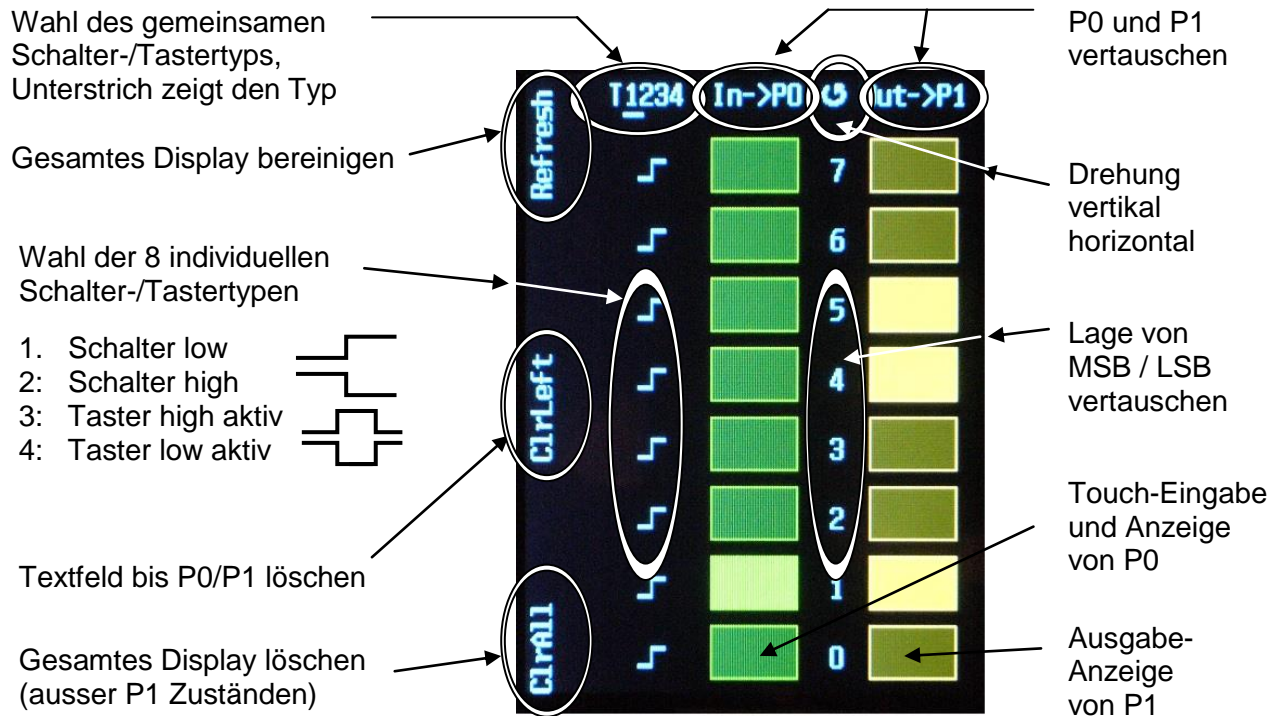


MCB32 Bibliotheken der BZU

- Touchscreenkontrolle der 'TouchP0P1' - Bibliothek



- Touchscreenkontrolle aus dem Quellcode

Der Projekt-Ordner muss **TouchP0P1.h** und **TouchP0P1.lib** enthalten:

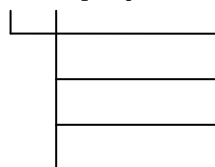
Im Projekt-Manager sind zum startupxx.s die **Quelldatei.c** und **TouchP0P1.lib** aufzunehmen:

```
/* Beschreibung der 8 Bit P0- und P1-Kontrolle über den Touchscreen des MCB32
*****/
#include <stm32f10x.h>           // Mikrokontrollertyp
#include "TouchP0P1.h"          // P0-, P1-Definition

void main(void)                 // Hauptprogramm
{
    InitTouchP0P1 ("1");        // Touchscreen aktivieren
    while(1) { }               // Benutzerprogramm
}
```

1) InitTouchP0P1 ("0") ;

2) InitTouchP0P1 ("1rmp") ;



Der Touchscreen bleibt ausgeschaltet
P0, P1 bleibt an den Pins wirksam

Der Touchscreen wird aktiviert:

1..4: Gemeinsamer Schalter- Tastertyp

r: Rotiert horizontal, sonst vertikal

m: MSB oben/rechts, sonst unten/links

p: P0 aussen, sonst mittig

- Touchscreen Textfunktionen

a) vertikal, 20 Zeilen à 30 Zeichen

b) horizontal, 15 Zeilen à 40 Zeichen

```

0: Text-, Variablenausgaben
1: -----

Variablenwerte dezimal:
0, -444, 1234567890

Variablenwerte binär:
1 Bit: 0,
8 Bit: 1010'1010
16 Bit: 1111'1000'1111'1000

Variablenwerte hexadezimal:
16 Bit: 0x2AFB

Textfarbenwechsel:
32 Bit: 1111'1000'1111'1000
      1111'1000'1111'1000
18:
19:
  
```

```

0: Text-, Variablenausgaben
1: -----

Variablenwerte dezimal:
0, -444, 1234567890

Variablenwerte binär:
1, 8, 16 Bit

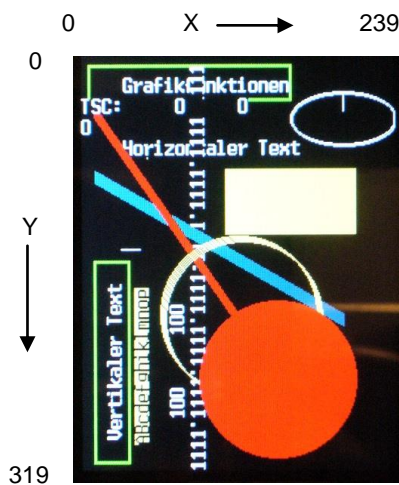
32-Bit:
1111'1000'1111'1000:1111'1000'1111'1000

13:
14:
  
```

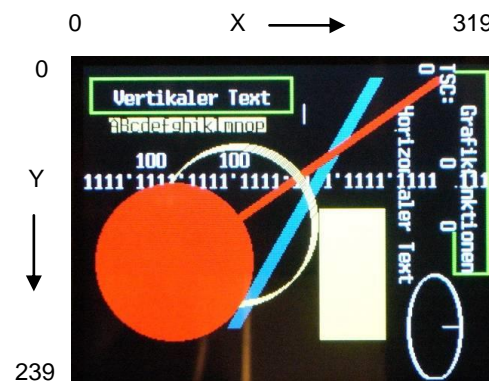
Syntax	Funktion	Beispiel
<code>void InitTouchScreen (void);</code>	Initialisiert den Touchscreen ohne P0P1 für Text, Grafik und Peripherie	<code>InitTouchScreen ();</code>
<code>void setScreenDir (char dir);</code>	Display Ausrichtung wechseln horizontal / vertikal	<code>setScreenDir (HOR);</code> <code>setScreenDir (VER);</code>
<code>void clearScreen (long color);</code>	Füllt den Touchscreen mit color	<code>clearScreen (BLACK);</code>
<code>void setTextcolor (long color);</code>	Farbwechsel für nachfolgenden Text	<code>setTextcolor (WHITE);</code>
<code>void print (char *txt);</code>	Schreibt Text hinter die letzte Position	<code>print ("Text");</code>
<code>void println (char *txt);</code>	Schreibt Zeile hinter die letzte Position und springt an den nächste Zeilenanfang	<code>println ("Text");</code> <code>println ("");</code>
<code>void printAt (char n, char *txt);</code>	Schreibt Text an den Anfang der Zeile mit Nummer n	<code>printAt (12, "Text");</code>
<code>void printBin (char n, long num);</code>	Konstanten- und Variablenwerte im Binär-code wie 1111'0000 mit der Bitanzahl n	<code>printBin (8, 250);</code> <code>printBin (32, variable);</code>
<code>void printHex (char n, long num);</code>	Konstanten- und Variablenwerte im Hex-code wie 0xFF00123E mit der Bitanzahl n	<code>printHex (8, 250);</code> <code>printHex (32, variable);</code>
<code>void printDec (char form long num);</code>	Ganzzahlige Werte aller Typen mit Feldlänge und Vorzeichen in form - vorgegebene Feldlänge für Typ unsigned - vorgegebene Feldlänge mit Vorzeichen - wertabhängige Feldlänge, Typ unsigned - wertabhängige Feldlänge mit Vorzeichen da zu kurze Feldlängen erweitert werden	<code>printDec (12, variable);</code> <code>printDec (-8, 123456);</code> <code>printDec (1, variable);</code> <code>printDec (-1, -123456);</code>

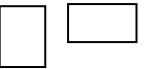







- Touchscreen Grafikfunktionen

a) vertikal 240x320 Pixel



b) horizontal 320x 240 Pixel



	Syntax	Funktion	Beispiel
	<code>void InitTouchScreen (void);</code>	Touchscreen ohne P0P1 für Text, Grafik, Peripherie	<code>InitTouchScreen () ;</code>
	<code>void clearScreen (long color);</code> <code>void setScreenDir (char dir);</code>	Löschen mit color Display horiz, vertikal	<code>clearScreen(BLACK);</code> <code>setScreenDir (VER);</code> <code>setScreenDir (HOR);</code>
Text	<code>void textxy (char x,char y, *txt, long forcol, long backcol);</code>	Schreibt an x y Text mit Vorder-, Hintergrundfarbe	<code>textxy (10, 20, "Text", RED, BLACK);</code>
	<code>void plotDot (char x,char y, long col);</code>	Setzt einen Farbpixel bei x, y mit der Farbe col	<code>plotDot (10, 20, CYAN);</code>
	<code>void line (char x1, char y1, char x1, char y1,</code>	Zieht eine Gerade von x1, y1 nach x2, y2 mit Dicke thick, Farbe col	<code>line (10, 20, 50, 60, 2, RED);</code>
	<code>void rectang (char x1, char y1, char x1, char y1,</code>	Leeres Rechteck von x1, y1 nach x2, y2 mit Dicke thick, Farbe col	<code>rectang (10, 20, 50, 60, 2, RED);</code>
	<code>void filledRect (char x1, char y1, char x1, char y1,</code>	Gefülltes Rechteck von x1, y1 nach x2, y2 mit Farbe col	<code>filledRect (10, 20, 50, 60, RED);</code>
	<code>void circle (char x, char y, char x1, char y1,</code>	Kreis mit Zentrum x, y, Radius r, Dicke thick, Farbe col fill=0: leer, fill=1: gefüllt	<code>circle (90, 70, 50, 2, RED, 0);</code>
	<code>void ellipse (char x, char y,, char x1, char y1,</code>	Ellipse mit Zentrum x, y, Radius rx und ry, Dicke thick, Farbe col fill=0: leer, fill=1: gefüllt	<code>ellipse (90, 70, 30, 20, 2, RED, 1);</code>
	<code>void loadPicture (x, y)</code> ---> In Bearbeitung <---	Bitmap laden	<code>loadPicture ()</code>

- Touchscreen Farben

Vordefinierte Farbkonstanten:

BLACK	WHITE		DARK_GRAY	LIGHT_GRAY
OLIVE	GREEN	BRIGHT_GREEN	DARK_GREEN	LIGHT_GREEN
BROWN	BLUE	BRIGHT_BLUE		LIGHT_BLUE
	YELLOW	BRIGHT_YELLOW	DARK_YELLOW	
	CYAN	BRIGHT_CYAN		LIGHT_CYAN
	RED	BRIGHT_RED		LIGHT_RED
	MAGENTA	BRIGHT_MAGENTA		LIGHT_MAGENTA

Der 16 Bit Farbcode hat 32 Rot-, 64 Grün- und 32 Blauanteile

in Bit: 5 Bit R, 6 Bit G, 5 Bit B

RRRR ' RGGG ' GGGB ' BBBB

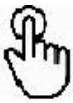



Mischbeispiel: long sattgrün = 63<<5;

0000 ' 0111 ' 1110 ' 0000

long hellgrün = 15<<11 + 63<<5 + 15;

0111 ' 1111 ' 1110 ' 1111

- Touchscreen Touchfunktionen

	Syntax	Funktion	Beispiel
	char getTSCtouched (void)	Touchscreenberührung: Rückgabe 0 / 1 0: unberührt 1: während Berührung	if (getTSCtouched ()) { }
	void getTSCxy (void)	Erfasst die x/y - Werte der berührten Position	getTCSxy () ;
 x	unsigned int getTSCx (void)	Gibt die x-Position der letzten Erfassung zurück	xPos = getTSCx () ;
 y	unsigned int getTSCy (void)	Gibt die y-Position der letzten Erfassung zurück	yPos = getTSCy () ;

- Ressourcennutzung der Libraries

TouchP0P1.lib mit TouchP0P1.h 8..17KByte	P0/P1 definiert an Ports	P0/P1 auf Screen	Touch, Grafik, Text, Periphere Funktionen	Sys- Timer belegt	SysTick_ Handler belegt
InitTouchP0P1 ("1..");	ja	ja	ja	1ms	aktiv
InitTouchP0P ("0");	ja	nein	ja	1ms	passiv

TouchScreen.lib mit TouchScreen.h InitTouchScreen(); 3..10KByte	nein	nein	ja	frei	frei
---	------	------	----	------	------

- Peripheriefunktionen

Einfache Nutzung der integrierten peripheren Funktionen des stm32f107 auf dem MCB32 ohne Registerkenntnisse.

Beachte:

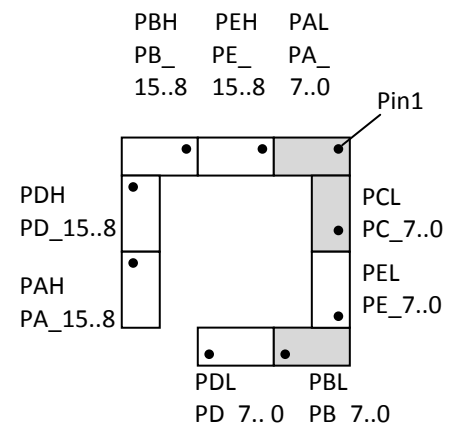
- In: floating 3.3V!
- Out: Open Drain wegen Kurzschlussgefahr!

Nicht 5V-tolerant!

a) 5x GPIO General Purpose Input Output

```
GPIOInit("PxH/L", "yyyyyyyy"); // x:A-E
GPIOPutByte("PxH/L", Byte); // y:0 In
char GPIOGetByte("PxH/L"); // 1 Out
// H/L High Low

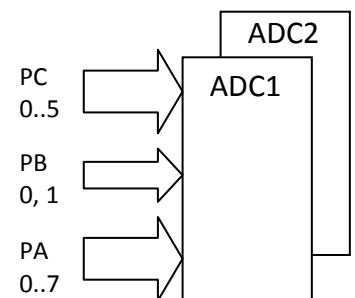
zB:
GPIOInit("PAL", "11110000"); // 4Out,4In
GPIOGetByte("PAL"); // PAL->var
GPIOPutByte("PEH", 0xAA); // 0xAA->PEH
```



b) 2x ADC Analog Digital Converter (0..3.3V)

```
ADCInit(1/2, "Pin");
char ADCGetVal(1/2);

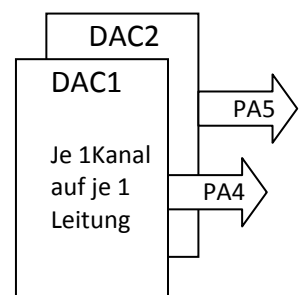
zB:
ADCInit(1, "PC4"); // ADC1 an PC4
var = ADCGetVal(1); // Analog von PC4
```



c) 2x DAC Digital Analog Converter (0..3.3V)

```
DACInit(1/2);
DACPutVal(1/2, 0..255);

zB:
DACInit(1); // DAC1 an PA4
DACPutVal(1, 100); // Analog Out PA4
```

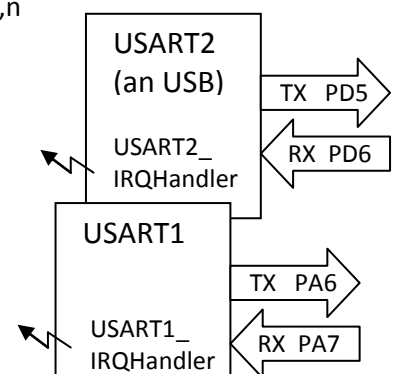


d) 2x USART Universal Synchronous Asynchronous Receiver Transmitter

USART2 liegt an USB zum PC; default: 9600Bd, 1,8,1,n

```
USARTInit (1/2, IRQPrio0..15);
USARTWrite (1/2, 'char');
char USARTRead (1/2); // ret c
char USARTToRead (1/2); // ret 0/1

zB:
USARTInit(2, 0); // USART2 ohne Intr.
USARTWrite(2, 'c'); // Sende Zeichen 'c'
c = USARTRead(2); // Warte auf Zeichen
c = USARTToRead(2); // 0/1 ob empfangen
```



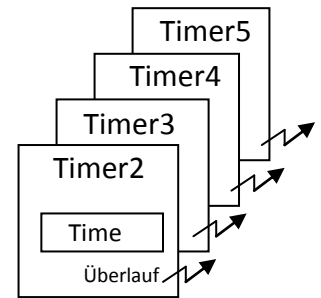
e) 4x General Purpose Timer (oder Counter)

```

TimerInit(2..5, n*100us, IRQPrio0..15);
int  TimerGetTime(2..5);    // 16 Bit
char TimerGetFlag(2..5);

zB:                                     // T2 auf 500ms
TimerInit (2, 5000, 0);    // ohne Intrpt
var = TimerGetTime(2);    // Zeit *100us
var = TimerGetFlag(2);    // 0/1 Überlauf

```



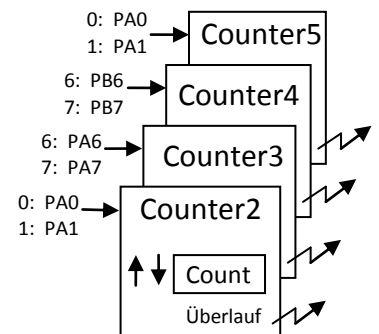
f) 4x General Purpose Counter (oder Timer)

```

CounterInit(2..5,"Pin",UD0/1,IRQPrio0..15);
void CounterPutCount(long l, 2..5);
int  CounterGetCount(2..5);    // 16 Bit

zB:                                     // Counter2<-PA1
CounterInit(2,"PA1",0,0); // Up ohne Intrpt
CounterPutCount(2, 0);    // Count = 0
var = CounterGetCount(2);  // return Count

```



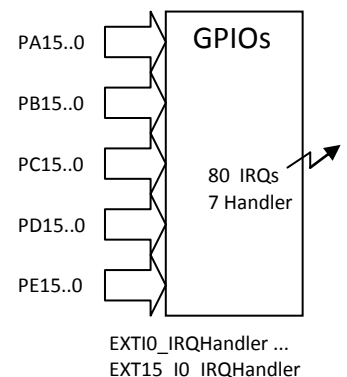
g) 7x Externe Interrupt Requests , vereinfacht jede PinNr nur 1x!

```

ExtIRQInit("Pin", Flanke0/1, IRQPrio0..15);

zB:                                     // IRQ PA0,pos Flanke
ExtIRQInit("PA0",0,4);    // Priorität 4

```



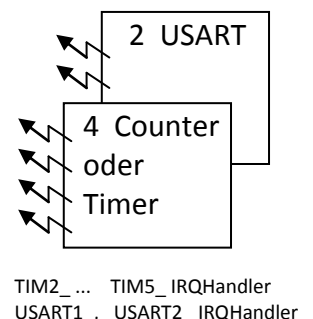
h) 6x Interne Interrupt Requests

```

TimerInit (... ,IRQPrio0..15); // Priorität
CounterInit(... ,IRQPrio0..15); // 0: OFF
USARTInit (... ,IRQPrio0..15); // 1: höchst
                                     // 15: tiefst

zB:
CounterInit(2,5,0,3);    // Priorität 3

```



i) Interrupthandler, die Bezeichner sind vorgeschrieben!

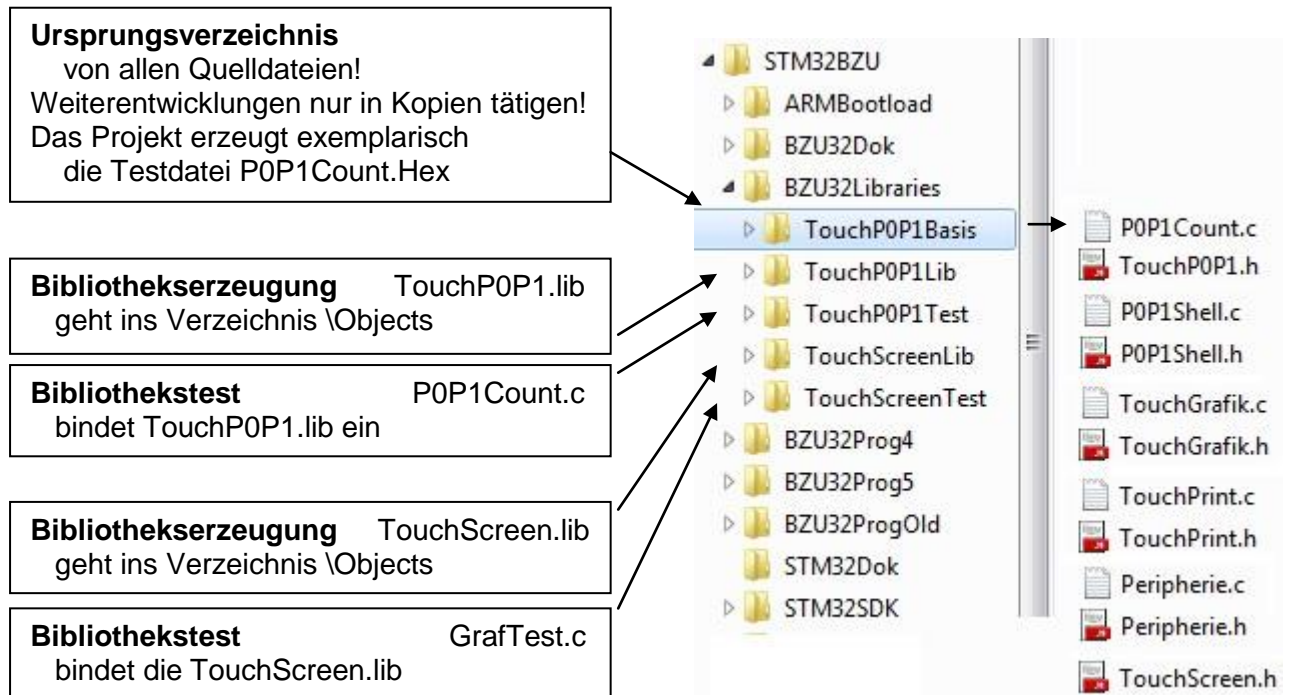
```

void TIM2_IRQHandler(void)
{
    // Immer zuerst:
    IRQClearFlag("T2");    // IRQClearFlag(..)
    // dann IntrService
}

```

Timer/Counter-IRQ:	TIM2_IRQHandler ... TIM5_IRQHandler	-> IRQClearFlag ("T2") ... ("T5")
USART-IRQ:	USART1_IRQHandler, USART2_IRQHandler	-> IRQClearFlag ("U1"), ("U2")
Ext. IRQ 0..4:	EXTIO_IRQHandler... EXT14_IRQHandler	-> IRQClearFlag ("PA0") ... ("PE4")
Ext. IRQ 5..9:	EXTI9_5_IRQHandler (gemeinsam)	-> IRQClearFlag ("PA5") ... ("PE9")
Ext. IRQ 10..15:	EXTI15_10_IRQHandler (gemeinsam)	-> IRQClearFlag ("PA10") ... ("PE15")

- Struktur der Bibliotheksdateien



P0P1Count.c:	Beispiel-Hauptprogramm zur Entwicklung und zu Tests von Bibliotheksfunktionen
TouchP0P1.h:	Sammlung aller Definitionen und Header zu P0P1Shell.c, TouchGrafik.c, TouchPrint.c, Peripherie.c damit die Lernenden zur Bibliotheksdatei P0P1Touch.lib nur die eine Headerdatei TouchP0P1.h einbinden müssen.
TouchScreen.h	Sammlung der Definitionen und Header ohne P0P1Shell mit TouchGrafik.c, TouchPrint.c, Peripherie.c zur Bibliotheksdatei TouchScreen.lib
P0P1Shell.c P0P1Shell.h	Definiert P0 und P1 als 8 Bit-Zugang zu den GPIOs und interagiert auf dem Touchscreen als Input und Output
TouchGrafik.c TouchGrafik.h	Enthält alle Grafik- und Touchfunktionen zur Nutzung des Touchscreens
TouchPrint.c TouchPrint.h	Enthält die Funktionen zur Textausgabe auf den Touchscreen
Peripherie.c Peripherie.h	Enthält Funktionen zur einfachen Nutzung der peripheren Funktionen ohne Kenntnisse der µC-Register oder von CMSIS

- Einsatz der Bibliotheken und deren Funktionen

