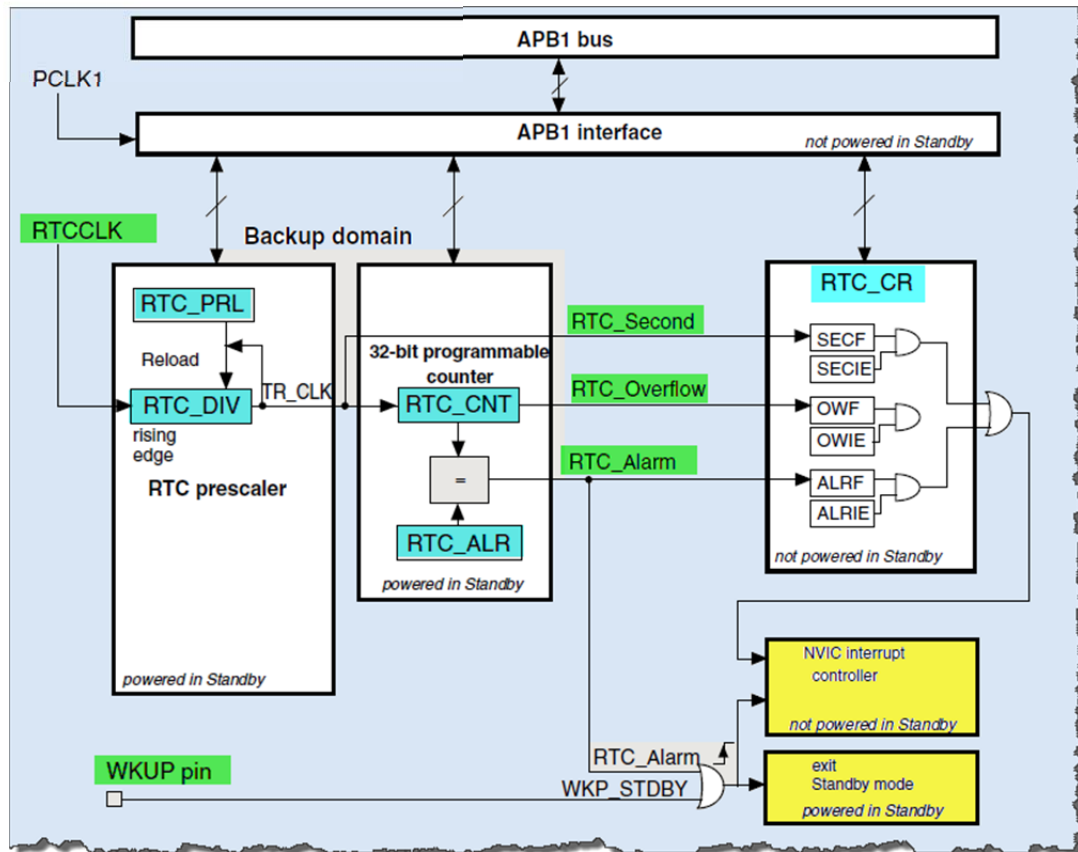


ENTWURF

Mikr

Mikrocontroller MCB32

Real Time Clock



MCB32 - Embedded Programmierung RTC

Version: 1745.001

Bitte beachten. Diese Unterlagen können ohne Vorankündigung jederzeit angepasst, verbessert und erweitert werden. Wir bitten Sie Wünsche und auch Fehler zu melden. (info@mcb32.ch)

Version C: Print mit **transparenter** Bodenplatte. Muss mit der LIB für Ver. C betrieben werden.

Version D: Print mit **grüner** Bodenplatte. Muss mit der LIB für Ver. **D** betrieben werden.

1 RTC und STM32F107xx

1.1 Einleitung

Der RTC (Real Time Clock Timer) ist ein unabhängiger Timer. Der RTC unterstützt ein Set von unabhängigen kontinuierlich laufenden Timern welche genutzt werden können. Z. Bsp für eine Kalender / Clock Funktion.

Die RTC Haupt (Core) und Clock-Konfigurations-Register befinden sich in der sogenannten Backup-Domain, was bedeutet, dass die RTC-Settings und auch die Zeitdaten nach einem Reset oder Wakeup aus dem Standby Mode, gesichert sind.

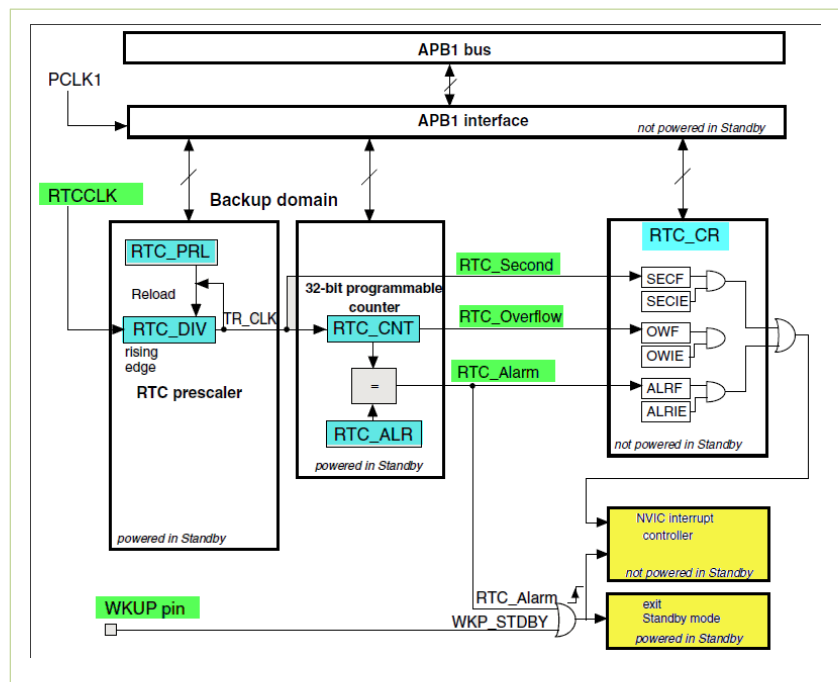
Nach einem Reset sind die Daten in den Backupregistern gegen ungesicherten Zugriff geschützt. Um einen Zugriff zu erhalten muss folgende Prozedur eingehalten werden:

- Durch setzen der PWREN und BKPEN-Bits für die Power- und Backup-Interface Clocks im RCC_APB1ENR Register. (siehe 18.1 in RM0008)
- Setzen des DBP Bit im Power Control Register (PWR_CR) ermöglicht dann den Zugriff zu den Backup- und den RTC-Register. [1, p. 473ff / Kapitel 18.]

1.2 Wichtige RTC Features

- Programmierbarer Teiler: bis 2^{20}
- Speisung via eigene Batterie wenn VDD wegfällt.
- 32-Bit Counter für Langzeitmessungen
- 2 separate Clocks: PCLK1 für das APB1 Interface und einen RTC Clock.
- 2 separate Reset Arten:
 - APB1 Interface wird vom Systemreset zurückgesetzt
 - Der RTC-Core (Teiler, Alarm, Counter, Teiler) wird nur von einem Backup-Domain Reset gesetzt. (siehe 7.1.3 Backup Domain Reset)
- 3 unabhängige Interrupt Leitungen:
 - Alarm Interrupt für SW Alarms
 - Sekunden Interrupt für periodische Interrupts
 - Overflow IR wenn der interne Counter 0 erreicht.

Das nebenstehende Blockdiagramm zeigt eine vereinfachte Darstellung. Der RTC besteht aus 2 Hauptfunktionen. Die Erste (APB1 Interface) verbindet den RTC mit dem APB Bus. Diese Einheit beinhaltet einige 16Bit Register welche vom APB1 Bus erreicht werden können. Die 2te Einheit (RTC Core) besteht aus einer Kette von programmierbaren Countern welche in 2 Blöcke unterteilt sind. Dem Prescaler Block welche den TR_CLK erzeugt. Der 2te Block ist ein 32Bit-Counter.



1.2.1 Reset der RTC Register [1, p. 473ff / Kapitel 18.]

Alle System Register werden nach einem Reset zurückgesetzt.

Die Register **RTC_PRL**, **RTC_ALR**, **RTC_CNT**, und **RTC_DIV** sind davon aber ausgenommen.

Die Register **RTC_PRL**, **RTC_ALR**, **RTC_CNT**, und **RTC_D** werden erst nach einem BackupDomain-Reset zurückgesetzt.

1.2.2 Lesen der RTC Register [1, p. 473ff / Kapitel 18.]

Der RTC Core ist komplett unabhängig vom RTC-APB1 Interface. Der Zugriff (via SW) auf den RTC Prescaler, Counter, Alarm geschieht via das APB1 Interface aber die betroffenen Backupregister Register werden intern bei jeder steigenden Flanke des RTC-Clock gesichert, dies gilt auch für die Flags. Daher muss bei einem ersten Lesen der RTC APB1 Register nach einem APB1-Interface Enable aufgepasst werden, da sie korrupt sind und es wird meistens nur 0 zurückgegeben. Das kann in folgenden Situationen passieren:

- System Reset
- Wakup aus Standby oder Stop Modus.
- Usw. Siehe dazu Ref Manual 18.3.3

1.2.3 Konfigurieren der RTC Register [1]

Um in die RTC_PRL, RTC_CNT, RTC_ALR Register zu schreiben, muss zuerst der Konfigurations-Modus erreicht werden. Das geschieht in dem das CNF Bit im RTC_CRL Register gesetzt wird. Zudem ist das Schreiben in ein RTC Register nur möglich, wenn vorgängige Schreiboperationen beendet sind.

Die SW kann dies durch Abfragen des RTOFF Status Bits im RTC_CR Register abgefangen werden. Wenn das Bit 1 ist kann ein neuer Wert geschrieben werden.

1.2.3.1 Konfiguration Prozedur:

1. Polle RTOFF, warte bis Bit 1 wird
2. Setze das CNF Bit um in den Konfigurationsmode zu gelangen.
3. Schreibe in eines oder mehrere RTC Register
4. Clear das CNF Bit um den Konfigurationsmode zu verlassen
5. Pole RTOFF, warte bis das Bit 1 wird um das Ende der Schreiboperation (in die Register) abzuwarten. Die Schreiboperation wird ausgeführt, wenn das CNF Bit zurückgesetzt wird, das dauert mindestens 3 RTCCLK Zyklen.

1.2.4 RTC Flags

- I. Das RTC Second Flag (SECF) ist vor jedem Update des RTC Counters aktiv.
- II. Das RTC Overflow Flag (OWF) ist während dem letzten Clock Zyklus des RTC Core Clock aktiv, bevor der Zähler wieder 0x0000 wird.
- III. Das RTC_Alarm und RTC Alarm Flag (ALRF) (siehe Bild 1: RTC) ist während dem letzten RTC Clock Zyklus aktiv bevor der Counter den Wert im Alarm Register (RTC_ALR+1) erreicht

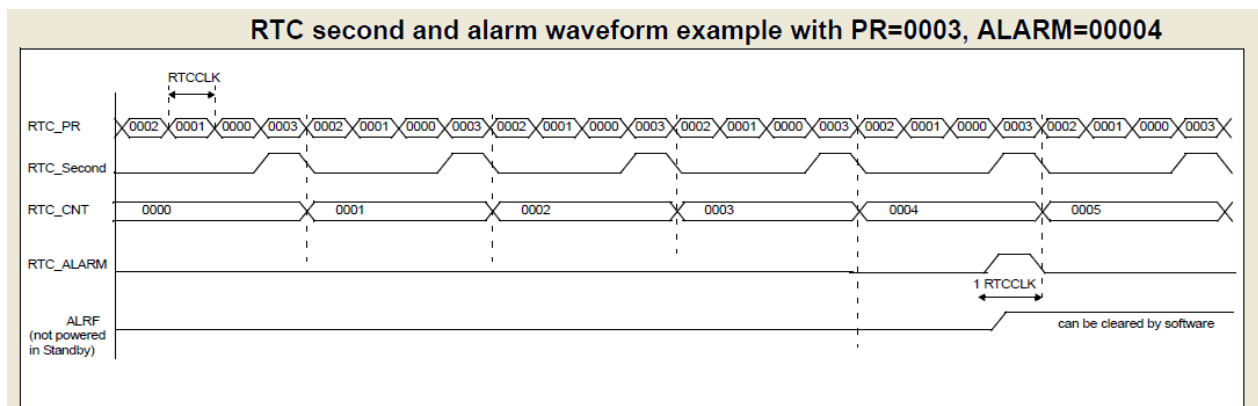


Bild 1: RTC

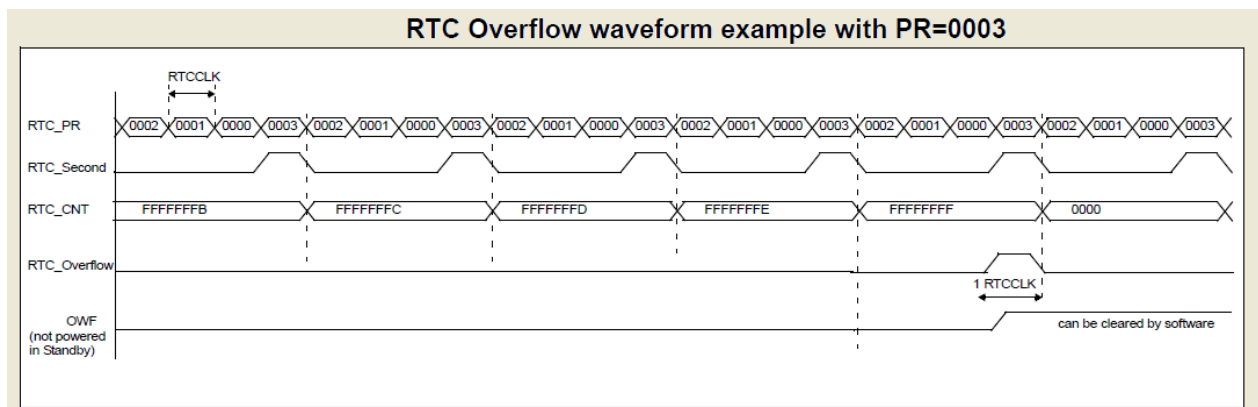


Bild 2: RTC Overflow

1.3 RTC Register

Die Register können via 16Bit (Half Word) oder 32Bit (Words) Zugriffen manipuliert werden.

1. RTC control register high (RTC_CRH)
2. RTC control register low (RTC_CRL)
3. RTC prescaler load register (RTC_PRLH / RTC_PRL)
4. RTC prescaler divider register (RTC_DIVH / RTC_DIVL)
5. RTC counter register (RTC_CNTH / RTC_CNTL)
6. RTC alarm register high (RTC_ALRH / RTC_ALRL)

1.3.1 RTC register map und Reset Werte

Offset	Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0x00	RTC_CRH	Reserved																												OWIE	ALRIE	SECF			
	Reset value																													0	0	0			
0x04	RTC_CRL	Reserved																												RTOFF	CNF	RSF	OWF	ALRF	SECF
	Reset value																													1	0	0	0	0	0
0x08	RTC_PRLH	Reserved																												PRL[19:16]					
	Reset value																													0	0	0	0		
0x0C	RTC_PRL	Reserved																PRL[15:0]																	
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x10	RTC_DIVH	Reserved																DIV[31:16]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x14	RTC_DIVL	Reserved																DIV[15:0]																	
	Reset value																	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x18	RTC_CNTH	Reserved																CNT[13:16]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x1C	RTC_CNTL	Reserved																CNT[15:0]																	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x20	RTC_ALRH	Reserved																ALR[31:16]																	
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
0x24	RTC_ALRL	Reserved																ALR[15:0]																	
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		

Bild 3: RTC Overflow

[1, p. p 191]

[1, p. p 210]

[1, p. p 211]

[1, p. p 211]

[1, p. p 212]

4 Anhang: Nested Vector Interrupt Controller NVIC

für insgesamt 58 Interrupts.

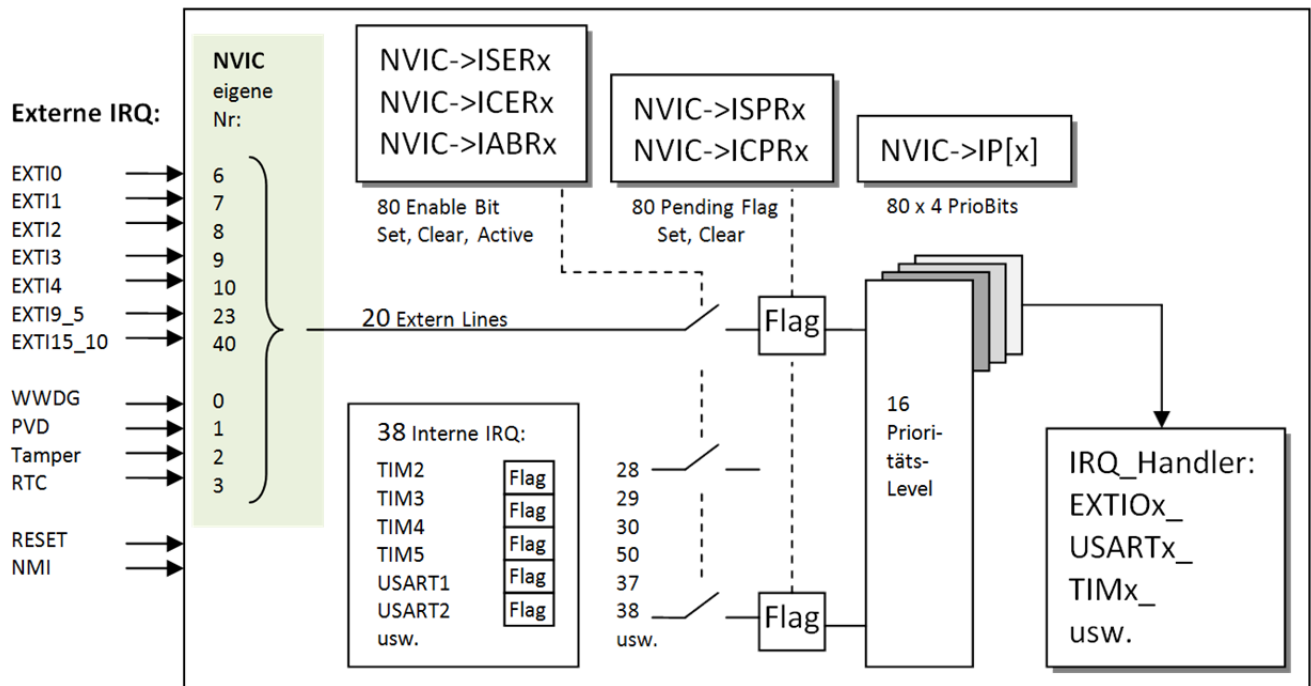


Abbildung 10: NVIC Blockdiagramm

Der NVIC→IP[x]-Block legt den Prioritäts-Level des Interrupts zwischen 0 und 15 fest. Interrupts mit einer niedrigeren Nummer besitzen eine höhere Priorität. Wird ein Interrupt mit einer höheren Priorität ausgelöst, während ein Interrupt einer niedrigeren Priorität abgearbeitet wird, so wird letzterer unterbrochen und die Abarbeitung des Interrupt Handlers des höher priorisierten Interrupts gestartet.

5 Anhang: Interrupt Vektorliste und Servicefunktionsaufrufe

NVIC NR	Priority	Type of priority	Acronym	Description	Address	Interrupt Handler Service Name
	-	-	-	Reserved	0x0000_0000	Beispiel Interrupthandler void EX- TIO_IRQHandler(void)
	-3	fixed	Reset	Reset	0x0000_0004	
	-2	fixed	NMI	Non maskable interrupt. The RCC Clock Security System (CSS) is linked to the NMI vector.	0x0000_0008	
	-1	fixed	HardFault	All class of fault	0x0000_000C	
	0	settable	MemManage	Memory management	0x0000_0010	
	1	settable	BusFault	Pre-fetch fault, memory access fault	0x0000_0014	
	2	settable	UsageFault	Undefined instruction or illegal state	0x0000_0018	
	-	-	-	Reserved	0x0000_001C - 0x0000_002B	
	3	settable	SVCall	System service call via SWI instruction	0x0000_002C	
	4	settable	Debug Monitor	Debug Monitor	0x0000_0030	
	-	-	-	Reserved	0x0000_0034	
	5	settable	PendSV	Pendable request for system service	0x0000_0038	
	6	settable	SysTick	System tick timer	0x0000_003C	
0	7	settable	WWDG	Window Watchdog interrupt	0x0000_0040	WWDG_IRQHandler
1	8	settable	PVD	PVD through EXTI Line detection interrupt	0x0000_0044	PVD_IRQHandler
2	9	settable	TAMPER	Tamper interrupt	0x0000_0048	TAMPER_IRQHandler
3	10	settable	RTC	RTC global interrupt	0x0000_004C	RTC_IRQHandler
4	11	settable	FLASH	Flash global interrupt	0x0000_0050	FLASH_IRQHandler
5	12	settable	RCC	RCC global interrupt	0x0000_0054	RCC_IRQHandler
6	13	settable	EXTI0	EXTI Line0 interrupt	0x0000_0058	EXTI0_IRQHandler
7	14	settable	EXTI1	EXTI Line1 interrupt	0x0000_005C	EXTI1_IRQHandler
8	15	settable	EXTI2	EXTI Line2 interrupt	0x0000_0060	EXTI2_IRQHandler
9	16	settable	EXTI3	EXTI Line3 interrupt	0x0000_0064	EXTI3_IRQHandler
10	17	settable	EXTI4	EXTI Line4 interrupt	0x0000_0068	EXTI4_IRQHandler
11	18	settable	DMA1_Channel1	DMA1 Channel1 global interrupt	0x0000_006C	DMA1_Channel1_IRQHandler
12	19	settable	DMA1_Channel2	DMA1 Channel2 global interrupt	0x0000_0070	DMA1_Channel2_IRQHandler
13	20	settable	DMA1_Channel3	DMA1 Channel3 global interrupt	0x0000_0074	DMA1_Channel3_IRQHandler
14	21	settable	DMA1_Channel4	DMA1 Channel4 global interrupt	0x0000_0078	DMA1_Channel4_IRQHandler
15	22	settable	DMA1_Channel5	DMA1 Channel5 global interrupt	0x0000_007C	DMA1_Channel5_IRQHandler
16	23	settable	DMA1_Channel6	DMA1 Channel6 global interrupt	0x0000_0080	DMA1_Channel6_IRQHandler

NVIC NR	Priority	Type of priority	Acronym	Description	Address	Interrupt Handler Service Name
17	24	settable	DMA1_Channel7	DMA1 Channel7 global interrupt	0x0000_0084	DMA1_Channel7_IRQHandler
18	25	settable	ADC1_2	ADC1 and ADC2 global interrupt	0x0000_0088	ADC1_2_IRQHandler
19	26	settable	CAN1_TX	CAN1 TX interrupts	0x0000_008C	CAN1_TX_IRQHandler
20	27	settable	CAN1_RX0	CAN1 RX0 interrupts	0x0000_0090	CAN1_RX0_IRQHandler
21	28	settable	CAN1_RX1	CAN1 RX1 interrupt	0x0000_0094	CAN1_RX1_IRQHandler
22	29	settable	CAN1_SCE	CAN1 SCE interrupt	0x0000_0098	CAN1_SCE_IRQHandler
23	30	settable	EXTI9_5	EXTI Line[9:5] interrupts	0x0000_009C	EXTI9_5_IRQHandler
24	31	settable	TIM1_BRK	TIM1 Break interrupt	0x0000_00A0	TIM1_BRK_IRQHandler
25	32	settable	TIM1_UP	TIM1 Update interrupt	0x0000_00A4	TIM1_UP_IRQHandler
26	33	settable	TIM1_TRG_COM	TIM1 Trigger and Commutation interrupts	0x0000_00A8	TIM1_TRG_COM_IRQHandler
27	34	settable	TIM1_CC	TIM1 Capture Compare interrupt	0x0000_00AC	TIM1_CC_IRQHandler
28	35	settable	TIM2	TIM2 global interrupt	0x0000_00B0	TIM2_IRQHandler
29	36	settable	TIM3	TIM3 global interrupt	0x0000_00B4	TIM3_IRQHandler
30	37	settable	TIM4	TIM4 global interrupt	0x0000_00B8	TIM4_IRQHandler
31	38	settable	I2C1_EV	I2C1 event interrupt	0x0000_00BC	I2C1_EV_IRQHandler
32	39	settable	I2C1_ER	I2C1 error interrupt	0x0000_00C0	I2C1_ER_IRQHandler
33	40	settable	I2C2_EV	I2C2 event interrupt	0x0000_00C4	I2C2_EV_IRQHandler
34	41	settable	I2C2_ER	I2C2 error interrupt	0x0000_00C8	I2C2_ER_IRQHandler
35	42	settable	SPI1	SPI1 global interrupt	0x0000_00CC	SPI1_IRQHandler
36	43	settable	SPI2	SPI2 global interrupt	0x0000_00D0	SPI2_IRQHandler
37	44	settable	USART1	USART1 global interrupt	0x0000_00D4	USART1_IRQHandler
38	45	settable	USART2	USART2 global interrupt	0x0000_00D8	USART2_IRQHandler
39	46	settable	USART3	USART3 global interrupt	0x0000_00DC	USART3_IRQHandler
40	47	settable	EXTI15_10	EXTI Line[15:10] interrupts	0x0000_00E0	EXTI15_10_IRQHandler
41	48	settable	RTCAlarm	RTC alarm through EXTI line interrupt	0x0000_00E4	RTCAlarm_IRQHandler
42	49	settable	OTG_FS_WKUP	USB On-The-Go FS Wakeup through EXTI line interrupt	0x0000_00E8	OTG_FS_WKUP_IRQHandler
-	-	-	-	Reserved	0x0000_00EC - 0x0000_0104	-_IRQHandler
50	57	settable	TIM5	TIM5 global interrupt	0x0000_0108	TIM5_IRQHandler
51	58	settable	SPI3	SPI3 global interrupt	0x0000_010C	SPI3_IRQHandler
52	59	settable	UART4	UART4 global interrupt	0x0000_0110	UART4_IRQHandler
53	60	settable	UART5	UART5 global interrupt	0x0000_0114	UART5_IRQHandler
54	61	settable	TIM6	TIM6 global interrupt	0x0000_0118	TIM6_IRQHandler
55	62	settable	TIM7	TIM7 global interrupt	0x0000_011C	TIM7_IRQHandler
56	63	settable	DMA2_Channel1	DMA2 Channel1 global interrupt	0x0000_0120	DMA2_Channel1_IRQHandler
57	64	settable	DMA2_Channel2	DMA2 Channel2 global interrupt	0x0000_0124	DMA2_Channel2_IRQHandler

NVIC NR	Priority	Type of priority	Acronym	Description	Address	Interrupt Handler Service Name
58	65	settable	DMA2_Channel3	DMA2 Channel3 global interrupt	0x0000_0128	DMA2_Channel3_IRQHandler
59	66	settable	DMA2_Channel4	DMA2 Channel4 global interrupt	0x0000_012C	DMA2_Channel4_IRQHandler
60	67	settable	DMA2_Channel5	DMA2 Channel5 global interrupt	0x0000_0130	DMA2_Channel5_IRQHandler
61	68	settable	ETH	Ethernet global interrupt	0x0000_0134	ETH_IRQHandler
62	69	settable	ETH_WKUP	Ethernet Wakeup through EXTI line interrupt	0x0000_0138	ETH_WKUP_IRQHandler
63	70	settable	CAN2_TX	CAN2 TX interrupts	0x0000_013C	CAN2_TX_IRQHandler
64	71	settable	CAN2_RX0	CAN2 RX0 interrupts	0x0000_0140	CAN2_RX0_IRQHandler
65	72	settable	CAN2_RX1	CAN2 RX1 interrupt	0x0000_0144	CAN2_RX1_IRQHandler
66	73	settable	CAN2_SCE	CAN2 SCE interrupt	0x0000_0148	CAN2_SCE_IRQHandler
67	74	settable	OTG_FS	USB On The Go FS global interrupt	0x0000_014C	OTG_FS_IRQHandler

6 Library TouchP0P1.Lib

Die bekannte Library umfasst neben den Befehlen für die Grafikansteuerung auch Befehle für das einfache Handling der Hardware (AD-DA Wandler usw.) sowie Befehle für das Handling der Interrupts.



6.1 Ressourcennutzung

Für den Einstieg in die Programmierung mit Hardware (1-4 Semester) wird der Touchscreen benutzt. Dort wird die Ein- und die Ausgabe auf die Ports P0/P1 grafisch dargestellt.

Die folgende Tabelle beschreibt den Init-Befehl für das Setup des Touchscreens.

Mit InitTouchP0P1 ("1") wird der Betrieb mit dem Port P0 und P1 aufgesetzt. Dabei läuft im Hintergrund ein 1ms Timer welcher das Handling der Ein- und Ausgaben übernimmt. Damit steht auch die Funktion *delay_ms(ms)* zur Verfügung. Sobald externe Schalter und LED angeschlossen werden, kann dieser Betrieb mit dem Befehl InitTouchPOP ("0") abgeschaltet werden.

Mit InitTouchP0P1 ("0"); ist der SysTick_Handler() nicht aktiv, daher keine Behandlung von *delay_ms(ms)*.

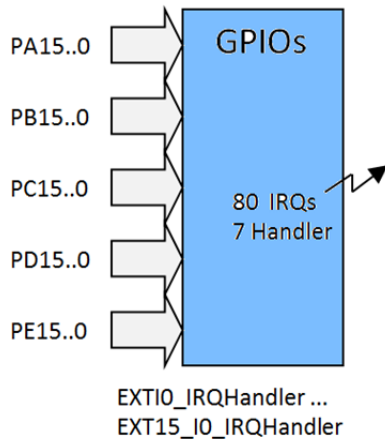
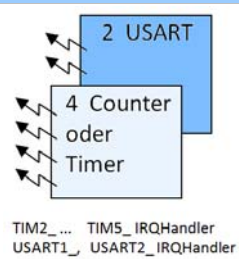
TouchPOP1. Lib					
TouchP0P1.lib 8..17KByte	P0/P1 definiert an Ports	P0/P1 auf Screen 	Touch, Grafik, Text, Periphere Funktionen 	Sys-Timer belegt	SysTick_Handler Ein/Aus
InitTouchP0P1 ("1.."); ¹	ja	ja	ja	1 ms	aktiv
InitTouchP0P1 ("0"); ²	ja	nein	ja	--	aus

Der SysTick_Handler() wird via einen Interrupt aufgerufen, wenn der System-Timer den Wert 0 erreicht.

¹ SysTick_Config (72* 1030); // SysTick_Handle() all 1000us/ (Initializes the System Timer and its interrupt, and starts the System Tick Timer.
Hinweis: Die Funktion SysTick_Config() erwartet als Parameter den Zählerwert für den SysTick-Timer. Mit dem Parameter SystemCoreClock/100 legen wir fest, dass nach xyz Systemtaktten der SysTick-Interrupt ausgelöst und der SysTick_Handler aufgerufen wird.

² SysTick_Handler() wird nicht gebraucht. *delay_ms(ms)* steht nicht zur Verfügung.

6.2 Interrupt Funktionen

		Version und Stand: 1745.001
4 x Externe Interrupt Requests (Vereinfacht, jede pinNr nur 1x)		
<div>ExtIRQInit ("Pin", Flanke0/1/2, IRQPrio0..15);</div>	<div>Pin[" "]: "PA0".. "PE15" Interrupt auslösender Pin</div> <div>Flanke: 0,1,2 Auslösende Flanke: 0 pos, 1 neg, 2 beide</div> <div>IRQPrio: 0, 1..15 Interruptpriorität 0 = AUS, 1 höchste, 15 tiefste.</div> <div>Px0..Px4 lösen 5 separate Interrupts aus. Px5..Px9 und Px10..Px15 je einen Interrupt. Das Programm springt in die zugehörigen 7 IRQ-Handler EXTI0_IRQHandler().. EXTI15_10_IRQHandler().</div> <div>// IRQ PA0,pos Flanke Priorität 4 ExtIRQInit("PA0",0,4);</div>	<div></div> <div>EXTI0_IRQHandler ... EXTI15_IO_IRQHandler</div>
6 x Interne Interrupt Requests		
<div>Priorität: 0= OFF, 1 = Höchste 15=Niedrigste</div> <div>TimerInit (2..5,n*100us,IRQPrio0..15);</div> <div>CounterInit (2..5,"Pin",UD0/1,IRQPrio0..15);</div> <div>USARTInit1/2, " IRQPrio0..15);</div>	<div>Siehe unter Timer</div> <div>Siehe unter Counter</div> <div>CounterInit(2,"PA0",0,3); // Prio 3</div> <div>Siehe unter USART</div>	<div></div> <div>TIM2_... TIM5_IRQHandler USART1_ USART2_IRQHandler</div>
Interrupt Handler (ACHTUNG: die Namen/Bezeichner sind vorgeschrieben)		
<div>// Beispiel void TIM2_IRQHandler(void) { IRQClearFlag("T2"); // Immer zuerst: // IRQClearFlag(..) // dann IntrService Programm }</div>	<div>Siehe auch folgende Tabelle für das Handling der ISR</div>	

Interrupt Quelle	Namen der Service-Routinen ISR	IRQClearFlag Bezeichner
Timer/Counter-IRQ:	TIM2_IRQHandler ... TIM5_IRQHandler	-> IRQClearFlag ("T2") ... ("T5")
USART-IRQ:	USART1_IRQHandler, USART2_IRQHandler	-> IRQClearFlag ("U1") , ("U2")
Ext. IRQ 0..4:	EXTI0_IRQHandler... EXTI4_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")

8 Anhang Anschlüsse am µC-Board MCB32

Entwicklungsanschlüsse

Fremdspeisung
genau 5V! oder ...

USB Speisung (**roter Stecker**)

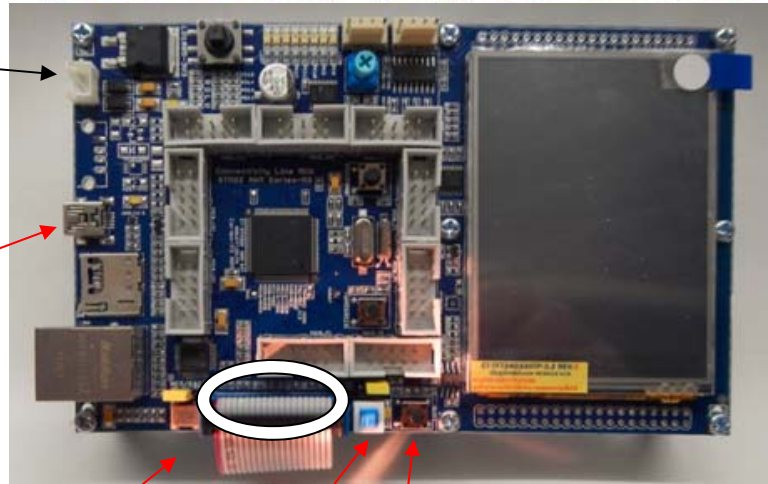
USB - ST-Link (**schwar-
zer USB Stecker**)
Zielsystemdebugging,
Boardspeisung
wie oben

Bootloadschalter:

Ungedrückt lassen **LED OFF**

Reset-Taster:

Programmrestart



Digitale Ein- und Ausgaben am µC-Board MCB32. **ACHTUNG** mit Potentiometer (Pot)

P1: 8x onboard LEDs
parallel zu 8x extern LEDs

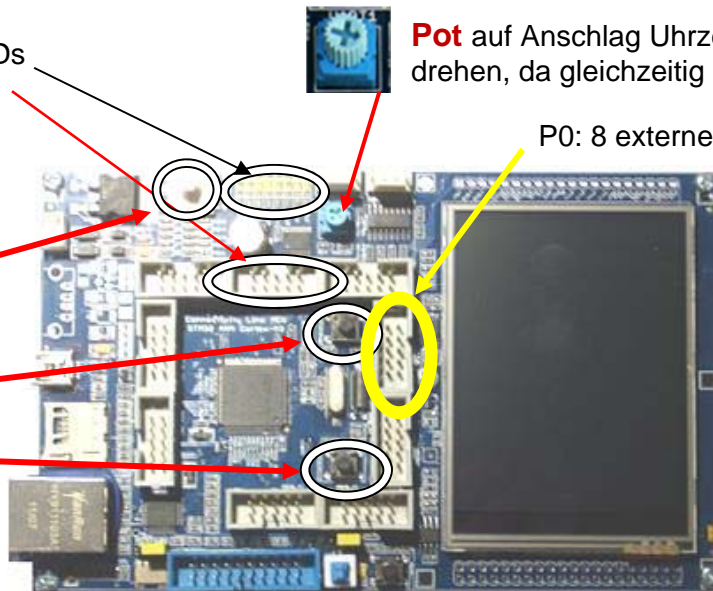
Pot auf Anschlag Uhrzeigersinn
drehen, da gleichzeitig P0_4

P0: 8 externe Schalter

ControlStick

Button 1

Button 0

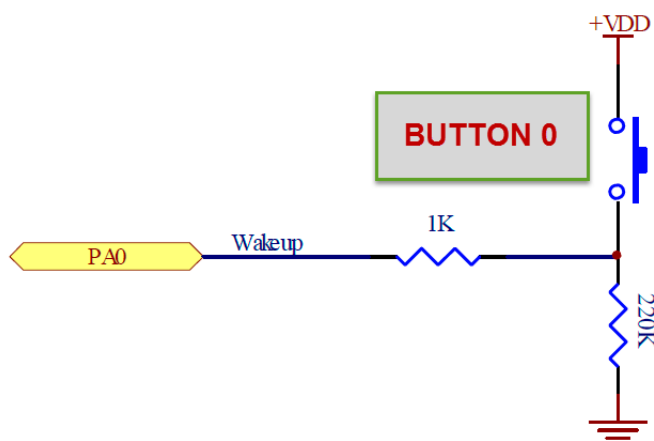


// In TouchP0P1.h definierte Pin-Bezeichnungen PA_0 .. PD_11, ohne Bezeichner wie Button .. !

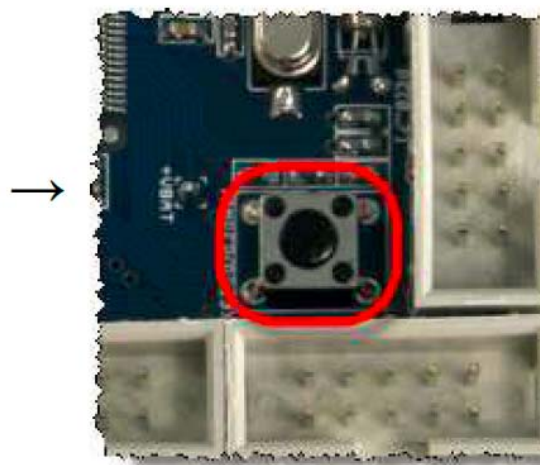
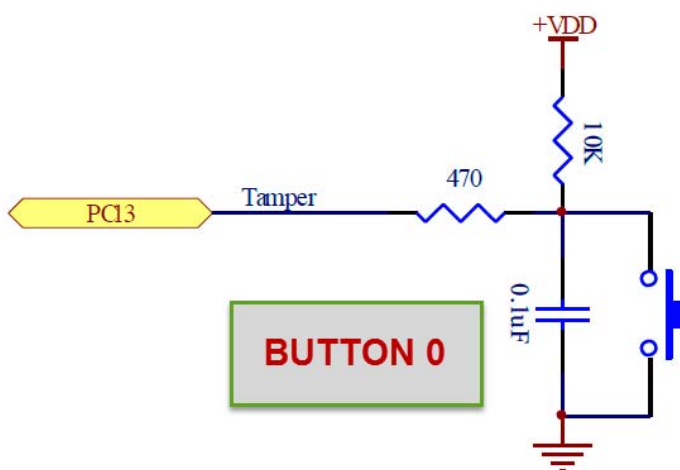
```
char Button0      = PA_0;           // Bitwert 1/0, aktiv low, prellt wenig
char Button1      = PC_13;

char Stick        = PD_High;        // als Byte 0xF8 open, aktiv low, alle entprellt
char StickSelect  = PD_15;          // Bitwert 1/0; Bytewert 0x80
char StickDown    = PD_14;          //           1/0;           0x40
char StickLeft    = PD_13;          //           1/0;           0x20
char StickUp      = PD_12;          //           1/0;           0x10
char StickRight   = PD_11;          //           1/0;           0x08
```


Button 0 (PA 0)

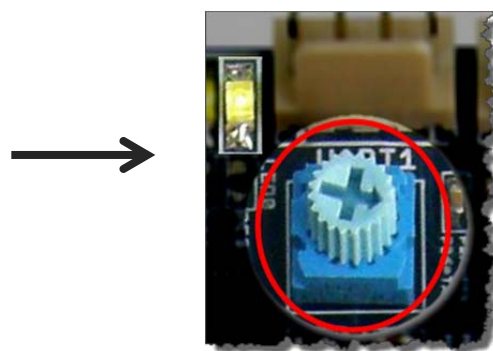
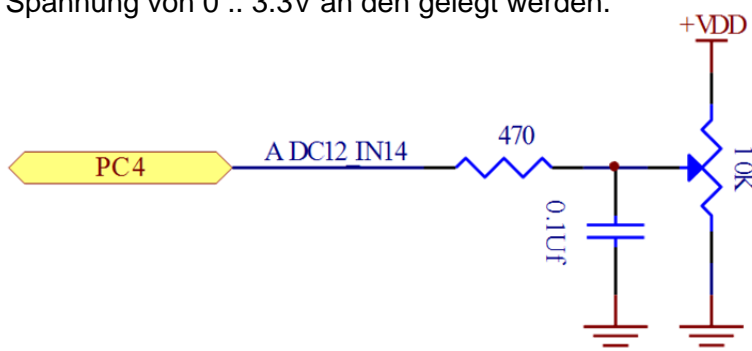


Button 1 (PC 13)



Potentiometer (PC 4) // resp. P0_4 (Library)

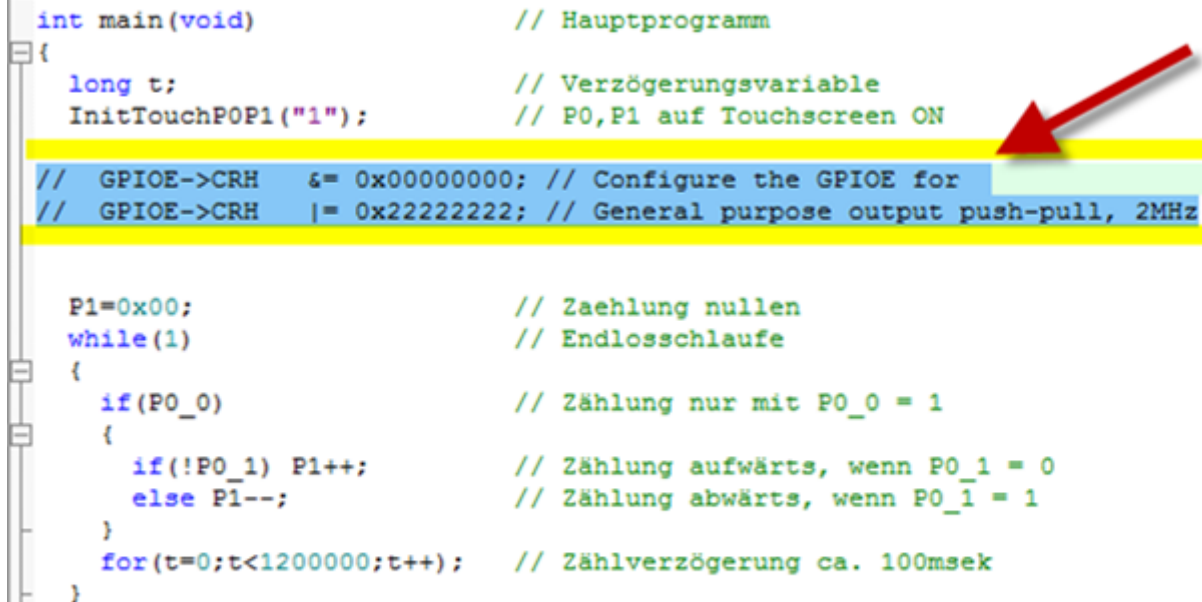
Wenn der Port PC4 als Analog-Input (AD-Wandler) geschaltet ist kann mit dem Potentiometer eine Spannung von 0 .. 3.3V an den gelegt werden.



LED auf Board aktivieren

Mit den folgenden Befehlen wird Port PE[8..15] so gesetzt dass die LEDs auf dem Board auch angesteuert werden. Siehe Beispiel Code weiter unten.

```
GPIOE->CRH   &= 0x00000000;    // Configure the GPIOE for
GPIOE->CRH   |= 0x22222222;    // General purpose output push-pull, 2MHz
```



```
int main(void)                // Hauptprogramm
{
    long t;                    // Verzögerungsvariable
    InitTouchPOP1("1");        // P0,P1 auf Touchscreen ON

    // GPIOE->CRH   &= 0x00000000; // Configure the GPIOE for
    // GPIOE->CRH   |= 0x22222222; // General purpose output push-pull, 2MHz

    P1=0x00;                    // Zaehlung nullen
    while(1)                    // Endlosschleife
    {
        if(P0_0)                // Zählung nur mit P0_0 = 1
        {
            if(!P0_1) P1++;      // Zählung aufwärts, wenn P0_1 = 0
            else P1--;           // Zählung abwärts, wenn P0_1 = 1
        }
        for(t=0;t<1200000;t++); // Zählverzögerung ca. 100msek
    }
}
```

9 Anhang: Referenzen

- [1] ST, «ARM_STM_Reference manual_V2014_REV15,» ST, 2014.
- [2] J. Yiu, The definitive Guide to ARM Cortex-M3 and M4 Processors, 3 Hrsg., Bd. 1, Elsevier, Hrsg., Oxford: Elsevier, 2014.
- [3] R. Jesse, Arm Cortex M3 Mikrocontroller. Einstieg und Praxis, 1 Hrsg., www.mitp.de, Hrsg., Heidelberg: Hütigh Jehle Rehm GmbH, 2014.
- [4] Diller, «System Timer,» 06 07 2014. [Online]. Available: http://www.diller-technologies.de/stm32.html#system_timer. [Zugriff am 06 07 2014].
- [5] A. Limited, «DDI0337E_cortex_m3_r1p1_trm.pdf,» ARM Limited, http://infocenter.arm.com/help/topic/com.arm.doc.ddi0337e/DDI0337E_cortex_m3_r1p1_trm.pdf, 2005, 2006 ARM Limited.
- [6] A. C. Group, «http://www.vr.ncue.edu.tw/esa/b1011/CMSIS_Core.htm,» 2007. [Online].
- [7] E. Malacarne, *Glossar Malacarne*, V11 Hrsg., Rüti: Cityline AG, 2014.
- [8] R. Weber, «General Purpos Input Output,» 2014.
- [9] R. Weber, «Projektvorlagen (div) MCB32,» 2013ff.
- [10] STM, «STM32F10x Standard Peripherals Firmware Library,» STM, 2010ff.
]
- [11] E. Schellenberg und E. Frei, «Programmieren im Fach HST,» TBZ, Zürich, 2010ff.
]