# IS1200-Datorteknik-Labb3

## Fråga 1:

* Test pressing BTN3 and BTN2 at the same time. What happens? Why?

Båda siffra nr: 3 och nr: 2 blir uppdaterad på skärmen eftersom. Detta händer eftersom vårt program exekverar alla if-satserna. Vi har ”bruteforce statements” vilket säger att om vi tex trycker ned knapp 2 och 3 (011) ska två av våra if-satser ha den jämförelse och exekveras.

* Three device-registers for input/output control are TRISE, TRISESET, and TRISECLR.

Their functions are related. How? What are the differences?

TRISE: Avgör om ett stift ska vara INPUT (1) eller OUTPUT (0).

TRISESET: Sätter alla mappade stift till INPUTS (1)

TRISECLR: Sätter alla mappade stift OUTPUTS (0)

* In the generated assembly code, in which MIPS register will the return values from functions getbtns and getsw be placed in. You should be able to answer this question without debugging the generated assembly code.

Retur-värdena från funktionerna getbtns och getsw kommer att placeras i $v0 och $v1. Då det är standard returneringsregister för funktioner.

* In this exercise, we explained which bits that should be used in Port D and Port E. How can you find this information in the PIC32 and ChipKIT manuals? Be prepared to demonstrate how to find this information in the manuals.  
  *Advice:* check the lecture slides from lecture 5 for ideas.

PORTD: Table 4-25 till 4-28 från (0-15 & 31-16) PIC32 Family Data Sheet

PORTE: SW2: (pin 7), SW3 (pin 8), SW4 (pin 35) PIC32 Family Data Sheet

A picture containing text, indoor, display

Description automatically generated

## Fråga 2:

* When the time-out event-flag is a "1", how does your code reset it to "0"?

Koden återställer event-flag till 0 genom att genom att använda CLR register för ISF0 och rensa bort den 8:de biten.

IFSCLR(0) = 0x0100;

* What would happen if the time-out event-flag was not reset to "0" by your code? Why?

Om time-out-event-flag inte skulle återställas till 0 skulle timern på displayen börja räkna riktigt fort eftersom hela delay funktionen skulle ignoreras. Programmet skulle aldrig kolla efter time-outs, Timer registret skulle aldrig räkna upp till sin period och därmed skulle vi inte få önskvärd delay.

* Which device-register (or registers) must be written to define the time between time-out events? Describe the function of that register (or of those registers).

PR är periodens register. När timern når den specifika perioden återgår den till 0 och sätter TxIF biten in i IFS0 interrupt flagregistret.

PRx register kan man hitta i p14 PIC32 Family Reference Manual Section 14 Timers.

PRx register är period-register för timern, Den innehåller värden som är längden av en period.

* If you press BTN3 quickly, does the time update reliably? Why, or why not? If not, would that be easy to change? If so, how?

/\*Den uppdaterar klockan korrekt men man behöver inte hålla in knappen eftersom den sparar värde direkt när man klickar på knappen istället för att det värdet som man klickar på ska sparas och uppdateras bara om man håller in knappen när interruptus sker som i time4io.\*/

Klockfrekvensen går 10ggr / sekund

Om man skulle klicka snabbare än 10ggr / sekund skulle det bli fel

Den uppdaterar alltså tiden korrekt.

80MHz

Interupts 256 (gör frekvensen långsammare)

1000ms/10 = 100ms = 1s

Diagram, schematic

Description automatically generated

## Fråga 3:

* When the time-out event-flag is a "1", how does your code reset it to "0"?

Koden återställer time-out-event-flag till 0 genom att exekvera: IFSCLR(0) = 0x100

* What would happen if the time-out event-flag was not reset to "0" by your code? Why?

Om time-out-event-flag inte skulle återställas till 0 skulle timern på displayen börja räkna riktigt fort eftersom hela delay funktionen skulle ignoreras. Programmet skulle aldrig kolla efter time-outs, Timer registret skulle aldrig räkna upp till sin period och därmed skulle vi inte få önskvärd delay.

* From which part of the code is the function user\_isr called? Why is it called from there?

Funktionen user\_isr anropas från Vectors.S på rad 209 (jalu ser\_isr; nop). Derför att det är en vektor. Den anropas där ifrån för att pågrund av non och preserved register (mer förklarat på nästa fråga).

* Why are registers saved before the call to user\_isr? Why are only some registers saved?

Registerna som sparas innan anropet till user\_isr av typen non-preserved type. Eftersom vi har en funktion \_isr\_trampoline (caller ($t0)) som anropar på user\_isr (calle ($s0)), är det callers ($t0) uppgift att spara non-preserved register så att det kan ändvändas efter funktionens slut.

Vi sparar alltså Caller register ($t0,1,2,3..)

Register $1 - $15, $24 och $25 kan ändras av vilken C-funktion eller assembler subroutine.

Register $ra kan bara ändras genom att anropet till C-funktionen i.e.

Jal instruktionerna är viktiga att spara innan C-funktionen user\_isr anropas.

* Which device-register (or registers), and which processor-register (or registers) must be written to enable interrupts from the timer? Describe the functions of the relevant registers.

IECO – interrupt register med interrupt enable kontroll, bit 8 är T2IE.

IPC2 – prioritet. //(vi satte den till 7)