Computer Interfacing *Segon Parcial* *21/11/2016*

Cognoms i Nom: __ UNA POSSIBLE SOLUCIÓ _____ Doc. Identitat: _____

Totes les respostes han d'estar degudament justificades

1) Explica en quins casos podem utilitzar la instrucció RETFIE FAST i en quins casos haurem d'utilitzar la instrucció RETFIE per retornar d'una rutina d'atenció a la interrupció? (1,5 pts.)

Durant la crida a una interrupció sempre es guarda a la pila l'adreça de la instrucció que s'estava executant en el moment que es va produir la interrupció. A més, en els "*Shadow registers*" es guarda el context d'execució del programa, és a dir, el valor dels registres WREG, BSR i STATUS. Si no s'han definit prioritats en les interrupcions no hi ha problema en fer servir la instrucció RETFIE FAST per restaurar el valor d'aquests registres automàticament quan es torna el flux d'execució al programa principal. Ara bé, si hem programat diverses interrupcions amb prioritats diferents, podria donar-se el cas d'estar executant una interrupció de baixa prioritat i que es produís una interrupció d'alta prioritat durant l'execució de la mateixa. En aquest cas la rutina d'atenció a la interrupció d'alta prioritat passaria a ser executada sobreescrivint els "shadow registers" i per tant, destruint els valors que es van escriure quan es va iniciar l'execució de la rutina de baixa prioritat. Si la rutina de baixa prioritat fes servir un RETFIE FAST per retornar, restaurarem el registres de context amb un valor equivocat. Per evitar aquesta situació la rutina de baixa prioritat ha de salvar els registres de context "a mà" en variables temporals i també restaurar-les "a mà" abans de cridar la instrucció RETFIE.

2) Es vol utilitzar el TIMER3 per a generar una base de temps. (1,5 pts.)

a) A quina freqüència s'activa el TMR3IF si T3CON= 0x91, i Fosc= 32MHz.

El valor de T3CON indica que està en mode 16 bits, prescaler 1:2, i ON.
Fosc= 32 MHz -> Fosc/4= 8 MHz -> F_{TIMER}= 4 MHz -> T_{TIMER}= 0,25 µs
Com el TIMER3 treballa en free-running, el TMR3IF s'activa a cada volta completa (2^{16}) del TIMER.
4 MHz/65536= 61,03 Hz

Una alternativa a l'hora de fer els càlculs
2^{16}*T_{TIMER}= 65536*0,25 µs= 16384 µs= 16,383 ms -> f=1/T=1/16,383 ms= 61,03 Hz

b) Escriu el codi C per a la inicialització del TIMER3 i del mecanisme d'interrupcions, i la rutina de servei a la interrupció, per a generar una interrupció de baixa prioritat cada 10 ms amb el TIMER3. Fosc= 32MHz.

Continuem amb la mateixa configuració que l'apartat a) -> T3CON= 0x91

10 ms / T_{TIMER}= 10 ms/0,25 µs = 40000 tics, per tant cal inicialitzar TMR3= 65535-40000= 25535

INICIALITZACIÓ

```
milis10= 0;
// Boleà que s'assigna a cert en el tractament de la
// interrupció TIMER3 i a fals en el codi de l'aplicació


T3CON= 0x91;

TMR3= 25535;  // Overflow en 40000 tics = 10 ms

TMR3IF=0; TMR3IP=0; TMR3IE=1;  // Inicialitza TIMER3

IPEN=1; GIEH=1; GIEL=1; // Bits globals interrupcions
```

RUTINA INTERRUPCIÓ

```
void interrupt low ISRL (){

    if (TMR3IF && TMR3IE) {
      TMR3IF=0;
      milis10= 1; // S'ha produït una interrupció del TIMER3
      TMR3= 25535+TMR3;
      // Tenim en compte els cicles que ha avançat TIMER3
    }
    // Tractament d'altres interrupcions de baixa prioritat
}
```

3) Per una aplicació, s'han de comptabilitzar el nombre de canvis (flancs de pujada i flancs de baixada) que es produeixen en el senyal binari **S**, generat per un sensor, utilitzant un PIC18F4550 –Fosc= 8MHz-. Per les dues opcions proposades:

1) Connexió del senyal **S** a l'entrada RA4, i detecció dels canvis per enquesta d'aquesta entrada
2) Connexió del senyal **S** a l'entrada RB0, i detecció dels canvis per interrupcions INT0. (3 pts.)

CODE 1 – Pooling

```
#include <xc.h>

void init_pic1 () {
// INIT PIC to read Sensor
// by pooling
}

void main(void) {
int comptador=0;
byte lastS, S;

init_pic1 () ;
S= PORTAbits.RA4;
lastS= S;
  while (1) {
    S=PORTAbits.RA4;
    if ( S!=lastS ) comptador++;
    lastS= S;
    // APPLICATION CODE
    // EXECUTION TIME 350 mseg.
  }
}
```

CODE 2 - Interruption

```
#include <xc.h>

int comptador=0;

void interrupt ISR() {
   // Complete the
   // Interrupt Service Routine
}

void init_pic2() {
// INIT PIC to read Sensor
// using interruptions
}

void main(void){
 init_pic2();
 while (1) {
  // APPLICATION CODE
  // EXECUTION TIME 350 mseg.
 }
}
```

a) Completa la rutina **init_pic1** del codi 1 amb la finalitat de configurar adequadament el PIC segons l'opció 1.

```
void init_pic1 () {  TRISAbits.RA4=1;   ADCON1= 0x0F;  CMCON= 0x07;  } // Tots els port digitals, RA4 input
```

b) Completa les rutines **init_pic2** i **ISR** del codi 2 amb la finalitat de configurar i programar adequadament el PIC segons l'opció 2.

```
void init_pic2 () {
  TRISBbits.RB0=1;   ADCON1= 0x0F;
  // Configurem RB0 com a entrada digital
  INT0IF=0;   INTFIE=1;   INT0IP=1;
  INTEDG0= 1; // Configurem INT0  prioritat alta
  IPEN=1;   GIEH= 1;
  // Configurem i habilitem el mecanisme interrupcions
}
```

```
void interrupt ISR() {
   if (INT0IE && INT0IF) {
      INTEDG0=  ~ INTEDG0; // Canvi polaritat del flanc a detectar
      INT0IF=0; // Esborrem interruption flag
      comptador++;  // Comptabilitzem nou flanc
   };
// Aquí va el codi per tractar les altres interrupcions d'alta priorita
}
```

c) Quina és la latència (temps entre el canvi del senyal **S** i l'actualització de la variable de comptatge) en el pitjor cas, en cada una de les opcions.

Opció 1. El pitjor cas en la situació permanent, és que **S** canviï de valor just després de consultar el seu valor (execució de la línia de codi S=PORTAbits.RA4 dintre del while), llavors la detecció no és realitza fins la propera lectura de **S** que es produeix

350 mseg + temps execució instruccions relacionades amb if i assignació ≈ 350 mseg.

Opció 2. De forma quasi instantània al canvi del senyal s'activa la interrupció i s'executa la rutina del servei a la interrupció. Per tant el temps entre el canvi del senyal S i l'actualització de la variable comptador és de 4 cicles (per anar a la ISR) + temps execució del codi d'ISR abans d'executar l'increment de la variable (comptador++) ≈ 44 cicles = 22 μseg.

d) Per a cada opció, determinar el temps necessari, com a mínim, entre canvis consecutius de **S** per a que puguin ser detectats.

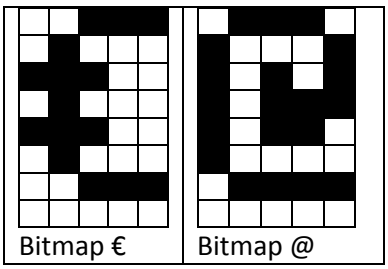
Si es produeix un nou canvi abans que s'hagi detectat i processat l'anterior es perdrà aquesta informació, i per tant no comptabilitzarem el nombre total de canvi de **S**. Aquest fet té a veure amb la latència de l'apartat anterior.
Opció 1. Si es produeixen 2 o més canvis en menys de 350 mseg no els podrem detectar tots. Només en detectarem 0 o 1 canvi.
Opció 2. Si es produeixen un nou o més canvis abans d'esborrar INT0IF (INT0IF=0) dintre de la ISR (segons hem calculat prèviament, en menys de 22,5 µseg) no els podrem detectar tots.

e) Quina de les dues opcions considereu millor. Justifiqueu la resposta.

A la vista de la latència i del temps mínim entre canvis del senyal, l'opció 2 –tractament per interrupcions- és preferible.

4) A més dels caràcters alfanumèrics, es vol mostrar al GLCD diferents símbols (p.ex. €, @, $, *, →, ...). Amb aquesta finalitat s'ha d'ampliar la llibreria "rutines_GLCD" que utilitzeu al laboratori, amb el mapa de caràcters, els bitmaps dels símbols (relació entre el codi, el símbol, i els valors dels píxels associats a cada símbol, els símbols es representa en un bitmap de 8x5 pixels), i la rutina writeSym per mostrar el símbol al GLCD. (2 pts.)

```
byte symbols [5*N]={ .... };   //  Bitmap for N symbols, N<=100
void writeSym (byte page, byte y, byte codi);   // Write symbol at GLCD
```



a) Completeu els valors del vector **symbols** per a representar € i @ segons el següents bitmaps. Contempleu únicament els símbols € (codi 0) i @ (codi 1) (N=2).

b) Escriure el codi de la rutina **writeSym** per mostrar un símbol, a la pàgina i columna indicats.

Nota: A les llibreries rutines_GLCD disposeu de les rutines/funcions void writeByte (byte page, byte y, byte data), byte readByte (byte page, byte y), void SetDot(byte x, byte y), void ClearDot (byte x, byte y), entre d'altres.

a)

```
byte symbols [5*2]={ 0x14, 0x3E, 0x55, 0x41, 0x41, 0x3E, 0x41, 0x5D, 0x59, 0x4E };
//  Bitmap per 2 símbols. posicions 0-4 símbol €, i posicions 5-9 símbol @
//  El píxel superior de la columna està associat amb el bit 0 de la pàgina, i el píxel inferior amb el bit 7.
```

b)

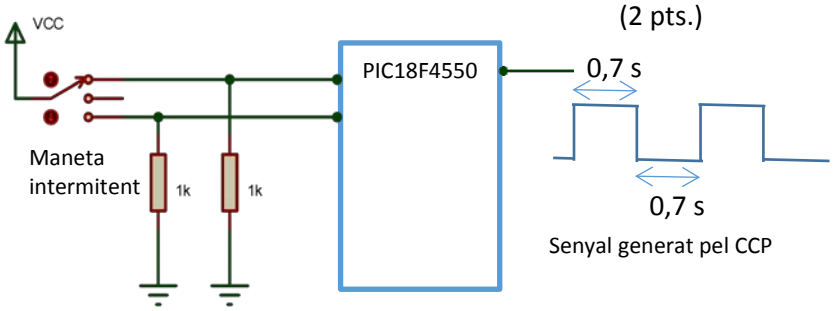
```
void writeSym (byte page, byte y, byte codi) {
  byte i;
  for (i=0; i<5; i++)
        if (y+i>127) break;
        writeByte(page,  y+i,  symbols[(int)(codi*5)+i];  // escrivim a la memòria del GLCD les 5 columnes del símbol
}
```

Cognoms i Nom: _____ Doc. Identitat: _____

Totes les respostes han d'estar degudament justificades

5) Un fabricant de vehicles ens demana dissenyar diversos circuits pel seu nou model. Per raons econòmiques volen gestionar diferents dispositius del cotxe amb un únic microcontrolador PIC18F4550 que funciona a una freqüència de rellotge de 2MHz. Un dels dispositius que volen gestionar són els intermitents. Ens demanen llegir des del PIC dos senyals provinents de la maneta de l'intermitent del panel de control del conductor, i que indiquem si s'ha d'encendre l'intermitent esquerra o dret. (2 pts.)

a) Com implementaríeu amb el PIC la lectura de l'estat de la maneta dels intermitents. Quina/es unitat/s del PIC faries servir i com la/es configuraria/es?



Podem fer servir els ports d'entrada/sortida. Configuraríem dos pins d'algun port que no entrin en conflicte amb el mòdul CCP com a pins d'entrada digital i connectaríem el senyal provinent de les palanques dels intermitents.

b) El fabricant també vol generar des del PIC un senyal per encendre els llums dels intermitents. Durant l'activació dels intermitents, el temps en que els intermitents estan encesos (0,7 segons) ha de ser igual al que estan apagats. Volem implementar aquesta solució fent servir el mòdul CCP2 del PIC amb el TIMER3 com a timer associat.
Quins valors assignaries als registres T3CON, CCP2CON, CCPR2H i CCPR2L per tal d'implementar aquesta solució?
Indica nomes els bits que has de modificar i el seu valor. Justifica la resposta i fes els càlculs pertinents.

T3CON. Cal escriure els bits T3CCP2 i T3CCP1 amb el valor '01' per assignar el Timer3 a la unitat CCP2.

CCP2CON. Caldrà configurar el mòdul CCP2 en mode comparació tocant els bits CCP2M3..CCP2M0. La configuració òptima per la solució seria la que complementa la sortida cada cop que es produeix una igualtat entre el valor del TIMER3 i els registres de comparació del CCP2 (cada 0,7 segons). Per tant els bits CCP2M3..CCP2M0 s'han de configurar amb el valor '0010'

CCPR2H i CCPR2L. Com que volem que el bit de sortida del CCP es complementi cada 0,7 segons i tenim un PIC que funciona a una freqüència de 2Mhz, significa que el comptador del timer s'incrementarà a una freqüència de 500.000 Hz. Per tant necessitem comptar 350.000 tics de rellotge per mesurar 0,7 segons. Els registres del TIMER3 són de 16 bits i no podem comptar fins un número tant alt. Per tant necessàriament haurem de fer servir un prescaler. El valor del prescaler que necessitem fixar en el TIMER3 és de 8 (bit T3CKPS1 = 1 i T3CKPS0 = 1 del registre T3CON). Fent servir aquest prescaler haurem de fixar els valors de registre CCPR2 a:

350.000/8 = 43.750 = 0xAAE6

CCPR2H = 0xAA

CCPR2L = 0xE6

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 7 **RD16:** 16-Bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6, 3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the capture/compare clock source for both CCP modules
01 = Timer3 is the capture/compare clock source for CCP2;
Timer1 is the capture/compare clock source for CCP1
00 = Timer1 is the capture/compare clock source for both CCP modules

bit 5-4 **T3CKPS1:T3CKPS0:** Timer3 Input Clock Prescale Select bits
11 = 1:8 Prescale value
10 = 1:4 Prescale value
01 = 1:2 Prescale value
00 = 1:1 Prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the device clock comes from Timer1/Timer3.)
When TMR3CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR3CS = 0:
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI (on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

REGISTER 15-1: CCPxCON: STANDARD CCPx CONTROL REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—(1)	—(1)	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as '0'(1)

bit 5-4 **DCxB1:DCxB0**: PWM Duty Cycle Bit 1 and Bit 0 for CCPx Module
Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
These bits are the two LSbs (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 **CCPxM3:CCPxM0**: CCPx Module Mode Select bits
0000 = Capture/Compare/PWM disabled (resets CCPx module)
0001 = Reserved
0010 = Compare mode: toggle output on match (CCPxIF bit is set)
0011 = Reserved
0100 = Capture mode: every falling edge
0101 = Capture mode: every rising edge
0110 = Capture mode: every 4th rising edge
0111 = Capture mode: every 16th rising edge
1000 = Compare mode: initialize CCPx pin low; on compare match, force CCPx pin high (CCPxIF bit is set)
1001 = Compare mode: initialize CCPx pin high; on compare match, force CCPx pin low (CCPxIF bit is set)
1010 = Compare mode: generate software interrupt on compare match (CCPxIF bit is set, CCPx pin reflects I/O state)
1011 = Compare mode: trigger special event, reset timer, start A/D conversion on CCPx match (CCPxIF bit is set)
11xx = PWM mode

REGISTER 9-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(1)

REGISTER 9-10: RCON: RESET CONTROL REGISTER

R/W-0	R/W-1(1)	U-0	R/W-1	R-1	R-1	R/W-0(2)	R/W-0
IPEN	SBOREN	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$

REGISTER 9-2: INTCON2: INTERRUPT CONTROL REGISTER 2

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP