Datorteknik

Tomas Nordström

Föreläsning 8

Timers

För utveckling av verksamhet, produkter och livskvalitet.

HÖGSKOLAN HALMSTAD

Föreläsning 8

Timers

- Timerbegrepp
- Timer på SAM3U
- Avbrottskodexempel

- References:
- [SUM3U-complete] ATMEL AT91SAM ARM-based Flash MCU - SAM3U Series (Complete manual), 2012
- [Yiu-DefGuide] Joseph Yiu, “The definitive guide to the ARM cortex-M3”, second edition, Newnes, 2010

Timer

- Timers är normalt inte en del av processorarkitekturen, men finns nästan alltid med i en mikroprocessor.

Användning vid tidmätning:

- En räknare räknar fritt upp hela tiden. Genom externt avbrott kan man läsa av räknaren. Genom detta kan man mäta tiden mellan två händelser (avbrott), dvs mäta tid!

Utföra periodiska händelser:

- En räknare räknar fritt upp. Genom att jämföra räknarvärdet med ett förinställt värde och generera avbrott kan kod fås att utföras periodiskt.

- Mycket effektivare och enklare än att göra fördröjningar med loopar. Kan göra annat under tiden!

Timer funktioner i vår ARM Cortex M3 och ATMEL SAM3U

ARM Cortex M3 är lite speciell för processorarkitekturen definierar en speciell timer ***SYSTICK*** som alltid ska finnas.

"24-bit down counter; Self-reload capability"

ATMEL som skapat SAM3U, vår mikroprocessor, har dessutom lagt till

- Tre ***Timer/Counter block***.

  "3-Channel 16-bit Timer/Counter (TC) for capture, compare and PWM"

- En ***Real-time Timer***

  "32-bit Free-running back-up Counter; Integrates a 16-bit programmable prescaler running on slow clock; Alarm Register capable to generate a wake-up of the system"

- En ***Watchdog Timer***

  "16-bit key-protected once-only Programmable Counter; prevents the processor from being in a dead-lock on the watchdog access"

SysTick Timer

- The system timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter
- To set up the SysTick Timer, the recommended programming sequence is as follows:
  - Disable SysTick by writing 0 to the SYSTICK Control and Status register.
  - Write new reload value to the SYSTICK Reload Value register.
  - Write to the SYSTICK Current Value register to clear the current value to 0.
  - Write to the SYSTICK Control and Status register to start the SysTick timer.

SysTick example

```
; Setup SYSTICK exception handler (only needed if vector table is
located in RAM)
MOV R0, #0xF                ; Exception type 15
LDR R1, =SysTick_handler    ; address of exception handler
LDR R2, =0xE000ED08         ; Vector table offset register
LDR R2, [R2]
STR R1, [R2, R0, LSL #2]    ; Write vector to VectTblOffset+ExcpType*4
```

To generate SYSTICK exception every 1024 processor clock cycle:

```
; Enable SYSTICK timer operation and enable SYSTICK interrupt
LDR R0, =0xE000E010 ; SYSTICK control and status register
MOV R1, #0
STR R1, [R0]        ; Stop counter to prevent interrupt triggered accidentally
LDR R1, =1023       ; Trigger every 1024 cycles (since counter decrement from 1023 to
                    ; 0, total of 1024 cycles, reload value is set to 1023)
STR R1, [R0,#4]     ; Write reload value to reload register address
STR R1, [R0,#8]     ; Write any value to current value register to clear current
                    ; value to 0 and clear COUNTFLAG
MOV R1, #0x7        ; Clock source = core clock, Enable Interrupt, Enable SYSTICK
STR R1, [R0]        ; Start counter
```

Real-Time Timer (RTT)

- The Real-time Timer can be used to count elapsed seconds. It is built around a 32-bit counter fed by Slow Clock divided by a programmable 16-bit value. The value can be programmed in the field RTPRES of the Real-time Mode Register (RTT_MR).
- Programming RTPRES at 0x00008000 corresponds to feeding the real-time counter with a 1 Hz signal (if the Slow Clock is 32.768 kHz). The 32-bit counter can count up to 232 seconds, corresponding to more than 136 years, then roll over to 0.

- The RTT (and RTC) alarm *can generate a wake up of the core power supply*. This can be enabled by writing respectively, the bits RTCEN and RTTEN to 1 in the Supply Controller Wake Up Mode Register (SUPC_WUMR).

Timer/Counter – TC
SAM3U

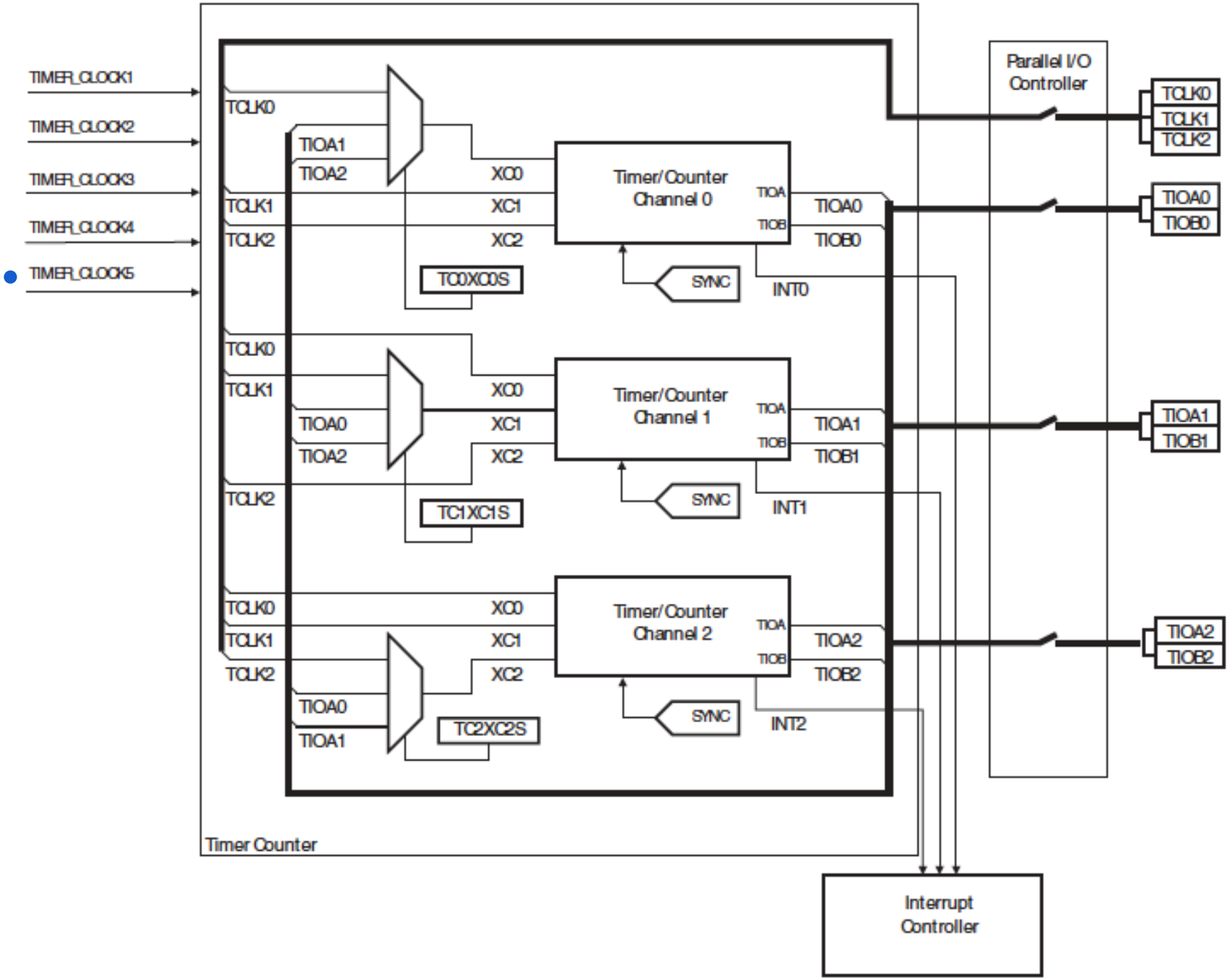
- Timer och Räknare (Timer/Counter - TC) på AT91SAM kan utföra flera funktioner, t.ex. frekvensmätningar, puls generation, skapa fördröjningar, räkna pulser, mäta intervall, pulsviddsmodulering (PWM), etc.

- Kapitel 36 i [SUM3U-complete].

Timer/Counter – TC
SAM3U

- Tre oberoende 16-bitars räknare (Channels)
- Stor flexibilitet i klockningen av dessa räknare
  - Tre externa klocksignaler (pinnar)
  - Fem internal klocksignaler (olika nedskalningar av MCLK)
  - Två "multi-purpose" I/O signaler kan också konfigureras av användaren.
- Två huvudanvändningssätt
  - Mäta yttre händelser (Capture mode)
  - Generera pulser eller vågformer (Waveform mode)
- Varje räknare styr var sitt avbrott

Timer Counter Block Diagram



Signaler till varje TC kanal

- Externa klocksignaler
- Externa I/O pinnar TIOA/B används som *insignal för Capture mode* och *utsignal för Waveform mode*

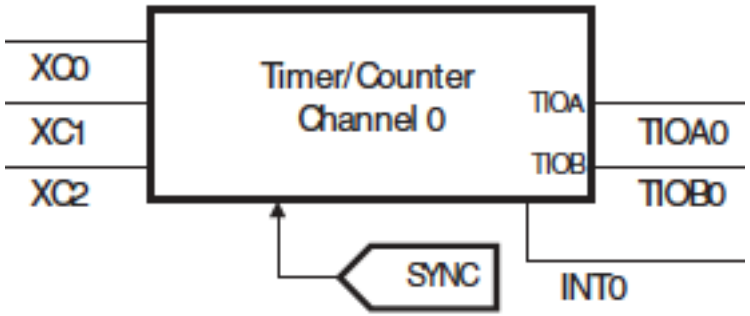


Table 36-2. Signal Name Description

Block/Channel	Signal Name	Description
Channel Signal	XC0, XC1, XC2	External Clock Inputs
	TIOA	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Output
	TIOB	Capture Mode: Timer Counter Input Waveform Mode: Timer Counter Input/Output
	INT	Interrupt Signal Output
	SYNC	Synchronization Input Signal

Externa signaler till TC

Table 36-3. TC pin list

Pin Name	Description	Type
TCLK0-TCLK2	External Clock Input	Input
TIOA0-TIOA2	I/O Line A	I/O
TIOB0-TIOB2	I/O Line B	I/O

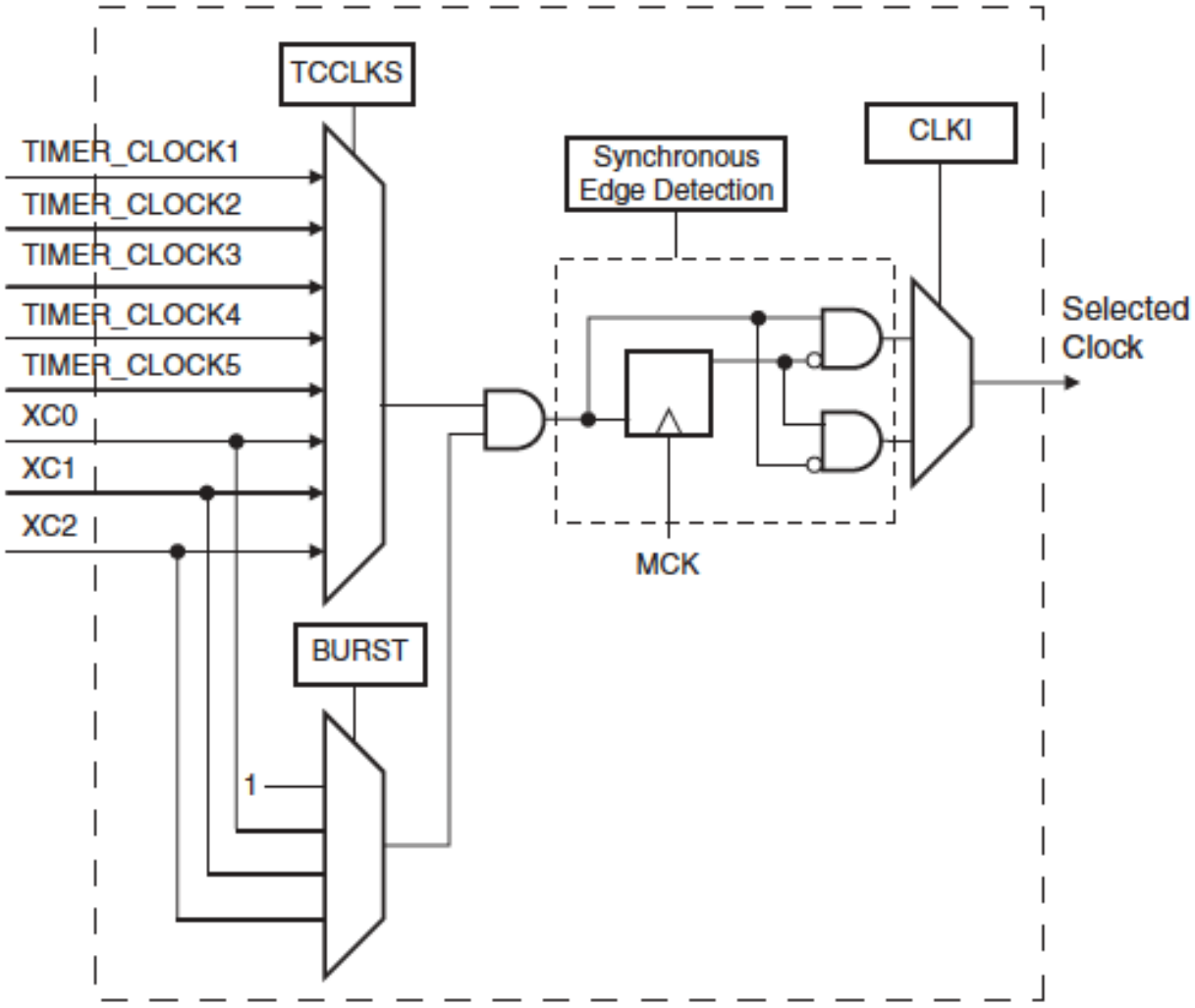
Table 36-4. I/O Lines

Instance	Signal	I/O Line	Peripheral
TC0	TCLK0	PA2	A
TC0	TCLK1	PB4	A
TC0	TCLK2	PA26	B
TC0	TIOA0	PA1	A
TC0	TIOA1	PB5	A
TC0	TIOA2	PA30	B
TC0	TIOB0	PA0	A
TC0	TIOB1	PB6	A
TC0	TIOB2	PA31	B

Borde nog vara TC1 och TC2 också

Klocksignaler till TC

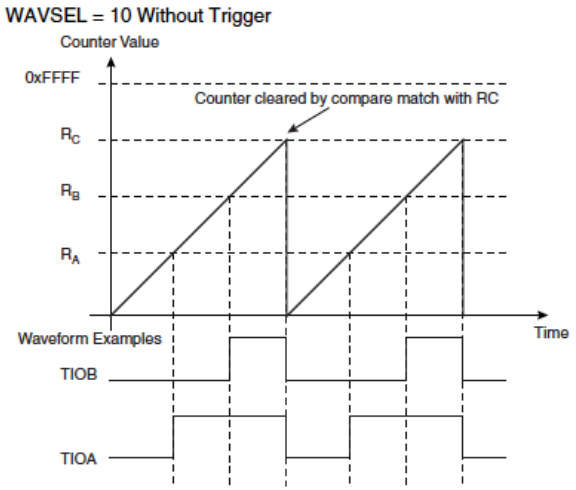
Clock Selection



Användning av TC

- Slå på huvudklockningen av TC (om den inte används så är den avstängd för att spara ström)
- Välj någon av räknarna (kanal/channel)
- Vilken funktion ska konfigureras?

  ska vi mäta/känna av yttre händelser, ska vi generera vågformer, eller bara periodiska signaler?
  - Bestäm capture eller waveform mode utifrån funktion
  - Sätt upp lämplig klockning och klockstyrning
  - Sätt upp lämplig triggfunktion (en triggning nollställer räknaren och startar den)
  - I waveform mode så sätts Register A,B,C till lämpliga värden för att skapa önskad vågform.
  - I Capture mode kan Register A och B laddas med räknarens värde när en viss programmerbar händelse sker på TIOA

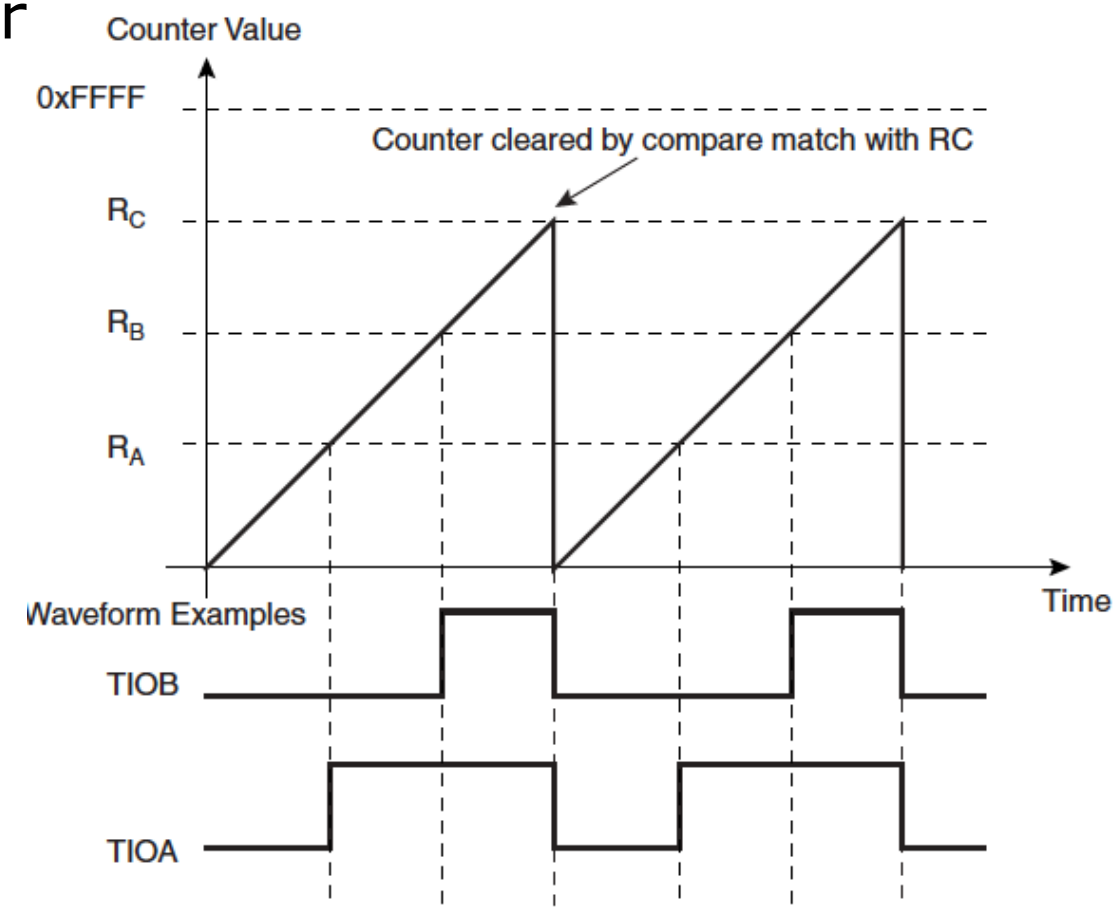


Hur kan en Timer/Counter användas?

- Periodiska avbrott
- Frekvensmätning,
- Puls generering,
- Skapa fördröjningar,
- Räkna pulser,
- Mäta intervall,
- Pulsviddsmodulering (PWM)

Pulsviddsmodulering

- Pulse-width modulation (PWM), or pulse-duration modulation (PDM)
- *”Duty-cycle”* - procent av tiden som är på jämfört med hela periodtiden



Användning av TC
Steg ett, slå på klockningen av TC

- För att minska strömförbrukningen, är de flesta periferienheterna normalt inte klockare. Genom att skriva ID för TC enheten i Peripheral Clock Enable Register (PCER) aktiveras dess klocka. Detta är det första steget vid initiering av en Timer Counter.
- För resten av konfigureringen bör sedan TC inaktiveras, tex om den har slagits på av en tidigare exekvering av programmet. Detta görs genom att sätta CLKDIS biten i den motsvarande Channel Control Register (CCR).

Användning av TC
Steg två, välj TC kanal och dess mode

- Välj vilken av de olika timer enheterna som ska användas (de kan eventuellt ha lite olika funktionalitet).
- Nästa steg består i att konfigurera Channel Mode Register (CMR). TC-kanaler kan arbeta i två olika lägen:
  - Capture-mode, som normalt används för att utföra mätningar på insignaler.
  - Waveform-mode, vilket möjliggör generering av pulser.

Användning av TC
Exempel: periodiska avbrott

- Är syftet med TC att generera en avbrottssignal i en jämn takt kan detta åstadkommas både i Capture och Waveform läget. (Eftersom ingen signal samplas eller genereras spelar läget ingen roll.) Men genom att sätta det i Waveform läget och skicka ut pulser på TIOA och TIOB kan felsökning underlättas.

- Waveform mode sätts genom att ettställa WAVE biten i TC_CMR (Channel Mode Register). Jmf. Kap 36.7.9.

Timer för att generera avbrott i jämn takt

- Ettställning av CPCTRG biten i TC_CMR återställer timern och startar om räknaren varje gång räknaren när värdet i TC_RC registret.
- Välj rätt TC_RC värde för att generera en specifik fördröjning. Man kan också välja mellan flera olika in-klockor för kanalen vilket möjliggör en förskalning av MCK. Eftersom timern har en upplösning på 16 bitar, så behövs en stor skalfaktor vid långa fördröjningar.
- Till exempel om en timmer måste generera en 500 ms fördröjning med en 48 MHz huvudklockfrekvens och RC är det antal klockcykler som genererar rätt fördröjning fås följande skalfaktorer

```
Clock = MCK/2       ; RC = 24000000 x 0.5 = 12000000
Clock = MCK/8       ; RC =  6000000 x 0.5 = 3000000
Clock = MCK/128     ; RC =   375000 x 0.5 = 187500
Clock = MCK/1024    ; RC =    46875 x 0.5 = 23437.5
Clock = 32kHz       ; RC =    32768 x 0.5 = 16384
```

Timer konfigurering

- Eftersom den maximala RC är 65535, framgår det av dessa resultat att användning av MCK delat med 1024 eller intern långsamma klockan är nödvändigt för att generera längre (ca 1 s) fördröjningar. I om vi tex ska skapa en 250 ms fördröjning behöver vi välja den långsammast möjliga inklocka i CMR.
- För att konfigurerar en 500 ms period genom att välja klockan MCLK/1024:

```
LDR R0,=TC0_BASE     ; 0x40080000
LDR R1,=0x0000C003   ; wave mode, wavesel=10b, timer_clock4
STR R1,[R0+#0x04]    ; TC0_CMR <- R1
LDR R1,=0x16E3       ; 12MHz/1024 * 0.5
STR R1,[R0+#0x1C]    ; TC0_RC <- R1
```

Enable TC interrupt

- Det sista initieringssteget består i att konfigurera avbrott när räknaren når programmerade värdet i TC_RC. För TC görs detta enklast genom att ställa in CPCS biten i Interrup Enable Register (TC_IER).

```
LDR R0,=TC0_BASE     ; 0x40080000
LDR R1,=0x10         ; CPCS bit to be enabled
STR R1,[R0+#0x24]    ; TC0_IER <- R1
```

Att bekräfta ett avbrott

- I vissa sammanhang måste man bekräfta ett avbrott I sin avbrottsrutin för att inte åka på ett avbrott direkt när man gör retur.

- I vår SAM3U manual är det inte speciellt tydligt om man måste göra det för TC avbrott eller inte.
- Jag har inte hunnit bekräfta det eller inte, men det kan vara så att man bör bekräfta ett TC-avbrott genom att göra en läsning av status registret TCx_SR (som då skulle släcka alla statusbitar)

```
LDR R0,=TC0_BASE     ; 0x40080000
LDR R1,[R0+#0x20]    ; dummy read of TC0_SR to acknowledge interrupt
```

Själva huvudprogrammet

- Mycket av det vi skrivit hittills är initieringen av kod och avbrott. Förutom avbrott ska ofta koden utföra något annat. Det som är själva huvudprogrammet
- I en del fall utför huvudprogrammet ingenting, utan man väntar endast på avbrott. Då bestå själva huvudprogrammet endast av en oändlig loop. Detta kallas ofta händelsestyrd kod.

```
MAIN B MAIN
```

Avbrott
Debugga

- Avbrottet går endast att prova genom att köra i fullfart. Går ej att stega!
- Smart att sätta brytpunkt på avbrottshanteraren. Genom detta kan man stega genom avbrottskoden då avbrottet väl kommit.

Fungerar det inte är följande fel vanliga:

- Glömt initiera avbrottsvektorn
- Glömt tillåta avbrott
- Fel i initiering av avbrottskällan

Sammanfattning

- Timer/Counter
- Basen är en räknare som kan
  - Klockas av olika klockor interna och externa
  - Kan jämföras med speciella register som genererar händelser, tex avbrott, när de är lika
  - Kan stoppas och startas av olika trigger händelser (interna och externa)
  - Beroende på vissa intervall (jmf med speciella register) kan pulsformer genereras på utgångar