Computer Interfacing (CI) FIB. Segon parcial **12/01/2022**

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

P1. (1 punt)

Estudiant exàmens d'altres anys hem trobat aquesta espantosa funció i la volem provar amb el nostre PIC 18F45K22 amb l'oscil·lador de 8MHz. Calcula (justificadament) quant de temps triga a executar-se.
Suposeu que la cridem al principi del main( ) sense haver configurat res que l'afecti. Podeu ignorar els temps de crida, retorn, etc.

```
void Patata (void)
{
        int i = 0;
        T0CON = 0x86;
        while (i < 25)
        {
                if (TMR0L == 250)
                {
                        TMR0L = 0;
                        i++;
                }
        }
}
```

Amb T0CON=0x86 "Engeguem" el TIMER0 amb un PRESCALER de 128 i entrada de Fosc/4 que són 500ns, per tant s'incrementarà cada 64us.
Per tant suposem que el programa farà 25 vegades una espera de 250 tics de 64us, això dóna que el temps serà de 25x250x64us=400000us=400ms=0,4s.

Caldria tenir en compte però que el TIMER0 ja estarà engegat quan s'executa el programa. Veieu al formulari que per defecte el bit TMR0on està a 1 després del RESET, per tant el primer cop, trobarem el TMR0L amb un valor entre 0 i 255, per tant el temps de la funció PATATA, serà, i d'aquí el seu nom:
1 cop (0..255)x64 us `+ 24 cops x 250 x 64 us = (384000 us .. 400320us) un valor dins d'aquest rang.

P2. (1,5 punt)

Uns companys volen usar la unitat 1 de Compare del PIC i el TIMER1 per generar una funció Delay que comptarà **microsegons** fent una espera activa. Tenen clar com serà la funció (i que de moment només servirà per valors de 0 a 65535). Suposeu Fosc = 8MHz.

```
void Delay_us (unsigned int temps)
{
        TMR1=0;
        CCPR1= temps;
        while (! CCP1IF);
        CCP1IF=0;
}
```

Creuen però que s'han oblidat inicialitzar algunes coses… Els pots ajudar omplint aquesta funció que caldrà cridar un cop al principi del main( ) per configurar-ho tot bé?

Cal arrancar el Timer1, configurar-lo a 1 us, deixar el Gate Enable a 0, associar el Timer1 a la unitat de CCP1 i posar la unitat CCP1 en mode COMPARE generant interrupció.

```
void Init_Delay_us (void)
{
        T1CON = 0b01110011; (o T1CON=0b00010011; o T1CON=0x73 o T1CON=0x13 )
        TMR1GE=0;
        CCPTMRS= 0bxxxxxx00; (o CCPTMRS=0x00 o CCPCCPTMRS &= 0xFC);
        CCP1CON= 0bxxxx1010; (o CCP1CON=0x0A)
}
```

No cal activar el IE ni res d'interrupcions perquè només usem el FLAG no la RSI. La funció Delay_us hauria de posar el flag a 0 al principi per si de cas estava ja a 1.

P3. (1 punt)

Comprensió de la unitat CCP.
Indica amb una X **quina** de les afirmacions és certa per cada plantejament (sols hi ha una correcta).

Encert suma 1/4. Error resta 1/12. Blanc no afecta. Només en aquesta, no cal justificar les respostes.

3.1 Pel PIC 18F45K22, amb Fosc=8MHz, el període del senyal generat amb la unitat de PWM…

	Serà com a mínim de 2048 us (microsegons).
X	Serà com a màxim de 2048 us (microsegons).
	Serà com a mínim de 125 ns (nanosegons).
	Totes les proposicions anteriors són falses.

El periode es determina amb (PR+1)x4xTosc**x**PRESCALER = (255+1)x4x125nsx16=2048us
Tosc és 125ns, el PRESCALER més gran 16, i PR major 255.
El mínim no pot ser ni 125 ni 2048 us.

3.2 Per generar una sortida en un PIN del micro amb la màxima freqüència possible…

	Usarem la unitat Capture perquè és la única lligada als Timers de 16 bits (1/3/5) i tindrem més resolució (fins a 65536 valors).
	Usarem la unitat Capture perquè els seus Timers associats (1/3/5) permeten connectar Fosc (no Fosc/4) i així tenir la freqüència base més alta.
	Podem usar les cinc unitats de Capture sincronitzadament, CCPTMRS0 = CCPTMRS1 = 0x00, així totes aniran amb el TIMER1.
X	Totes les proposicions anteriors són falses.

La unitat de Capture és per gestionar entrades, no sortides. Totes són falses.

Computer Interfacing (CI) FIB. Segon parcial **12/01/2022**

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

3.3 Amb el PIC 18F45K22, amb Fosc=8MHz, la unitat de PWMx farà una interrupció CCPxIF…

	Cada cop que el comparador amb PR indica que s'ha de fer reset al Timer.
	Cada cop que actualitza el Duty Cycle (còpia dels 8+2 bits de CCPRxL:CCPxCON a CCPRxH...).
	Només si el valor de Duty Cycle és menor que el de Període, per avisar de l'error.
X	Totes les proposicions anteriors són falses.

La unitat de PWM no pot generar interrupcions, mirar esquema. La idea és que queda funcionant autònomament, no té res a "comunicar" al programa principal.

3.4 Amb un PIC18F45K22 (Fosc max = 64MHz), volem generar un senyal PWM de període 10us (microsegons). La resolució (nombre de senyals diferents que podem generar a la sortida)…

X	Podrà ser major quan més gran sigui la freqüència Fosc que triem pel processador .
	Podrà ser major quan més petita sigui la freqüència Fosc que triem pel processador.
	Serà independent de la freqüència Fosc que tingui el processador.
	Totes les proposicions anteriors són falses.

Un cop determinat el període de 10 us, tindrem 10 us = (PR+1) x 4 x Tosc x PRESCALER
Aïllem (PR+1) i ens queda (PR+1) = 10 us / (4 x Tosc x PRESCALER)
Per tant (PR+1) = 10 us x Fosc / 4 x PRESCALER
Resolució és: Log2 (4 x (PR+1)) = Log2 (10us x Fosc / PRESCALER) =
Log2(10) + Log2(Fosc) - Log2(PRESCALER) = ct + Log2 (Fosc) + ct i serà creixent amb la Fosc.

P4. (1,5 punt)

Tenim la següent funció per rebre el caràcters al port sèrie d'un PIC18F45K22 funcionant amb una Fosc=4Mhz

```
void interrupt RSI (void)
{
        if (RC1IF && RC1IE){
                caracter_llegit=RCREG1;  // RC1IF bit is read-only. It gets cleared automatically
                                         //when received character is read
        }
}
```

Configura tot el que sigui necessari per a que la funció anterior rebi caràcters a 115200 bauds

La millor configuració disponible és amb BRG16=1, BRGH=1 i SPBRG=8 que s'obté un baudrate de 111111bauds. Això suposa un error d'un -3,55%. La configuració és:

```
SPEN=1;
TRISCbits.RC7=TRISCbits.RC6=1;
SPBRG=8;
BRGH=1;
BRG16=1;
SYNC=0;
RX9=0;
CREN=1;
RC1IE=1;
PEIE=1;
GEIE=1;
```

Creus que podries configurar-ho si enlloc d'una Fosc de 4MHz tinguéssim una d'1MHz?

Amb una Fosc = 1MHz la millor configuració disponible és 125000bauds. Aquest baudrate suposa un error de 8,5% respecte al baudrate desitjat. Amb errors tan grans es produeix framing error i no es pot fer servir.

P5. (1,5 punt)

Contesta breument a les següents preguntes sobre busos de comunicació:

Computer Interfacing (CI) FIB. Segon parcial **12/01/2022**

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

Quins fils componen un cable USB 2.0? L'USB és síncron o asíncron?

En un cable USB 2.0 podem trobar: GND (negre), 5V (vermell), D+ (verd) i D- (blanc).
La transmissió USB és asíncrona

Quines senyals (fils) podem trobar en una comunicació SPI? L'SPI té control d'errors?

A l'SPI trobem els senyals
MOSI - Master Output, Slave Input
MISO - Master Input, Slave Output
SCLK - Clock
SS - Slave select. Opcional segons el tipus de connexió

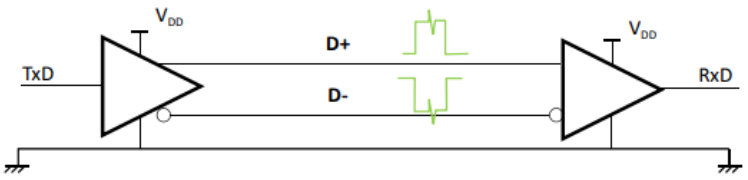
L'SPI no disposa de control d'errors

Com que el PIC18F45k22 no té hardware específic per a implementar el 1-wire, no podem fer servir el nostre micro amb aquest tipus de comunicacions. Hi estàs d'acord amb aquesta afirmació?

Fals. Tot i no tenir hardware específic es pot implementar mitjançant bit banging, és a dir, fent servir el firmware per controlar els pins implicats en la comunicació.

Què vol dir que un protocol de comunicació fa servir una senyal diferencial per enviar les dades? Quin és el principal avantatge de la senyal diferencial?

En transmissió diferencial s'envien les dades de forma complementaria fent servir un parell de senyals (les dades per un fil i les dades negades per un altre). El circuit de recepció respon a la diferència entre les dos senyals enviades.
La immunitat al soroll és el principal avantatge del senyal diferencial. Si s'introdueix soroll durant la transmissió, aquest quedará molt atenuat al realitzar la diferència al punt de recepció



Una comunicació I2C sempre comença amb l'enviament per part del master de la condició d'start seguida de 8 bits. Per a què serveixen aquest 8 bits?

Són els 7 bits d'adreça i el bit de Read/Write

P6. (1 punt)
Hem configurat el conversor AD del PIC amb les següents tensions de referència: V_{REF-} = 1V i V_{REF+} = 5V. Fosc és 12MHz, i el bit ADFM està a 0 (justificat a l'esquerra). Li posem un voltatge constant a l'entrada, el convertim (respectant les restriccions de Tad i Tacq) i obtenim als registres de resultat els següents valors: ADRESH = 0xCB, i ADRESL = 0xC0.
Quina tensió analògica (en Volts) tenim a l'entrada del conversor AD?

El primer que fem és obtenir el valor de 10 bits de la conversió. Com que sabem que el valor està justificat a l'esquerra, obtenim els 10 bits més alts entre els registres ADRESH i ADRESL.

ADRESH	ADRESL
11001011	**11**000000

Per tant, el valor de 10 bits és: 1100101111, equivalent a 815 en decimal.
Segons la fórmula per calcular la dada obtinguda per un convertidor AD, si aïllem Vin ens dóna:
Vin = [Dout / (2^N - 1)] * (Vref+ – Vref-) + Vref-

Sabem que el nostre PIC 18F45K22 té un ADC de 10 bits, i que V_{REF-} = 1V i V_{REF+} = 5V. Per tant:
Vin = [815 / 1023] * (5 – 1) + 1 = 4.187V

P7. (1,5 punts)
La revista "*Distorsió*" ha decidit emprendre una nova idea de negoci i fabricar guitarres elèctriques. Cadascuna de les 6 cordes té un sensor per a donar-nos informació de l'àudio de la corda. Cada corda es llegeix per un canal AD diferent, i es va fent multiplexació en el temps per a llegir tots els canals cíclicament amb igual temps de mostreig per cadascun (ch1-ch2-ch3-ch4-ch5-ch6-ch1-ch2- etc.).
Suposeu que la freqüència més elevada que genera l'instrument és de 4kHz.
S'ha utilitzat el PIC18F45K22 per aquesta aplicació. La Fosc és de 16MHz. Han configurat ACQT<2:0>=0b100, ADCS<2:0>=0b010. Si us calen, assumiu uns requisits de Tacq ≥ 7,45us i Tad ≥ 1us.

7.1) Calcula el temps total de mostreig d'un valor analògic.

Sabem que ADCS<2:0>=0b010, per tant Tad = 32 / Fosc = 32 / 16MHz = 2us (compleix amb mínim de 1us)
Sabem que ACQT<2:0>=0b100, per tant Tacq = 8 Tad = 8 * 2us = 16us (compleix amb mínim de 7,45us)

El temps total de mostreig d'1 valor engloba la fase d'adquisició, la de conversió (11 Tad) i després tenim 1 Tcy (descàrrega del condensador). Però habitualment simplifiquem arrodonint a temps d'adquisició + 12 Tad. Per tant:

Tmostra = Tacq + Tconv = 8 Tad + 12 Tad = 20 * 32 / 16MHz = 40us

7.2) Si ens fixem en una corda en concret qualsevol, calcula quantes mostres per segon llegirem d'aquesta corda.

La freqüència de mostreig total, comptant les mostres de totes les cordes, seria:
Fmostreig = 1 / Tmostreig = 1 / 40us = 25000 Hz

Ara bé, ens demanen per les mostres per segon només d'una corda en concret. Com que hi ha sis cordes, les mostres per segon s'han de repartir entre les 6 cordes. Per tant:
Fmostreig_1corda = Fmostreig / 6 ≈ 4167 Hz
7.3) Aquesta configuració permetrà adquirir l'àudio de la guitarra sense que es produeixi *aliasing*? Si no ho permet, podries fer algun canvi de configuració del perifèric per tal que funcionés bé?

Segons el criteri de Nyquist, per evitar *aliasing* hem d'adquirir mostres a una freqüència com a mínim del

Cognoms, Nom _____ DNI _____

Tota resposta sense justificar es considerarà nul·la !

doble de la freqüència més alta que genera l'instrument. L'enunciat ens diu que la més alta serà de 4kHz.

Per tant, hauríem de mostrejar cada corda a una freqüència de 8kHz. La freqüència de mostreig de cada corda hem calculat que serà de 4,17kHz aprox. Podem concloure que l'àudio mostrejat tindrà *aliasing*. Per tant la guitarra sonarà distorsionada, i la revista "*Distorsió*" haurà de demanar ajut a un/a Fiber per a dissenyar bé el seu producte.

Per tal que funcionés bé, caldria que cada corda agafés mostres almenys a 8kHz. Per tant, entre totes les cordes, hauríem de tenir:
Fmostreig ≥ 8kHz * 6 = 48kHz
Tmostra ≤ 1 / 48kHz = 20,83us
20 * x / 16MHz ≤ 20,83us x ≤ 16,67

Per exemple, la opció **ADCS<2:0>=0b101** compleix el requisit doncs divideix Fosc entre 16.

Amb el nou ADCS, tindrem Tad = 16 / Fosc = 16 / 16MHz = 1us (compleix amb mínim de 1us)
Comprovem que l'anterior ACQT segueixi sent vàlid: Tacq = 8 Tad = 8 * 1us = 8us (compleix amb mínim de 7,45us).
Tmostra = 8 Tad + 12 Tad = 20 * 16 / 16MHz = 20us Fmostreig = 1 / 20us = 50000 Hz
Fmostreig_1corda = Fmostreig / 6 ≈ 8333 Hz

La nova freqüència de mostreig de cada corda serà de 8,33kHz aprox, complint amb el criteri de Nyquist.

P8. (1 punt)
Una prestigiosa farmacèutica catalana, ha dissenyat una vacuna contra el Covid-19. Com a control de qualitat, cada vial és mesurat per saber quant líquid s'hi ha introduït. Disposem d'un sensor que ens dóna tensions entre 2V i 4V, corresponents a volums d'entre 0 i 3 mil·lilitres (de manera lineal). El senyal analògic provinent d'aquest sensor és mostrejat amb l'AD d'un PIC18F45K22. Les tensions de referencia són V_{REF-} = 0V i V_{REF+} = 5V.
Necessitem obtenir una resolució de 0.01 mil·lilitres en els valors llegits amb el AD.

¿Quants bits són necessaris per al conversor AD?
¿Són suficients els bits del conversor del nostre PIC, o caldria un conversor AD de més bits?

Primer calculem el número d'esglaons necessaris en la escala de mil·lilitres:
esglaons = (valor_max – valor_min) / resolució = (3ml – 0ml) / 0.01ml = 300 esglaons

Això vol dir que entre 2V i 4V hem de mesurar 300 esglaons amb el ADC.
Amb una regla de tres, podem calcular quants esglaons necessitem entre les tensions de referència mínima i màxima:

esglaons entre V_{REF-} i V_{REF+} = [300 / (4V – 2V)] * (5V – 0V) = 750 esglaons

El número de codis diferents necessaris serà de 751 codis.

Per tant, podem calcular el número de bits:

Nbits = $\log_2 751 \approx 9{,}55$ bits

Com que el nostre PIC18F45K22 té un ADC de 10 bits, sí que serà adequat per a obtenir la resolució demanada.