Svar till Övning3 Datorteknik, HH vt12 – Avbrott och timers

Avbrott generellt

F7.1.
Avbrott sköts med hårdvara i processorn, man läser av hårdvarumässigt. Polling är en enklare princip för I/O-hantering. Bygger på att man läser av i en evig loop. Fördelen med avbrott är att processorn kan göra annat medan den väntar på externa signaler.

F7.2.
För att avbrott kan ske var som helst i koden. Sker det precis innan ett test på flaggorna skulle ju avbrottskoden kunna förändra flaggorna. Genom att spara undan även statusregister förhindras detta.

F7.3.
En delad resurs är något som används av både huvudprogram samt avbrottskod. Kan till exempel vara en fysisk minnesplats, ett register eller en I/O-port.

F7.4.
a) Då avbrottet kommer förändras R7. I huvudprogrammet utgår man från att R7 innehåller värdet som lästs ifrån porten. Då den inte gör det efter ett avbrott skett får man felaktigt beteende. Lösning: spara undan arbetsregister i avbrottskoden.

b) PORT0 delas av både avbrott och huvudprogram. Anta avbrottet kommer efter instruktionen `LDRb R7,[R1]`. Anta även för exemplets skull att inläst värde i R7 blev 00110110. Då avbrottet direkt efter så nollställs PORT0. Så långt fungerar allt, men det som händer då huvudprogrammet fortsätter är att R7 ökas med l till 00110111 samt att detta värde skrivs till PORTO. Den nollställning som varade fick inte vara längre än tiden att köra 2 instruktioner!! Lösningen är att slå av avbrott i huvudprogrammet under själva arbetandet med PORT0.

```
;Huvudprogram som räknar upp (8) lysdioder kopplade till PortB
            LDR   Rl,=PIOB_SODR
main        ; slå av avbrott (tex. PIO_IDR)
            LDRb  R7,[Rl,#8] ; PIOB_ODSR
            ADD   R7,R7, #l
            STRb  R7,[Rl]     ; PIOB_SODR
            EOR   R7,R7,#FF
            STRb  R7,[Rl,#4]  ; PIOB_CODR
            ; slå på avbrott (tex. PIO_IER)
            BL    delay
            B     main


; Avbrottskod som ska nollställa port
avbrott     PUSH {R7}
            LDR   R0,=PIOB_SODR
            MOV   R7,#0
            STRb  R7,[R0]     ; PIOB_SODR
            EOR   R7,R7,#FF
            STRb  R7,[R0,#4]  ; PIOB_CODR
            POP   {R7}
            BX    LR          ; återgång från avbrott.

;Huvudprogram som räknar upp (8) lysdioder kopplade till PortB
            LDR   Rl, =PIOB_ODSR
main        ; slå av avbrott
            LDRb  R7, [R1]
            ADD   R7, R7, #l
            STRb  R7,[R1]
            ; slå på avbrott
            BL    delay
            B     main

; Avbrottskod som ska nollställa port
avbrott     PUSH {R7}
            LDR   R0, =PIOB_ODSR
            MOV   R7, #0
            STRb  R7, [R0]
            POP   {R7}
            BX    LR         ; återgång från avbrott.
```

F7.5.
Ett maskbart avbrott kan man slå av och på.

F7.6.
De flesta externa avbrott är maskbara. Dock inte alla, tex reset och NMI. Man vill alltid kunna göra reset på processorn!

Interna avbrott genererade vid olika fel är normalt inte maskbara, tex MemManagement Fault, Bus Fault, och Usage Fault (illegal instruction). Dessa tre kan dock slås av via NVIC och dess ”System Handler Control and State Register” (0xE000ED24), vilken kanske kan ses som en maskning. Om dessa fel inträffar kommer ett HardFault att genereras.

F7.7.
Interna avbrott genereras internt i processorn. Ofta vid olika fel. MemManagement Fault, Bus Fault, och Usage Fault (illegal instruction)

Externa genereras av enheter utanför själva processorn. Tex PIO, Timer/Counter

F7.8.
Att man vid avbrott hoppar till en specifik adress beroende på avbrottskälla

F7.9.
Att man har en liten kod, sk kärna (eng: kernal), som körs på jämna mellanrum med hjälp av timeravbrott Det är denna kärna som växlar mellan dom andra programmen. Genom att växla fort fås illusionen att programmen exekverar samtidigt.

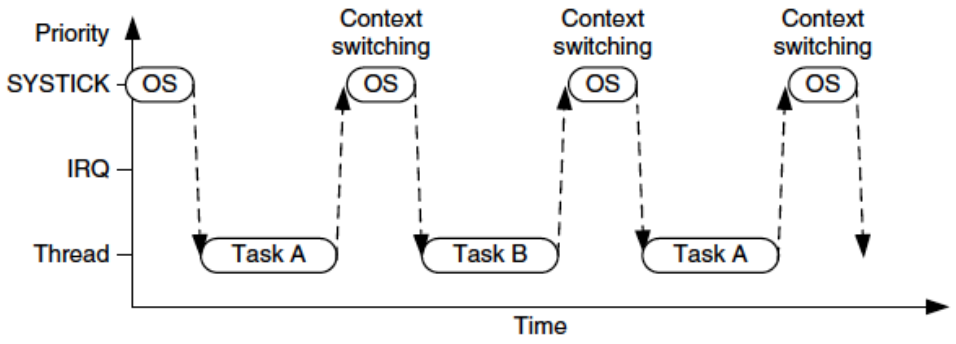


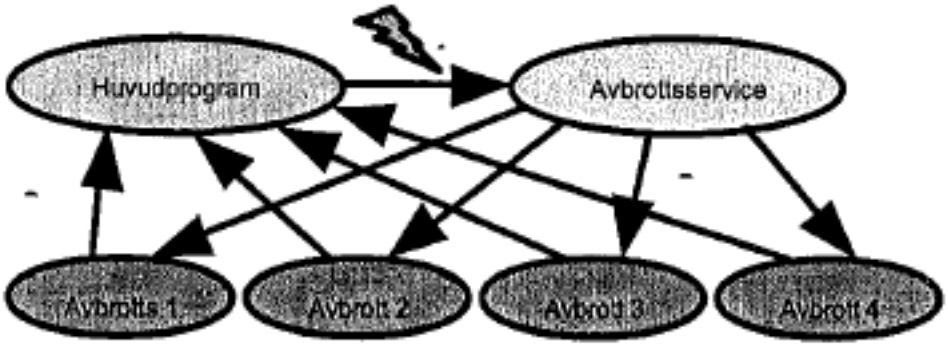
FIGURE 7.16
A Simple Scenario Using SYSTICK to Switch between Two Tasks.

F7.10.
Subrutinen anropas av genom kod med BL. Då vet man var anropet sker ifrån. Sparar PC för att hitta tillbaks

Avbrott anropas på externa signaler. Sparar PC, PSR (Statusregister), LR, samt R0-R3 och R12 för man inte vet när i koden avbrottet kommer att ske. Det är inte strikt nödvändigt att spara R0-R3 och R12, men då man nästan alltid behöver några register att arbeta med i avbrottsrutinen så är det bättre att undansparandet sker automatiskt.

F7.11.
Pågående instruktion körs klart. PC, PSR (Statusregister), LR, samt R0-R3 och R12 sparas på stacken (Main SP). Hoppar till förutbestämd adress beroende på avbrott som återfinns i avbrottsvektorn.

F7.12
Man kör i huvudprogrammet Då avbrottet kommer anropar man avbrottshanteraren och väljer därifrån vilket avbrott som skett. Sen anropas olika avbrottskoder beroende på avbrottskällan.



ARM-specifikt om avbrott

F7.21.
De stora huvudavbrotten är autovektoriserade. Sen är alla externa avbrott kopplade mot samma vektoradress. Därför måste senare läsa av vilket externt avbrott som skett. Detta görs i ett speciellt register PIO_ISR (Interrupt Status Register).

F7.22.
Då ett externt avbrott via PIO skett sätts en bit i respektive PIO_ISR (Interrupt Status Register). Man får i sin avbrottshanterare läsa avbrottsstatusregistret och därefter avgöra vad man ska göra. Man bör eventuellt även kvittera avbrottet genom att nollställa den bit som var satt efter man servat avbrottet.

F7.23.
Skriv kod for att initiera ett externt avbrott på PortA pinne 18. Använd register PIO_IER/PIO_IDR för att slå på avbrott för en speciell pinne i porten. Använd register PIO_REHLSR/PIO_FELLSR, PIO_LSR/PIO_ESR, samt PIO_AIMER/PIO_AIMDR för att ställa in flank eller nivå triggning för den pinnen. Anta man vill ha avbrott på positiv flank på signalen. Lägg även in ett hopp till avbrottshanteraren PIOA_IrqHandler på korrekt ställe i vektortabellen.

TBD

F7.24.

- Avbrotten styrs av: Interrupt Mask registers (PRIMASK, FAULTMASK, and BASEPRI)
- Vid avbrott ger „Interrupt Program Status register (IPSR)“ information om vilket avbrott som kommit.

F7.25

- Externt avbrott tillåts genom att ettställa motsvarande bit i SETENA ”Interrrupt set enable register” och slås av genom CLRENA ”Interrrupt clear enable register” i NVIC.

F7.26.

ARM-processorn arbetar i olika mode med olika rättigheter. Varje mode har egna uppsättningar av länkregister, stackpekare samt statusregister plus ibland vissa andra. Vid avbrott används de bankade registren för att "spara" statusregister och programräknare. Vid avbrott sparas aktuellt statusregister CPSR i ett speciellt register som heter SPSR (Saved Program Status Register) samt att PC sparas i "nya" modets LR.

Avbrott och timers

F7.31.

Skriv kod för att 100 ggr per sekund få ett avbrott ifrån systick. Använd registern Systick_CTRL, Systick_LOAD, Systick_VAL.

```
; Setup SYSTICK exception handler (Vector 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

; Enable SYSTICK timer operation and enable SYSTICK interrupt
LDR R0, =0xE000E010      ; SYSTICK_CTRL control and status register
MOV R1, #0
STR R1, [R0]        ; Stop counter to prevent spurious interrupt
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,
Systick_LOAD
STR R1, [R0,#8]     ; Write any value to current value register Systick_VAL
                    ; to clear current value to 0 and clear COUNTFLAG
MOV R1, #0x7        ; Clock src = MCK, Enable Interrupt, Enable SYSTICK
STR R1, [R0]        ; Start counter
```

Timer/Counters

F7.41.

Att det finns speciell hårdvara inbyggd i processorn som i princip bara är en vanlig binärräknare. Denna är färdig minnesmappad mot vissa adresser kan användas för att ta tid samt generera periodiska avbrott.

F7.42.

Vilka typer av timers finns i vår SAM3U?

Timer/Counter, SysTick, Real-Time Timer, Watchdog Timer

F7.43.

I vår SAM3U så finns en enhet som kallas Timer/Counter (TC). Ange alla funktioner som den kan användas till.

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

F7.44.

Då man använder en timer förekommer ett antal olika register. Ta reda på, ur datablad/manual, vilka register som styr användningen av TC kanal 0.

Table 36-5. Register Mapping

Offset[1]	Register	Name	Access	Reset
0x00 + channel * 0x40 + 0x00	Channel Control Register	TC_CCR	Write-only	–
0x00 + channel * 0x40 + 0x04	Channel Mode Register	TC_CMR	Read-write	0
0x00 + channel * 0x40 + 0x08	Reserved			
0x00 + channel * 0x40 + 0x0C	Reserved			
0x00 + channel * 0x40 + 0x10	Counter Value	TC_CV	Read-only	0
0x00 + channel * 0x40 + 0x14	Register A	TC_RA	Read-write[2]	0
0x00 + channel * 0x40 + 0x18	Register B	TC_RB	Read-write[2]	0
0x00 + channel * 0x40 + 0x1C	Register C	TC_RC	Read-write	0
0x00 + channel * 0x40 + 0x20	Status Register	TC_SR	Read-only	0
0x00 + channel * 0x40 + 0x24	Interrupt Enable Register	TC_IER	Write-only	–
0x00 + channel * 0x40 + 0x28	Interrupt Disable Register	TC_IDR	Write-only	–
0x00 + channel * 0x40 + 0x2C	Interrupt Mask Register	TC_IMR	Read-only	0
0xC0	Block Control Register	TC_BCR	Write-only	–
0xC4	Block Mode Register	TC_BMR	Read-write	0
0xC8	QDEC Interrupt Enable Register	TC_QIER	Write-only	–
0xCC	QDEC Interrupt Disable Register	TC_QIDR	Write-only	–
0xD0	QDEC Interrupt Mask Register	TC_QIMR	Read-only	0
0xD4	QDEC Interrupt Status Register	TC_QISR	Read-only	0
0xD8	Reserved			
0xE4	Reserved			
0xFC	Reserved	–	–	–

Notes: 1. Channel index ranges from 0 to 2.
2. Read-only if WAVE = 0

F7.45.
Skriv en avbrottshanterare som tar hand om capture/match avbrott från TC kanal 0. Anta avbrottskoden för TC kanal 0 heter TIMER. Anta dessa är korrekt initierade tidigare i koden.

Avbrottshanteraren ska:

- Läsa av vilket avbrott som skett samt anropa rätt avbrottskod
- Om det behövs, kvittera rätt avbrott
- Återgå från avbrottet

36.7.14 TC Status Register

Name: TC_SRx [x=0..2]

Address: 0x40080020 (0)[0], 0x40080060 (0)[1], 0x400800A0 (0)[2]

Access: Read-only

31	30	29	28	27	26	25	24
–	–	–	–	–	–	–	–
23	22	21	20	19	18	17	16
–	–	–	–	–	MTIOB	MTIOA	CLKSTA
15	14	13	12	11	10	9	8
–	–	–	–	–	–	–	–
7	6	5	4	3	2	1	0
ETRGS	LDRBS	LDRAS	CPCS	CPBS	CPAS	LOVRS	COVFS

```
TIMER       LDR R0, = 0x40080020
            LDR R1,[R0]
            ; Check for CPCS
            ANDS R1,R1,#0x10
            BNE TIMERRET
            ; Do something with this RC Compare Status
TIMERRET    BX LR
```

F7.47.
En timer kan användas för olika funktioner. Förklara kort principen för hur en timer används för att utföra följande:

1. Avbrott med jämna intervall
2. Mätning av tid på någon extern händelse

F7.48.
Konfigurera Timer (TC) kanal 0, så att avbrott genereras 2 ggr per sekund. Anta klockan till timern går med 10.5MHz.
Tips: Pre-scaler registret skalar ner (dividerar ner) klockan till timern

```
LDR R0,=TC0_BASE     ; 0x40080000
LDR R1,=0x0000C003   ; wave mode, wavesel=10b, timer_clock4
STR R1,[R0,#0x04]    ; TC0_CMR <- R1
LDR R1,=5127         ; 10.5MHz/1024 * 0.5
STR R1,[R0,#0x1C]    ; TC0_RC <- R1
```