

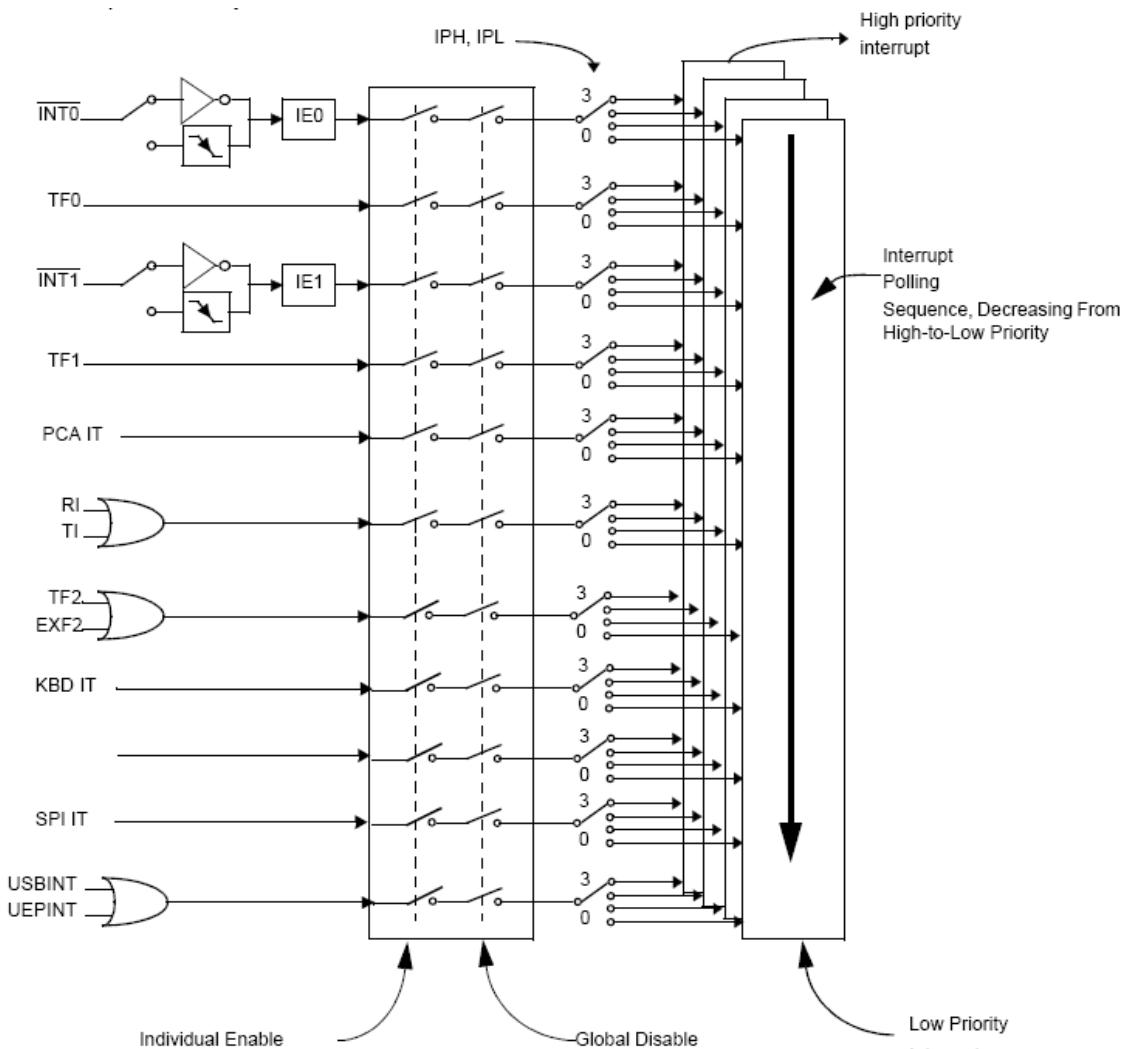
30

**12. Serielle Datenübertragung mittels RS232; Anwendung Timer...****12.1. Allgemeines zur Funktionsweise der RS232**

2.2.

- ✓ Seriell = _____
- ✓ Grundzustand der Leitung ist _____
- ✓ Setzt der Sender (TX) die Leitung auf log0 (Startbit) so erkennt der Receiver _____

- ✓ Danach werden 8 Datenbits (Bit0-Bit7) gesendet, hier Buchstabe _____
- ✓ Anschließend kann ein Paritätsbit gesendet werden = _____
- ✓ Dann erfolgt mind. 1 Stoppbit = _____
- ✓ Übertragungsrate legt fest _____
- ✓ Wir arbeiten oft mit 9600 Bit/s = _____
- ✓ Das Ein- und Ausgabe-Register für die serielle Schnittstelle ist _____
- ✓ Der serielle Interrupt **ES** hat die Nummer _____



- ✓ **RS232 senden:** Wird SBUF ein Wert zugewiesen, wird der Wert gesendet und durch das Senden des Stop-Bits löst das **TI**-Flag den Interrupt **ES** aus. Damit wird das Ende des Sendens angezeigt.

Entwurf RS232-Senden = TX Routine mit printf:

- ✓ **RS232 empfangen:** Wird ein Wert empfangen, ist der Wert nach dem Stoppbit in SBUF. Der Interrupt **ES** wird durch das Setzen des **RI**- Flag ausgelöst. Die Flags müssen in der Interrupt-Routine ausgewertet und nach der Auswertung durch das Programm zurückgesetzt werden.

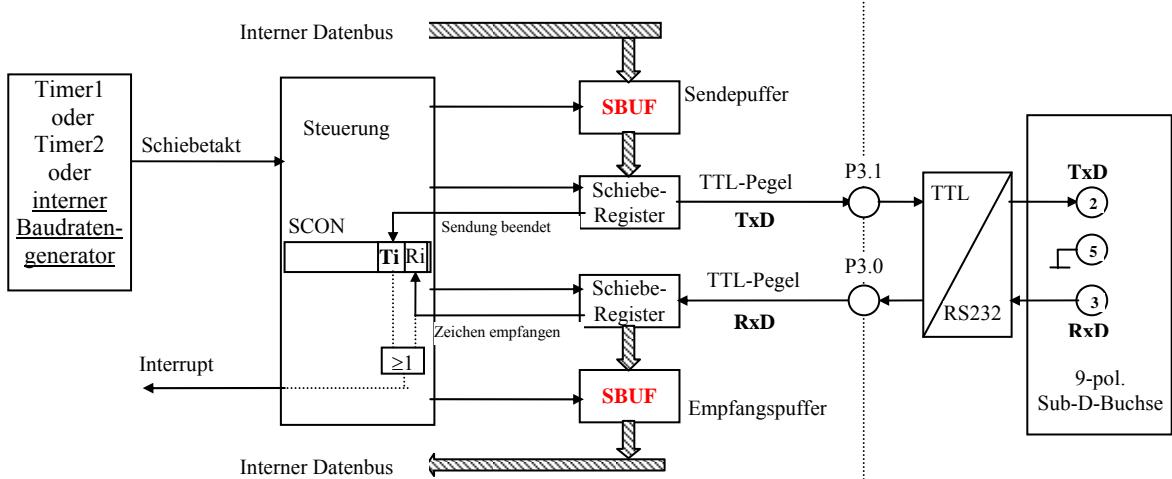
Entwurf RS232-Empfangen = RX ISR_Routine mit SBUF:



ASCII-Tabelle = Nur ASCII Zeichen werden übertragen!

ASCII-Code										DIN 66 003: 74-06									
Spalte Zeile	00		01		02		03		04		05		06		07				
00	NUL	0	DLE	10	SP	20	0	30	@	40	P	50	'	60	p	70			
		0		16		32	0	48		64		80		96	112				
01	SOH	01	DC ₁	11	!	21	1	31	A	41	Q	51	a	61	q	71			
		1		17		33	41	49		65		81		97	113				
02	STX	02	DC ₂	12	"	22	2	32	B	42	R	52	b	62	r	72			
		2		18		34	50	50		66		82		98	114				
03	ETX	03	DC ₃	13	#	23	3	33	C	43	S	53	c	63	s	73			
		3		19		35	51	51		67		83		99	115				
04	EOT	04	DC ₄	14	\$	24	4	34	D	44	T	54	d	64	t	74			
		4		20		36	52	52		68		84		100	116				
05	ENQ	05	NAK	15	%	25	5	35	E	45	U	55	e	65	u	75			
		5		21		37	53	53		69		85		101	117				
06	ACK	06	SYN	16	&	26	6	36	F	46	V	56	f	66	v	76			
		6		22		38	54	54		70		86		102	118				
07	BEL	07	ETB	17	>	27	7	37	G	47	W	57	g	67	w	77			
		7		23		39	55	55		71		87		103	119				
08	BS	08	CAN	18	(28	8	38	H	48	X	58	h	68	x	78			
		8		24		40	56	56		72		88		104	120				
09	HT	09	EM	19)	29	9	39	I	49	Y	59	i	69	y	79			
		9		25		41	57	57		73		89		105	121				
10	LF	0A	SUB	1A	*	2A	:	3A	J	4A	Z	5A	j	6A	z	7A			
		10		26		42	58	58		74		90		106	122				
11	VT	0B	ESC	1B	+	2B	;	3B	K	4B	[5B	k	6B	{	7B			
		11		27		43	59	59		75		91		107	123				
12	FF	0C	FS	1C	*	2C	<	3C	L	4C	\	5C	l	6C	:	7C			
		12		28		44	60	60		76		92		108	124				
13	CR	0D	GS	1D	-	2D	=	3D	M	4D]	5D	m	6D	}	7D			
		13		29		45	61	61		77		93		109	125				
14	SO	0E	RS	1E	.	2E	>	3E	N	4E	^	5E	n	6E	~	7E			
		14		30		46	62	62		78		94		110	126				
15	S	0F	US	1F	/	2F	?	3F	O	4F	-	5F	o	6F	DEL	7F			
		15		31		47	63	63		79		95		111	127				
	P000 1111	017	P001 1111	037	P010 1111	057	P011 1111	077	P100 1111	117	P101 1111	137	P110 1111	157	P111 1111	177			

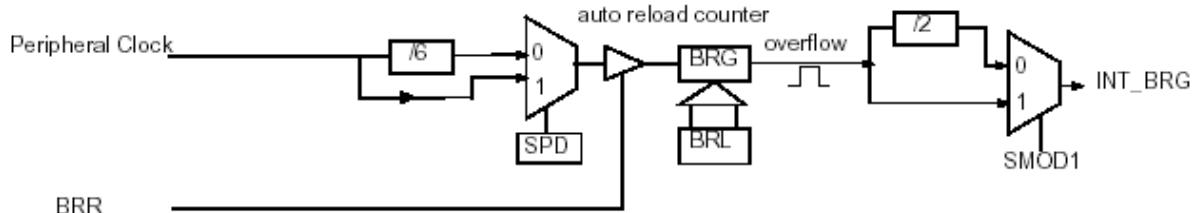
Beispiel zur UART = universal asynchronous receive transmit für RS232-Schnittstelle:



**Berechnung für das Timing: Wie schnell werden die Bits rausgeschickt?**

Als Baudratengenerator können der Timer1, Timer2 oder der interne Baudratengenerator verwendet werden. Im Unterricht wird der interne **BaudRatenGenerator (INT_BRG)** und der **Mode_1** (8-Bit-Übertragung) verwendet.

Internal Baud Rate



$$\text{BRL}_{\text{=Baudreloadwert}} = 256 - (2^{\text{SMOD1}} * \text{Fper}) / (6^{(1-\text{SPD})} * 32 * \text{Baudrate}) = 217$$

SPD = Speed Control Bit = 1 = Fast mode

Fper = Fquarz/2 = 6MHz wenn kein x2-Mode (bei X2-Mode Fper = 12 MHz)

SMOD1 = 1 für UART-Mode 1 = internal BRG, 8 Datenbits

Berechnung BRL für Baudrate 2400:

Optional: Welche maximale Baudrate kann mit einem 12Mhz-Quarz erreicht werden:



Übung RS232_IR_Test

Erstellen Sie ein Struktogramm sowie ein Programm in RIDE. Welches bei Tastendruck (S2 Interrupt) den Text *Wert_TX: var* über die RS232 Schnittstelle mit 9600 Baud, N, 8 1 überträgt und durch RS232-RX Interrupt den Text *Hello* vom PC empfängt. Nutzen Sie zur Baudrateeinstellung die Datei RS232_init.c und binden diese ein.

Geben Sie am LCD (1) die Zeichen für TX aus und am LCD(2) die empfangenen. Am PC können Sie das Tool Docklight zur Kommunikation benutzen.

Optional: Senden Sie im Sekundentakt die Uhrzeit über RS232 an den PC

**12.2. Beispiel zur seriellen Datenerfassung ohne UART****Beschreiben Sie die jeweilige Funktion mit eigenen Worten:****Serielles einlesen von Daten...**

EX0 = 1;
IT0 = 1;
EA = 1;

P3.2

µC

void ISR_RX (void) interrupt 0

data <<=1;

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

data = data |= P3.2;

0	0	0	0	0	0	0	h
---	---	---	---	---	---	---	---

void ISR_RX (void) interrupt 0

data <<=1;

0	0	0	0	0	0	h	0
---	---	---	---	---	---	---	---

data = data |= P3.2;

0	0	0	0	0	0	h	g
---	---	---	---	---	---	---	---

void ISR_RX (void) interrupt 0

data <<=1;

0	0	0	0	0	h	g	0
---	---	---	---	---	---	---	---

data = data |= P3.2; ...

0	0	0	0	0	h	g	f
---	---	---	---	---	---	---	---

Beschreiben Sie die Funktion:**Serielles einlesen von Daten...**

EX0 = 1;
IT0 = 1;
EA = 1;

P3.2

µC

void ISR_RX (void) interrupt 0

EX0 = 0;

0	0	0	0	0	0	0	0
---	---	---	---	---	---	---	---

data <<=1;

0	0	0	0	0	0	0	h
---	---	---	---	---	---	---	---

data = data |= P3.2;

0	0	0	0	0	0	0	h
---	---	---	---	---	---	---	---

data <<=1;

0	0	0	0	0	0	h	0
---	---	---	---	---	---	---	---

data = data |= P3.2;

0	0	0	0	0	0	h	g
---	---	---	---	---	---	---	---

data <<=1;

0	0	0	0	0	h	g	0
---	---	---	---	---	---	---	---

data = data |= P3.2;...

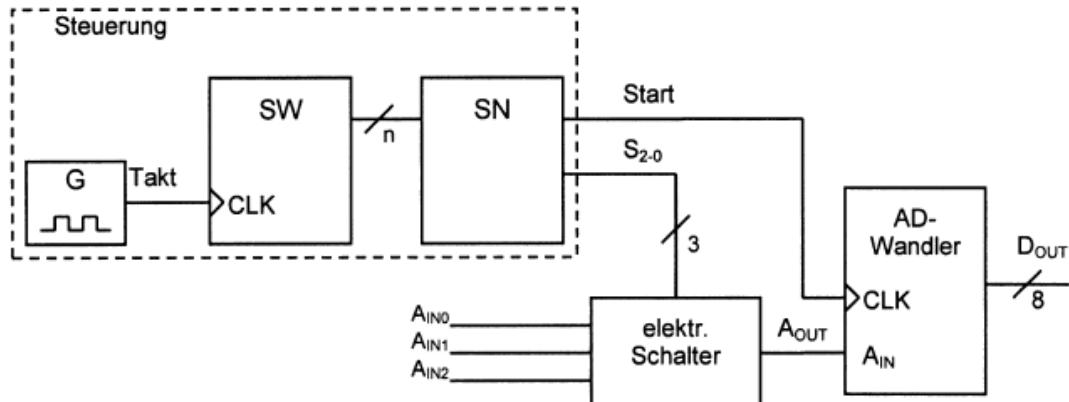
0	0	0	0	0	h	g	f
---	---	---	---	---	---	---	---

EX0 = 1;

Beschreiben Sie die Funktion:**Worin unterscheidet sich die Funktion der beiden Programme:**



Hauptprüfung 2011/2012	Berufliches Gymnasium (TG)	
1.5.2	Informationstechnik (Hardware)	
	Teil: 1 (Pflichtbereich)	Aufgabe: 1 (2 Seiten)

Messdatenerfassung

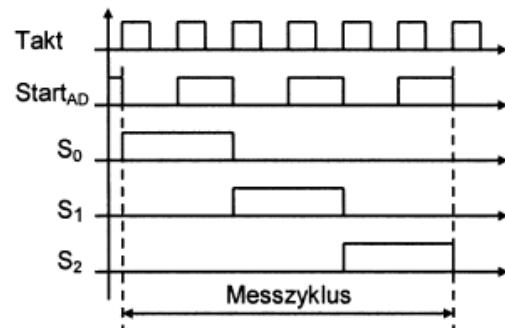
Drei analoge Messsignale (A_{IN0} , A_{IN1} , A_{IN2}) sollen fortlaufend mit einem AD-Wandler digitalisiert werden. Dazu werden die drei Messkanäle über einen elektronischen Schalter, wie in der Funktionstabelle rechts gezeigt, dem AD-Wandler im zeitlichen Wechsel zugeschaltet.

Funktionstabelle: elektr. Schalter			
Steuersignale			
S_0	S_1	S_2	Schaltverbindung
0	0	0	keine Verbindung
1	0	0	A_{IN0} nach A_{OUT}
0	1	0	A_{IN1} nach A_{OUT}
0	0	1	A_{IN2} nach A_{OUT}

Der AD-Wandler digitalisiert das Analogsignal an seinem Eingang A_{IN} mit jeder positiven Taktflanke von Start (=CLK-Eingang des AD-Wandlers).

Die hierfür erforderlichen Signale Start und $S_0..S_2$ erzeugt die Steuerung.

Im nebenstehenden Zeitablaufdiagramm sind die Ausgangssignale der Steuerung für einen kompletten Messzyklus dargestellt.

**1.1 Die Steuerung der Messdatenerfassung wird nun genauer untersucht.**

- | | |
|--|---|
| 1.1.1 Entwerfen Sie das Zustandsdiagramm für die Steuerung (SW+SN) und geben Sie im Zustandsdiagramm die Ausgangssignale Start S_0 , S_1 und S_2 der Steuerung an. | 2 |
| 1.1.2 Bestimmen Sie die Mindestanzahl der Speicher (Flip-Flops) für das Schaltwerk und kodieren Sie entsprechend die Zustände. | 3 |
| 1.1.3 Mit welchem Baustein lässt sich das SW am einfachsten realisieren? | 1 |
| 1.1.4 Bestimmen Sie die Funktionstabelle für das Ausgangsschaltnetz SN. | 2 |
| 1.1.5 Ermitteln Sie die vollständige und die vereinfachte Funktionsgleichung für S_1 | 2 |

1.2 Das Ausgangsschaltnetz SN soll nun entfallen.

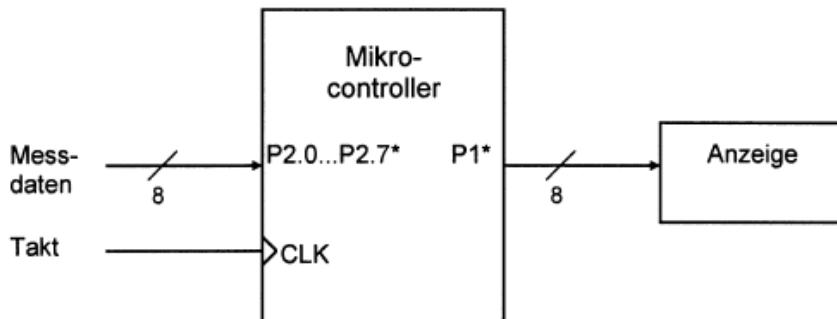
- | | |
|--|---|
| 1.2.1 Ermitteln Sie wie viele Speicher das Schaltwerk SW jetzt benötigt. | 1 |
| 1.2.2 Erstellen Sie die neue Zustandsfolgetabelle für das SW. | 3 |



Messdatenerfassung und -verarbeitung mit dem Mikrocontroller

Mit einem Mikrocontroller sollen die digitalisierten Messdaten erfasst und verarbeitet werden. Dazu werden nacheinander 8 Datenwerte an Port P2* mit einer Taktfrequenz von 10 kHz eingelesen und im internen Datenspeicher gespeichert. Das Einlesen der Datenwerte erfolgt interruptgesteuert durch einen Timer.

Nach der Datenerfassung soll der arithmetische Mittelwert der Messdaten berechnet und das Ergebnis als 8-Bit-Dualzahl an Port P1* ausgegeben werden. Die Mittelwertberechnung soll als Unterprogramm „calc“ aus der Timer-ISR aufgerufen werden.



*Für andere Controller wählen Sie bitte einen passenden Port.

- | | | |
|-------|--|---|
| 1.3.1 | Bestimmen Sie die Zeit, die zwischen dem Einlesen zweier Messwerte vergeht. | 1 |
| 1.3.2 | Bestimmen Sie für ihren Mikrocontroller eine geeignete Betriebsart ihres Timers aus und schreiben Sie die notwendige Initialisierung für den Timer in der Programmiersprache C oder in Assembler. Die Programmzeilen sind zu kommentieren. Begründen Sie Ihre Wahl der Betriebsart. Es kann davon ausgegangen werden, dass ein Maschinenzyklus 1µs benötigt. | 3 |
| 1.3.3 | Entwickeln Sie den Programmcode für die ISR des Timer-Interrupts in Assembler. Wählen Sie einen geeigneten Speicherbereich für die Messdaten und begründen Sie Ihre Wahl. | 4 |
| 1.3.4 | Entwickeln Sie einen Programmablaufplan (PAP) oder ein Struktogramm für das Unterprogramm „calc“, das den Mittelwert der 8 gespeicherten Messdaten berechnet und ausgibt.
Zur Vereinfachung der Mittelwertberechnung wird davon ausgegangen, dass die einzelnen Messdaten kleiner 32 sind, so dass die Summe aller Daten stets kleiner 255 ist. | 4 |
| 1.3.5 | Bestimmen Sie die Auswirkungen für ein Assemblerprogramm entsprechend der Aufgabe 1.3.4, wenn die Datenwerte nicht auf Werte kleiner 32 beschränkt sind. | 1 |
| 1.3.6 | Erläutern Sie einen Lösungsansatz für Assembler mit dem 8-Bitwerte ohne Größenbeschränkung addiert werden können. | 2 |
| 1.3.7 | Nennen Sie diesbezüglich einen wesentlichen Vorteil einer Lösung in C gegenüber Assembler für „calc“. | 1 |



Übung Mikrocontroller

Matrixtastatur

Eine Matrixtastatur besteht aus horizontalen- und vertikalen Leiterbahnen die sich nicht berühren. Durch Drücken einer Taste wird eine Verbindung zwischen der entsprechenden horizontalen und vertikalen Leiterbahn hergestellt. Legt man an die horizontalen Leiterbahnen ein bestimmtes Bitmusters (Zeilencode) an, lässt sich durch Abfragen der Logikzustände der vertikalen Leiterbahnen (Spaltencode) erkennen welche Taste betätigt wurde.

Beispiel:

Wenn der Zeilencode Z3 – Z0 gleich „0100“ und der daraus folgende Spaltencode SP2 – SP0 gleich „010“ ist, ist die Taste 5 betätigt.

Z3	1	2	3
Z2	4	5	6
Z1	7	8	9
Z0	*	0	#

SP2 SP1 SP0

1.1 Tastaturabfrage

Welche Bitmuster müssen für die Signale Z3 – Z0 angelegt werden, damit die Abfrage der Signale SP2 – SP0 ein eindeutiges Ergebnis liefert, also die betätigte Taste eindeutig identifizierbar ist.

1.2 Funktionstabelle

Jeder Taste sind eindeutig die Signale Z3-Z0 und SP2-SP0 zuzuordnen. Erstellen sie eine Funktionstabelle für die Tasten 3, 4, 9, 0. Verwenden Sie als Grundlage die folgende Anordnung:

Taste	Z3	Z2	Z1	Z0	SP2	SP1	SP0
Keine	X	X	X	X	0	0	0
5	0	1	0	0	0	1	0

1.3 Schaltwerk

Zeichnen Sie ein Blockschaltbild des Schaltwerks, das den für die Zeilenansteuerung erforderlichen 1-aus-4-Code, realisiert.

1.4 Prüfschaltung

Wenn der Spaltencode mehr als eine logische 1 aufweist, ist die betätigte Taste nicht mehr eindeutig erkennbar. Zeichnen Sie eine Prüfschaltung mit dem Ausgang P, die im Fehlerfall eine logische 0 führt.

1.5 Umwandlung

Die Tabelle aus 1.2 wird mit Hilfe einer Codierschaltung in ein 4-Bit-Muster umgewandelt, so dass jeder Taste ein eigenes 4-Bit-Muster zugeordnet ist. Dieses soll aber nur dann sichtbar sein, wenn die Taste eindeutig erkennbar ist (siehe 1.4) und eine Taste betätigt wurde. Zeichnen Sie ein Blockschaltbild der Schaltung.

Hinweis: Die Codierschaltung ist nicht zu entwickeln.



Matrixtastatur am Mikrocontroller

Die Matrixtastatur wird am Port 1 eines Mikrocontrollers entsprechend der folgenden Tabelle angeschlossen.

Port 1	Bit	7	6	5	4	3	2	1	0
	Funktion	n.c.	SP2	SP1	SP0	Z3	Z2	Z1	Z0

Zur Abfrage der Tastatur muss bei anlegen des entsprechenden Zeilencodes der Spaltencode ausgelesen werden.

Beispiel:

Zeilencode anlegen: P1 = 08H

Wenn Taste 1 betätigt wurde ergibt das Einlesen von Port 1 den Wert 48H.

2.1 Abfrage der Matrixtastatur

Erstellen Sie einen PAP zur Abfrage der Matrixtastatur. Hierbei muss nacheinander jede Zeile ausgewählt und dann die Spalten eingelesen werden. Der eingelesene 7-Bit-Code der Taste soll auf Port 3 ausgegeben werden.

2.2 Assemblerprogramm

Entwickeln Sie ein Assemblerprogramm zu 2.1.

2.3 UP Ausgabe

Ihr Programm gibt nun nacheinander die Spaltencodes für die entsprechenden Zeilen aus. Eine gedrückte Taste wird nur kurz angezeigt. Nach dem Loslassen der Taste erscheint immer der Code für „Keine-Taste“. Durch welche Maßnahme lässt sich erreichen, dass auf Port 3 immer der Tasten-Code der zuletzt betätigten Taste angezeigt wird.

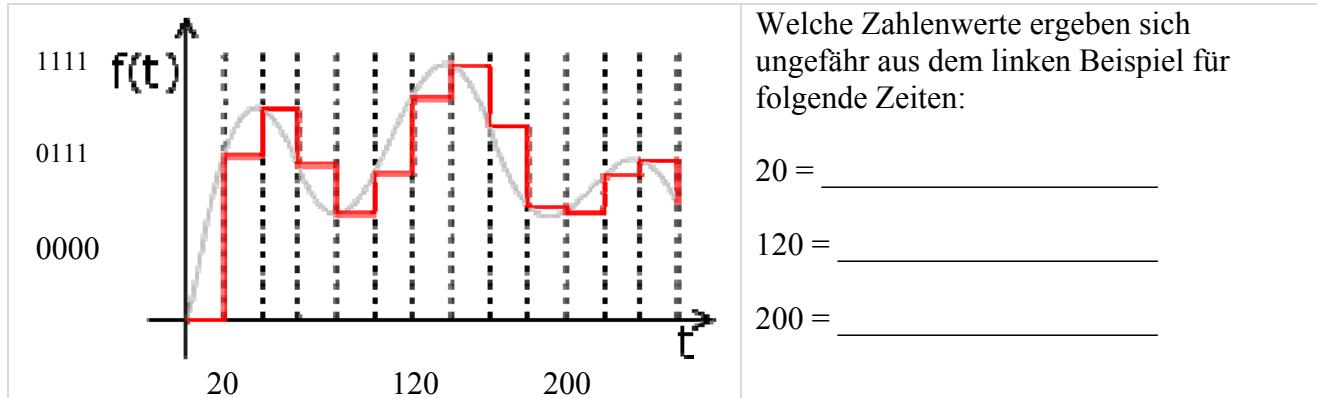
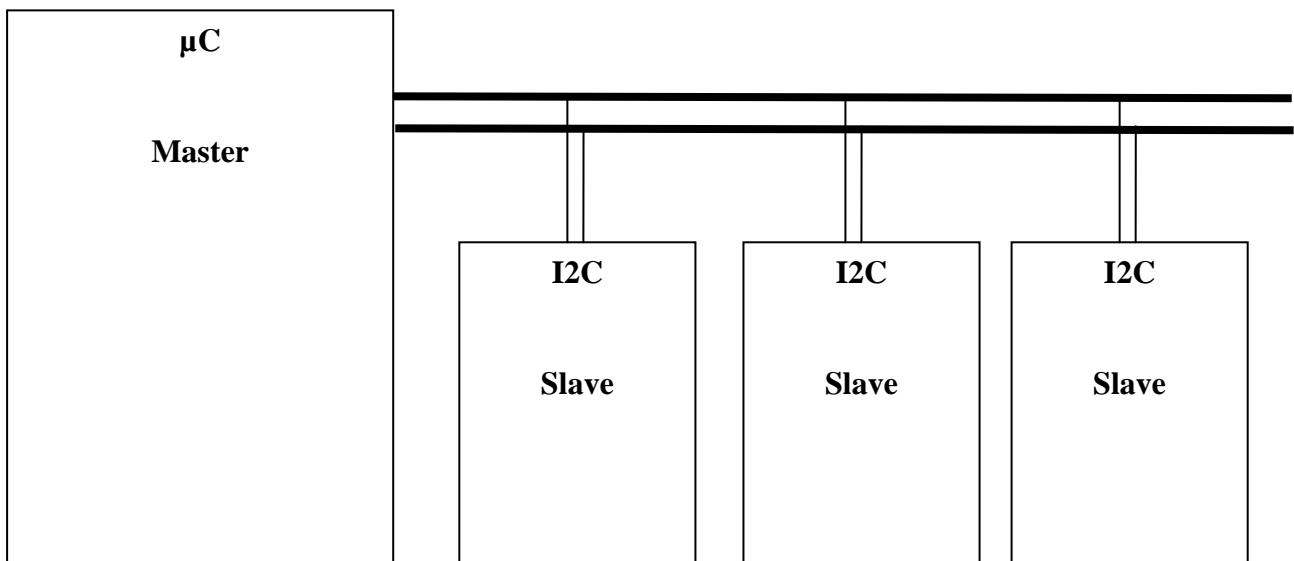
Zeichnen Sie einen PAP für ein entsprechendes Unterprogramm „Ausgabe“.

2.4 Assemblerprogramm

Schreiben Sie ein Assemblerprogramm das die geforderte Maßnahme aus 2.3 realisiert.

**13. AD-Wandlung mit I2C-Bus:****13.1. AD-Wandlung, Prinzip:**

Beispiel für gewandelte Analogspannung:

**13.2. IIC-Bus = I2C-Bus, I²C-Bus = 2-wire-Bus, Prinzip:**

Welche Komponenten / Parameter gehören zu einem funktionierenden Bussystem (Info wikipedia):



Wikipedia.org Auszug zum I²C-Bus:

Bussystem

I²C ist als [Master-Slave](#)-Bus konzipiert. Ein Datentransfer wird immer durch einen Master initiiert; der über eine Adresse angesprochene Slave reagiert darauf. Mehrere Master sind möglich ([Multimaster-Mode](#)). Im Multimaster-Mode können zwei Master-Geräte direkt miteinander kommunizieren, dabei arbeitet ein Gerät als Slave. Die [Arbitrierung](#) (Zugriffsregelung auf den Bus) ist per Spezifikation geregelt.

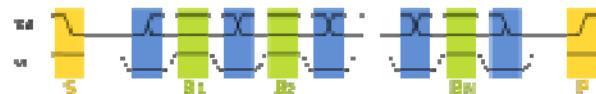
Elektrische Definition



I²C-Bus mit einem Master und drei Slaves

Im Diagramm rechts sind drei Geräte eingezeichnet. I²C benötigt zwei Signalleitungen: [Takt](#) (SCL) und [Datenleitung](#) (SDA). Beide liegen mit den [Pull-up-Widerständen](#) R_P an der Versorgungsspannung V_{DD} . Sämtliche daran angeschlossenen Geräte haben [Open-Collector-Ausgänge](#), was zusammen mit den Pull-up-Widerständen eine [Wired-AND-Schaltung](#) ergibt. Der High-Pegel soll mindestens $0,7 \times V_{DD}$ betragen und der Low-Pegel soll höchstens bei $0,3 \times V_{DD}$ liegen. Die im Bild nicht eingezeichneten Serienwiderstände R_S an den Eingängen der Geräte sind optional und werden als Schutzwiderstände verwendet. Der I²C-Bus arbeitet mit [positiver Logik](#), d.h. ein HIGH-Pegel auf der Datenleitung entspricht einer logischen '1', der LOW-Pegel einer '0'.

Takt und Zustände des Busses



Zeitverhalten am I²C-Bus: Zwischen dem *Start-Signal* (S) und dem *Stopp-Signal* (P) werden die Datenbits B_1 bis B_N übertragen.

Der Bustakt wird immer vom Master ausgegeben. Für die verschiedenen Modi ist jeweils ein maximal erlaubter Bustakt vorgegeben. In der Regel können aber auch beliebig langsamere Taktraten verwendet werden, falls diese vom Master-Interface unterstützt werden. Bestimmte ICs (z.B. A/D-Umsetzer) benötigen jedoch eine bestimmte, minimale Taktfrequenz, um ordnungsgemäß zu funktionieren. Die folgende Tabelle listet die maximal erlaubten Taktraten auf.

Modus	Maximale Taktrate
Standard Mode	100 kHz
Fast Mode	400 kHz
Fast Mode Plus	1 MHz
High Speed Mode	3,4 MHz



Daten ([Einzelbits](#)) sind nur gültig, wenn sich ihr logischer Pegel während einer Clock-High-Phase nicht ändert. Ausnahmen sind das *Start*-, *Stop*- und *Repeated-Start*-Signal. Das *Start*-Signal ist eine fallende Flanke auf SDA, während SCL high ist, das *Stop*-Signal ist eine steigende Flanke auf SDA, während SCL high ist. *Repeated-Start* sieht genauso aus wie das *Start*-Signal.

Eine Dateneinheit besteht aus 8 Datenbits = 1 [Oktett](#) (welche protokollbedingt entweder als Wert oder als Adresse interpretiert werden) und einem [ACK](#)-Bit. Dieses Bestätigungsbit (Acknowledge) wird vom Slave durch einen Low-Pegel auf der Datenleitung während der neunten Takt-High-Phase (welche nach wie vor vom Master generiert wird) und als NACK (für engl. *not acknowledge*) durch einen High-Pegel signalisiert. Der Slave muss den Low-Pegel an der Datenleitung anlegen *bevor* SCL auf high geht, andernfalls würden weitere eventuelle Teilnehmer ein *Start*-Signal lesen.

Adressierung

Eine Standard-I²C-Adresse ist das erste vom Master gesendete Byte, wobei die ersten sieben Bit die eigentliche Adresse darstellen und das achte Bit (R/W-Bit) dem Slave mitteilt, ob er Daten vom Master empfangen soll (LOW), oder Daten an den Master zu übertragen hat (HIGH). I²C nutzt daher einen [Adressraum](#) von 7 Bit, was bis zu 112 Knoten auf einem Bus erlaubt (16 der 128 möglichen Adressen sind für Sonderzwecke reserviert).

Jedes I²C-fähige [IC](#) hat eine vom Hersteller festgelegte Adresse, von der bisweilen die untersten drei Bits (*Subadresse* genannt) über drei Steuerpins festgelegt werden können. In diesem Fall können bis zu acht gleichartige ICs an einem I²C-Bus betrieben werden. Wenn nicht, müssen mehrere gleiche ICs mit getrennten I²C-Bussen angesteuert oder abgetrennt werden können.

Wegen Adressknappheit wurde später eine 10-Bit-Adressierung eingeführt. Sie ist abwärtskompatibel zum 7-Bit-Standard durch Nutzung von 4 der 16 reservierten Adressen. Beide Adressierungsarten sind gleichzeitig verwendbar, was bis zu 1136 Knoten auf einem Bus erlaubt.

Übertragungsprotokoll

Der Beginn einer Übertragung wird mit dem *Start*-Signal vom Master angezeigt, dann folgt die Adresse. Diese wird durch das ACK-Bit vom entsprechenden Slave bestätigt. Abhängig vom R/W-Bit werden nun Daten [byteweise](#) geschrieben (Daten an Slave) oder gelesen (Daten vom Slave).

Das ACK beim Schreiben wird vom Slave gesendet und beim Lesen vom Master. Das letzte Byte eines Lesezugriffs wird vom Master mit einem NACK quittiert, um das Ende der Übertragung anzudeuten. Eine Übertragung wird durch das *Stop*-Signal beendet. Alternativ kann auch ein *Repeated-Start* am Beginn einer erneuten Übertragung gesendet werden, ohne die vorhergehende Übertragung mit einem *Stop*-Signal zu beenden.

Alle Bytes werden dabei „[Most Significant Bit First](#)“ übertragen.

Für den *High-Speed-Mode* wird zuerst im Fast oder Standard Mode ein *Master-Code* geschickt, bevor auf die erhöhte Frequenz umgeschaltet wird.



Auszug Datenblatt PCF 8591 (A0-A2 = 0):

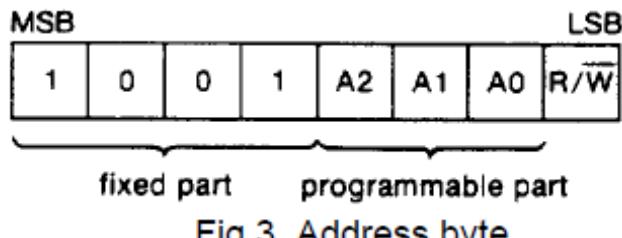


Fig.3 Address byte.

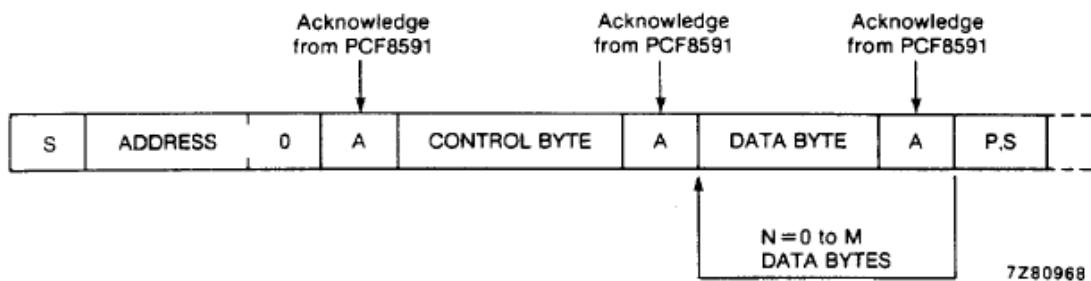


Fig.15 Bus protocol for write mode, D/A conversion.

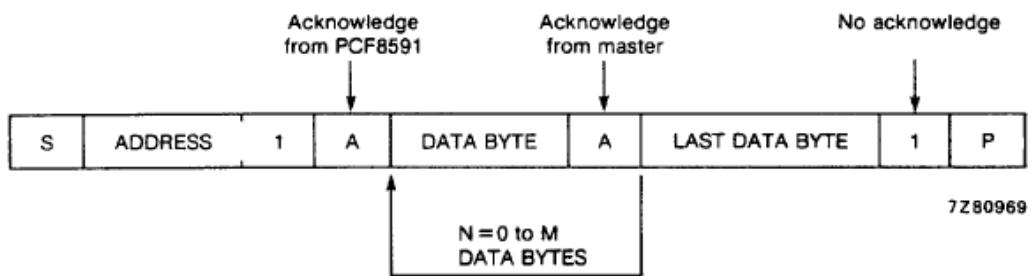
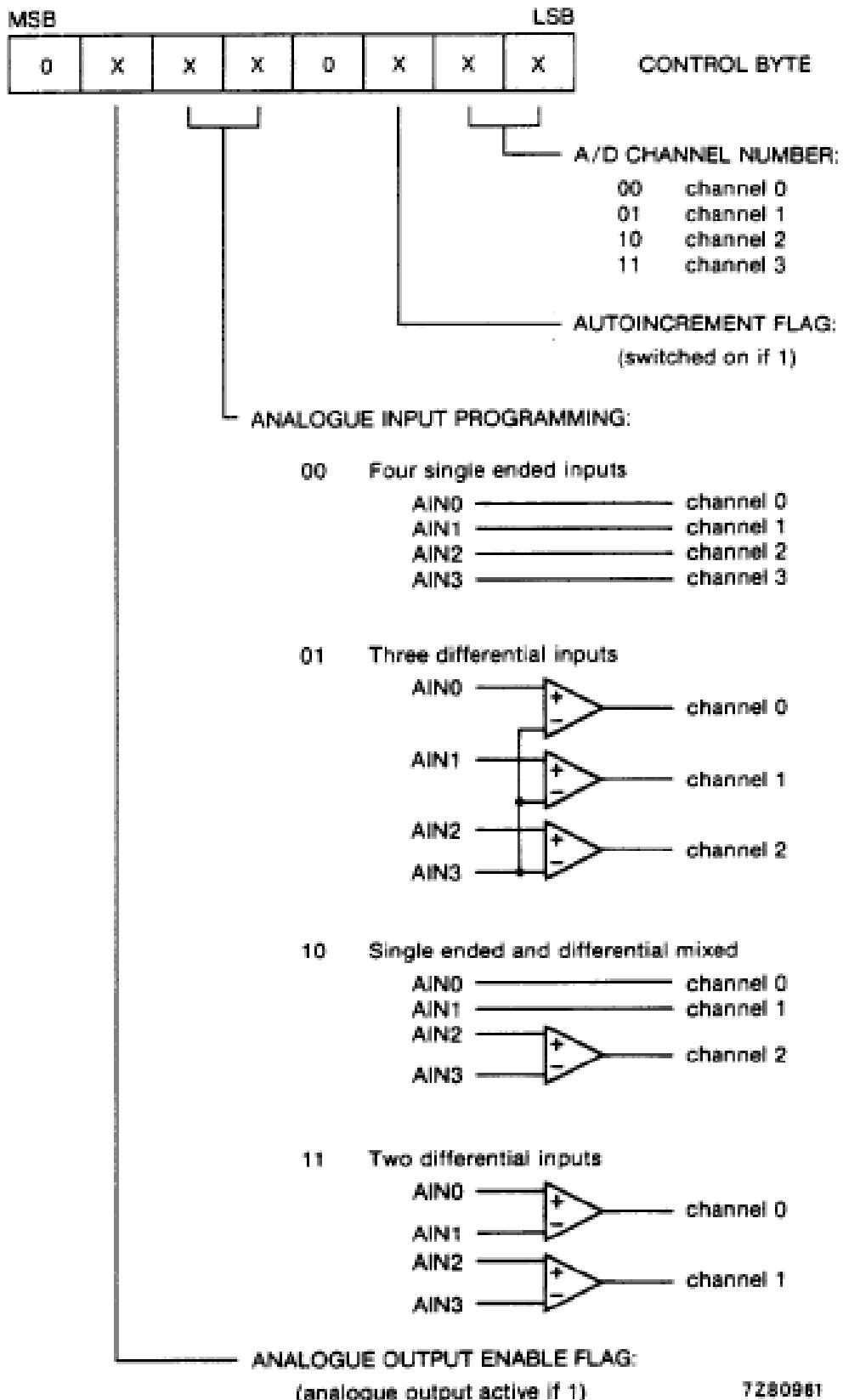


Fig.16 Bus protocol for read mode, A/D conversion.



**Übung IIC-Datenblätter:**

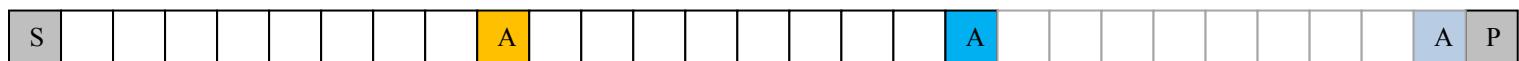
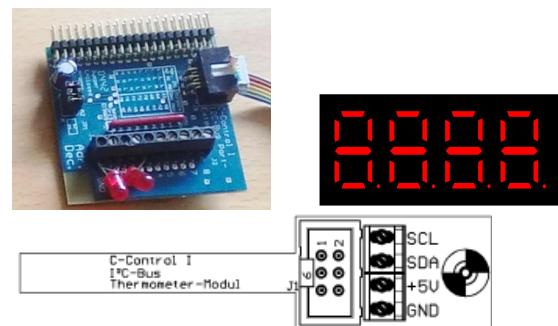
Ermitteln Sie anhand der Datenblätter die notwendigen Parameter um u.g. Aufgaben zu lösen:

Aufgaben	TX / RX	Adresse	Werte / Anzahl Bytes
Analogwert von dem AD-Wandler0 des PCF8591 einlesen			
Analogwert auf DA-Wandler PCF8591 ausgeben			
Jahreszahl für Segmenttest (alle an) an SAA1064 bzw. auf 4-fach-Siebensegmentanzeige ausgeben			
Datum (TT,MM) für Segmenttest (alle an) an SAA1064 bzw. auf 4-fach-Siebensegmentanzeige ausgeben			
Schreiben auf den IO-Expander PCF8574			
Temperaturmessung DS1631 starten			
Temperatur auslesen			
Obere Thermostat-Temperatur auf 26°C einstellen			
Untere Thermostat-Temperatur auf 24°C einstellen			

© A. Busch

Logik_Übung\Datasets\JDS\µC-IIC-Bus

Folie 12

Write mode:**Read mode:****2.3. Blockschaltbild unserer IIC-Komponenten:****13.2.1. I2C- Routinen zur Ansteuerung eines I2C-Slaves mit RIDE und i2c.c:**

- i2c_init()
- i2c_start()
- i2c_stop()
- i2c_schreiben(Wert)
- i2c_leSEN(A?)

**13.3. Übungen****13.3.1. Porterweiterungsbaustein PCF 8574:****Ermitteln Sie aus Datenblatt:**

Aufgaben	TX / RX	Adresse	Werte / Anzahl Bytes
Schreiben auf den Porterweiterungsbaustein PCF8574			

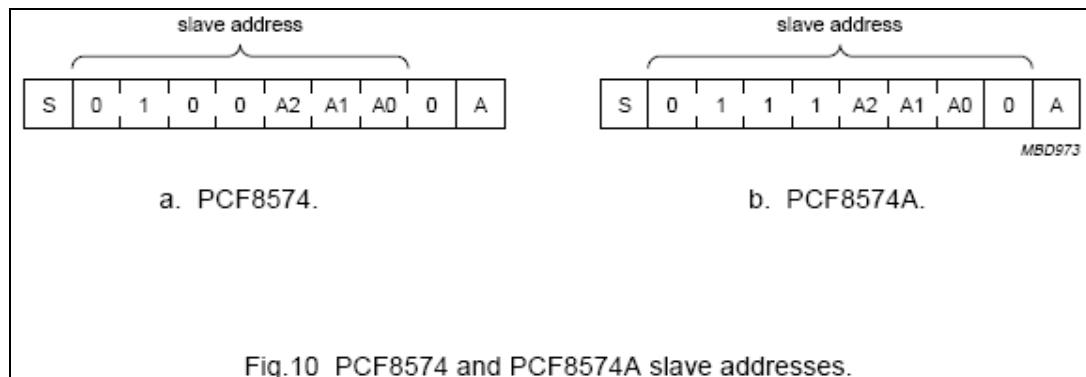
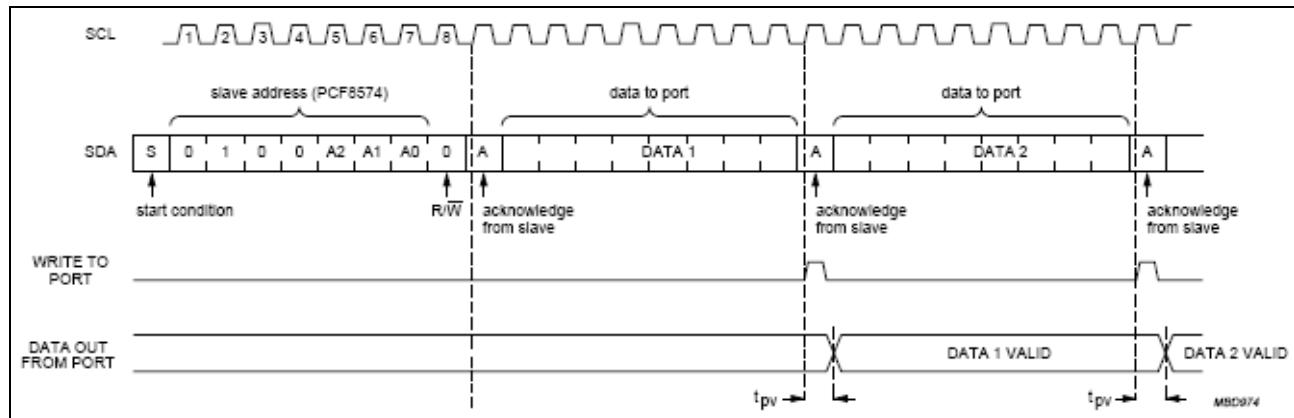
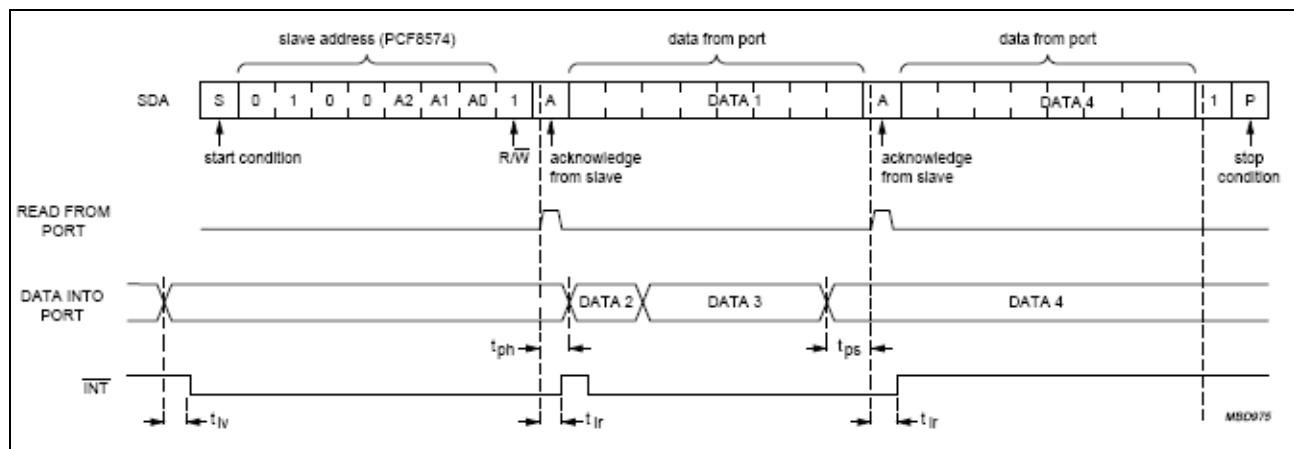
Datenblattauszug PCF8574: (A0-A2 = 0)

Fig.10 PCF8574 and PCF8574A slave addresses.



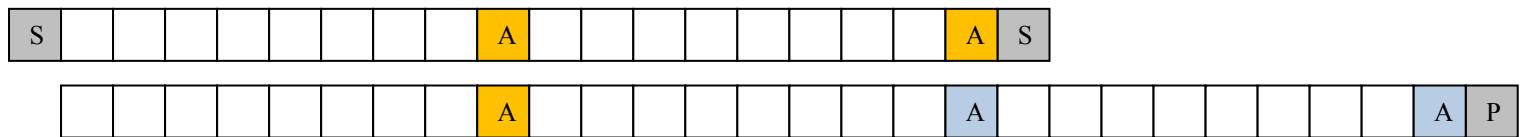
Write-Mode



Read-Mode

**13.3.2. Übung I2C_Temp (I2C-In):****Ermitteln Sie aus Datenblatt:**

Aufgaben	TX / RX	Adresse	Werte / Anzahl Bytes
I2C-Temperatur DS1631 Wandlung...			

Temp wandeln:**Temp lesen:****Datenblattauszug Thermostat / Temperatursensor DS1631 (A0-A2 = 1)****Figure 7. CONTROL BYTE**

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	0	1	A ₂	A ₁	A ₀	R/W

COMMAND SET

The DS1631/DS1631A/DS1731 command set is detailed below:

Start Convert T [51h]

Initiates temperature conversions. If the part is in one-shot mode (1SHOT = 1), only one conversion is performed. In continuous mode (1SHOT = 0), continuous temperature conversions are performed until a Stop Convert T command is issued.

Stop Convert T [22h]

Stops temperature conversions when the device is in continuous conversion mode (1SHOT = 0).

Read Temperature [AAh]

Reads last converted temperature value from the 2-byte temperature register.

Access TH [A1h]Reads or writes the 2-byte T_H register.**Access TL [A2h]**Reads or writes the 2-byte T_L register.**Access Config [ACh]**

Reads or writes the 1-byte configuration register.

Software POR [54h]

Initiates a software power-on-reset (POR), which stops temperature conversions and resets all registers and logic to their power-up states. The software POR allows the user to simulate cycling the power without actually powering down the device.



Figure 9 (a, b, c, d, e). 2-WIRE INTERFACE TIMING

THERM = DS1631, DS1631A, or DS1731

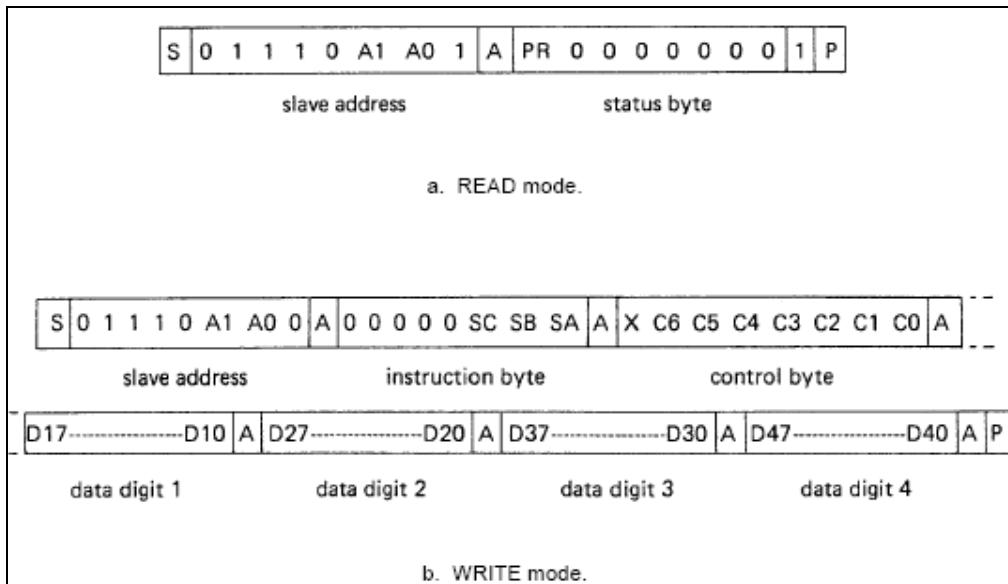
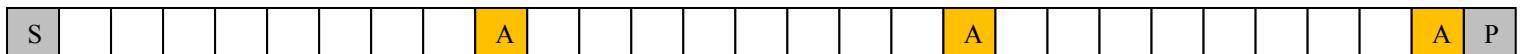




13.3.3. Übung I2C_LED_Test:

Ermitteln Sie aus Datenblatt I2C-LED (A0-A1 = 0):

Aufgaben	TX / RX	Adresse	Werte / Anzahl Bytes
I2C-7-Segment-Anzeie: LED_Test			



Subaddressing

The bits SC, SB and SA form a pointer and determine to which register the data byte following the instruction byte will be written. All other bytes will then be stored in the registers with consecutive subaddresses. This feature is called Auto-Increment (AI) of the subaddress and enables a quick initialization by the master.

The subaddress pointer will wrap around from 7 to 0.

The subaddresses are given as follows:

SC	SB	SA	SUB-ADDRESS	FUNCTION
0	0	0	00	control register
0	0	1	01	digit 1
0	1	0	02	digit 2
0	1	1	03	digit 3
1	0	0	04	digit 4
1	0	1	05	reserved, not used
1	1	0	06	reserved, not used
1	1	1	07	reserved, not used

Control bits (see Fig.4)

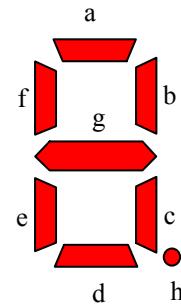
The control bits C0 to C6 have the following meaning:

- | | |
|----------|---|
| C0 = 0 | static mode, i.e. continuous display of digits 1 and 2 |
| C0 = 1 | dynamic mode, i.e. alternating display of digit 1 + 3 and 2 + 4 |
| C1 = 0/1 | digits 1 + 3 are blanked/not blanked |
| C2 = 0/1 | digits 2 + 4 are blanked/not blanked |
| C3 = 1 | all segment outputs are switched-on for segment test ⁽¹⁾ |
| C4 = 1 | adds 3 mA to segment output current |
| C5 = 1 | adds 6 mA to segment output current |
| C6 = 1 | adds 12 mA to segment output current |

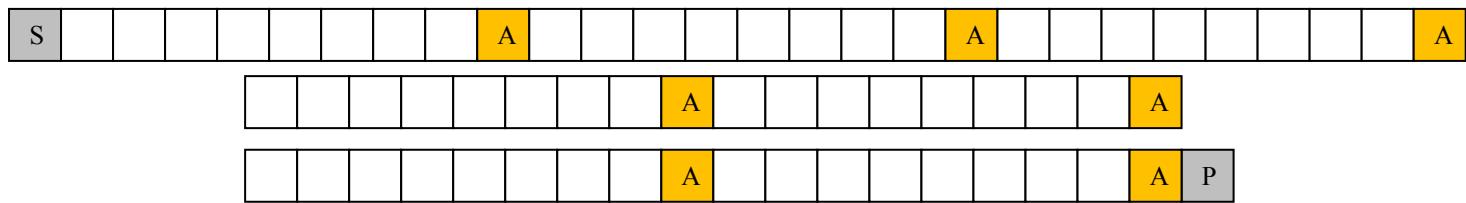
**13.3.4. Übung IIC-Jahreszahl:**

Die Anzeigeplatine mit Nummerierung der Anzeigen

Bit:	7	6	5	4	3	2	1	0
Segment:	h	g	f	e	d	c	b	a

**Ermitteln Sie aus Datenblatt I2C-LED (A0-A1 = 0):**

Aufgaben	TX / RX	Adresse	Werte / Anzahl Bytes
I2C-7-Segment-Anzeie: Jahreszahl			

**Optional: Verändern Sie die Helligkeit mittels der Stromstärke...**