1. (3 punti) Due UART che comunicano

   Segna tutte le risposte corrette:

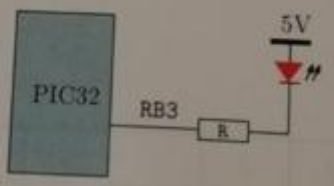
   - ○ si sincronizzano ad ogni carattere, per la durata della trama di trasmissione del carattere
   - ○ per sincronizzarsi usano il bit di start di ogni carattere, la velocità di comunicazione convenuta e il formato convenuto della trama
   - ○ usano una linea di clock per ogni direzione di comunicazione
   - ⊗ non si sincronizzano mai, come dice il loro nome "Universal **Asynchronous**..."

2. (3 punti) In un timer con prescaler = 2

   Segna tutte le risposte corrette:

   - ⊗ il *timer clock rate* è PBCLK× 2 (con PBCLK Peripheral Bus Clock)
   - ○ il *timer clock rate* è PBCLK÷ 2 (con PBCLK Peripheral Bus Clock)
   - ○ il registro contatore TMR viene incrementato ogni due fianchi attivi del clock del timer
   - ○ il registro contatore TMR viene incrementato ad ogni evento del *timer interrupt*

3. (3 punti) In un PIC32 il pin RB3 è collegato come in figura. Quali delle seguenti impostazioni permettono l'accensione del LED?



Segna tutte le risposte corrette:

- ○ `ANSELBbits.ANSB3 = 1; TRISBbits.TRISB3 = 0; LATBbits.LATB3 = 1;`
- ○ `ANSELBbits.ANSB3 = 1; TRISBbits.TRISB3 = 1; LATBbits.LATB3 = 0;`
- ○ `ANSELBbits.ANSB3 = 0; TRISBbits.TRISB3 = 1; LATBbits.LATB3 = 1;`
- ⊗ `ANSELBbits.ANSB3 = 0; TRISBbits.TRISB3 = 0; LATBbits.LATB3 = 0;`

4. (6 punti) Si supponga che un timer lavori con i seguenti parametri:
   - Peripheral Bus Clock = 20MHz
   - Period Register = 20000
   - Prescaler = 8

   Prima di abilitare il timer, il registro di conteggio TMR viene settato a 15000.
   Dopo quanto tempo, dall'abilitazione del timer, arriverà il primo *Event Flag*?

20000 - 15000 = 5000

20000 - 15000 = 5000

Tempo primo event flag $= 20 \cdot 10^6 \cdot 8 \cdot 5000 = 200\,000 \cdot 10^6 = 300$ [

Tempo primo event flag $= \frac{1}{20 \cdot 10^6} \cdot 8 \cdot 5000 = \frac{2000}{10^6} = 0{,}002$ [s]

$= 2$ [ms] ✓

5. (6 punti) Elencare in quattro punti che cosa fa la CPU in seguito ad un Interrupt ReQuest (IRQ, evento di interruzione).

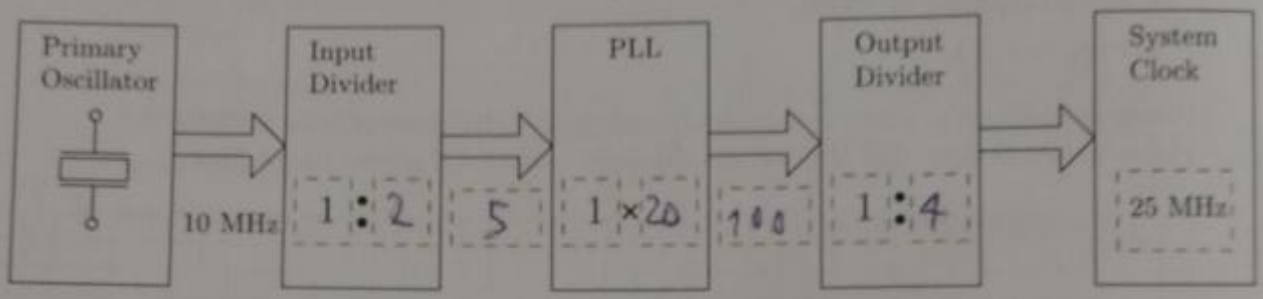
✓ - salva le informazioni necessarie a riavviare il l'operazione

✓ - legge "Interrupt vector table" per trovare la funzione da seguire

✓ - esegue la funzione di interrupt

✓ - riavvia l'operazione

6. (6 punti) Il PIC32 è provvisto di una parte hardware per la generazione del clock di sistema. In esso troviamo un circuito chiamato Phase Locked Loop (PLL).



a) A cosa serve il **blocco PLL**?

b) Su quali sorgenti di clock agisce?

c) Completa lo schema in modo da generare un SYSCLK di 25MHz e motiva.

Usa lo spazio quadrettato per i calcoli e rispondere ai punti.
Aiutatli col diagramma a blocchi che trovi di seguito.

b) Primary Oscillator, FRC Oscillator

a) Serve a alterare la frequenza dei 2 oscillatori nominati al punto b), e di conseguenza consente di variare il SYSCLK.

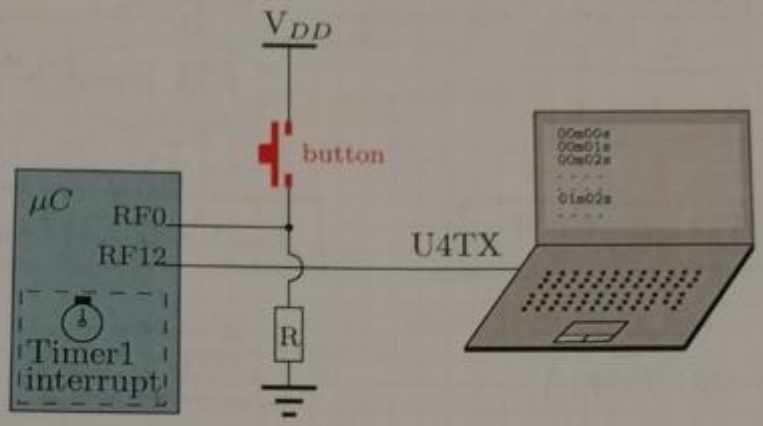
c) $10 \cdot \frac{1}{2} \cdot 20 \cdot \frac{1}{4} = 100 \cdot \frac{1}{4} = 25$

7. (8 punti) Viene fornito il seguente codice. Disegnare il flowchart corrispondente.

```
char flg_tx;
int count = 300;


void main(void)
{
        gpioInit();      // GPIO setting (inputs and outputs we need)
        Timer1Init();    // timer1 configuration, event every second
        UartInit();      // UART peripheral configuration and Rx/Tx functions

        while(PORTFbits.RF3 == 0); //button (RB3) to start countdown
        Timer1InterruptEnable();
        Timer1_on();

        while(1)
        {
                if(flg_tx && count>0)
                {
                        sprintf(strg, "%d sec", count);
                        putString(strg);
                        flg_tx = 0;
                }
                else if (count == 0)
                {
                        putString("Timer elapsed");
                        Timer1_off();
                }

        }//end while
}// end main


void _ISR(_TIMER_1_VECTOR, ipl2)
Timer1IntHandler(void)
{
        count --;
        flg_tx = 1;

        /* Clear interrupt flag*/
        IFS0bits.T1IF = 0;
}
```

8. (15 punti) **A real-time clock**
Consegnare "NomeCognome.zip" con source files e header files sulla piattaforma.

Scrivere un programma su μC che simula un orologio digitale.
Attraverso un pulsante (*button*) viene fatto partire il **Timer1** che tiene traccia di decimi di secondo, secondi e minuti.
Ogni secondo viene inviato su un terminale lo scorrere del tempo nella forma *XXm:YYs*.



Mappa I/O:

- RF0 per lettura pulsante
- RF12 linea di trasmissione UART4

Funzioni:
La pressione del pulsante attiva il Timer1. Il Timer1 viene configurato per generare un evento di **interrupt** ogni *0.1 secondi* a partire da:

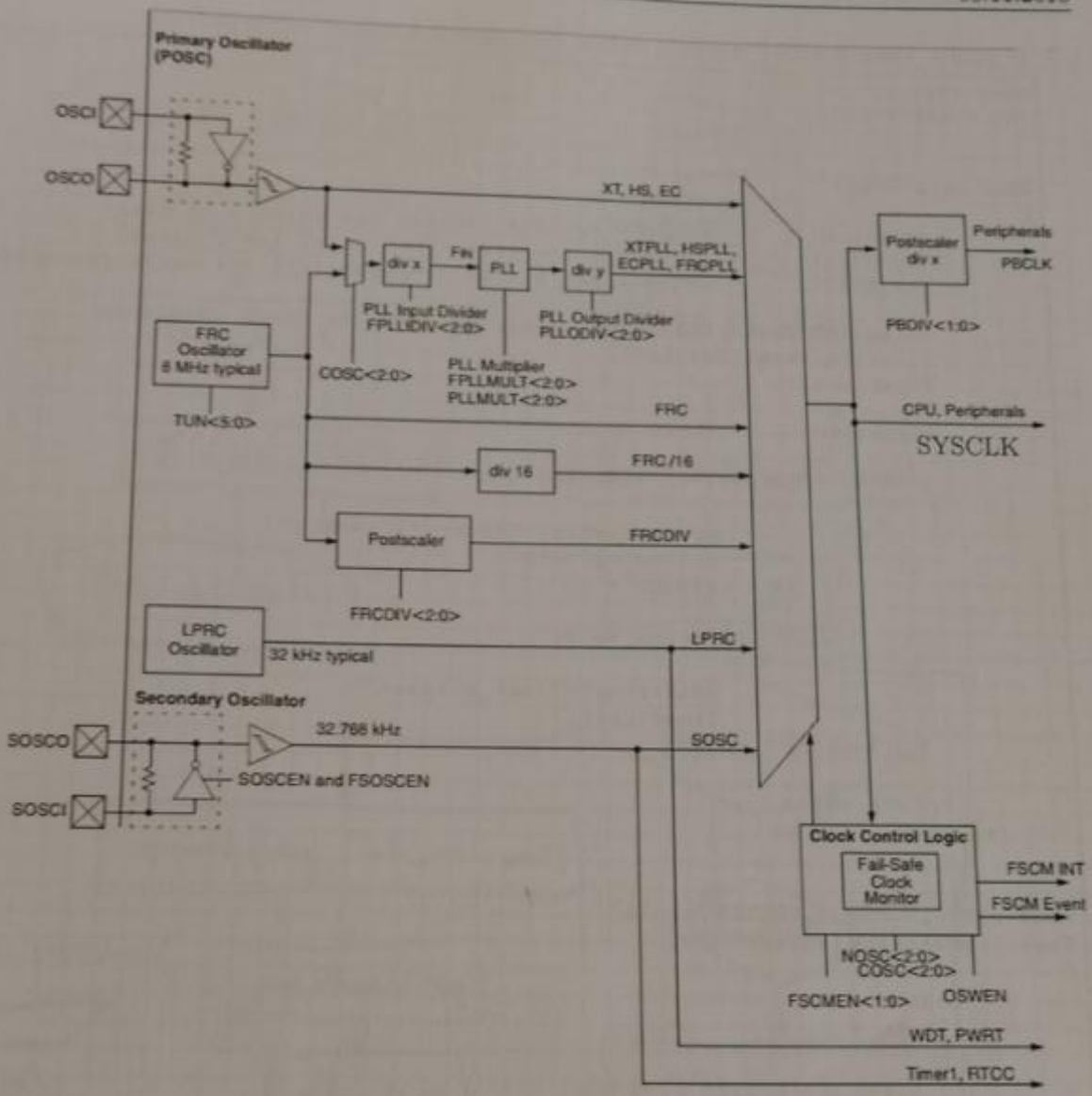
- Oscillatore interno primario (FRC) = 8MHz, System clock = 32MHz, PBCLK = 16MHz
- Prescaler timer = 64
- Period Register = 25000-1

Il servizio all'interruzione

- aggiorna le variabili globali *dSec* (decimi di secondo), *Sec* (secondi), *Min* (minuti) e
- abilita un flag per l'invio (nell'*endless loop*) su UART

Ogni secondo viene trasmessa, su un terminale, una stringa con minuti e secondi trascorsi.
La comunicazione seriale asincrona deve essere configurata in modo da avere:

- 8bit
- no parity
- 1 stop bit
- 115200 baud
- No HW control



FPLLIDIV<2:0>	PLLMULT<2:0>	PLLODIV<2:0>
000 = 1x divider	111 = clk mult by 24	111 = PLL out divider by 256
001 = 2x divider	110 = clk mult by 21	110 = PLL out divider by 64
010 = 3x divider	101 = clk mult by 20	101 = PLL out divider by 32
011 = 4x divider	100 = clk mult by 19	100 = PLL out divider by 16
100 = 5x divider	011 = clk mult by 18	011 = PLL out divider by 8
101 = 6x divider	010 = clk mult by 17	010 = PLL out divider by 4
110 = 10x divider	001 = clk mult by 16	001 = PLL out divider by 2
111 = 12x divider	000 = clk mult by 15	000 = PLL out divider by 1