FIB, Interfícies dels Computadors
Tercer parcial 12-1-2015 (1h30', Full 1/2)

COGNOMS:	
NOM:	**DNI:**

Responeu en aquest mateix full. Cal justificar totes les respostes.
Respostes sense un mínim text explicatiu no es tindran en consideració.

1. Es vol configurar la USART del microcontrolador PIC18F per realitzar transmissions asíncrones de 8 bits a 57600 bps. Quina seria la millor opció de configuració dels registres SYNC,BRG16, BRGH, SPBRGH: SPBRGL? El *clock* del sistema Fosc és de 10 MHz. (1.5 punts)

TABLE 20-1: BAUD RATE FORMULAS

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	Fosc/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	Fosc/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	Fosc/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = value of SPBRGH:SPBRG register pair

Si es calculen les tres opcions:

a) Fosc/(64 (n+1)) = 57600 -> n = 1.71 ; amb el valor enter més proper n = 2 s'obté un BaudRate = 52083 bps

b) Fosc/(16 (n+1)) = 57600 -> n = 9.85 ; amb el valor enter més proper n = 10 s'obté un BaudRate = 56818 bps

c) Fosc/(4 (n+1)) = 57600 -> n = 42,4 ; amb el valor enter més proper n = 42 s'obté un BaudRate = 58139 bps

L'opció c) és la més adient, doncs és la que introdueix menys error (Nota: BRG = 16bit, no vol dir que la tx. sigui amb 16 bits, sinó que indica que es faran servir els 16 bits del registre SPBRG).

2. Quina seria l'eficiència de transmissió (Bits de Dades Tx. / Bits Totals Tx.) d'una comunicació byte a byte I2C?

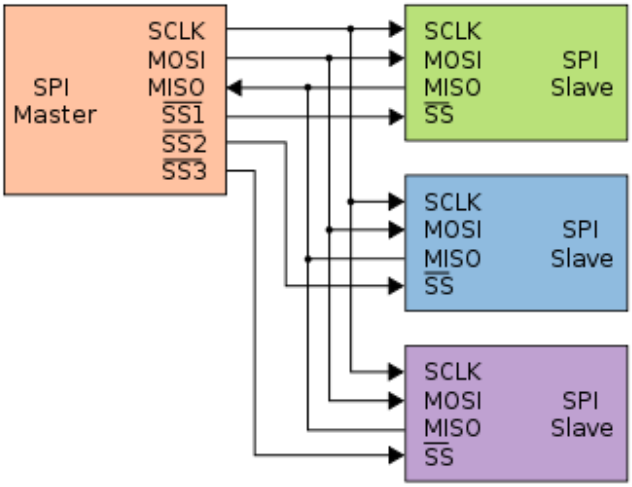
(1 punt)

Per tx. un byte són necessaris tx. 20 bits (START Condition + 7 Adreça I2C+ R/W + ACK + 8 bits Data + ACK + STOP)

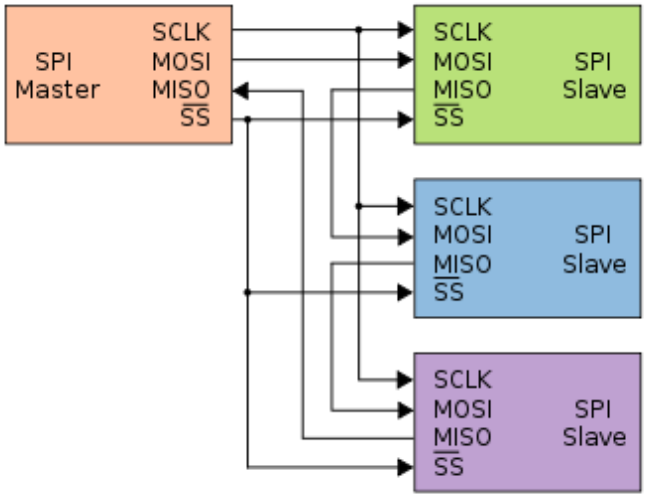
Llavors la eficiència és de 8/20 (40%)

3. Indiqueu, en dos esquemes separats, les connexions necessàries per establir una comunicació SPI entre un *master* i dos dispositius *slaves* en mode de circular i en mode esclaus independents. (1 punt)

Mode Independent

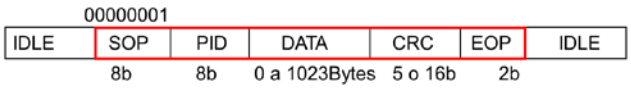


Mode circular



4. Calculeu el *bit rate* (*bps*) **efectiu** màxim en els casos següents: (2 punts)
a) en una transmissió USB de tipus *Bulk* i
b) en una transmissió USB de tipus Isòcrona, si:

Format dels paquets



Vel de Tx. (FS) 12 Mbps.
Mida del camp DATA en Tx. Isòcron = 128 Bytes
Mida del camp DATA en Tx. Bulk = 64 Bytes
Mida dels paquets IN/OUT = 34 bits
Mida dels paquets ACK = 18 bits
Mida dels paquets SOF (Start Of a Frame) = 34 bits
Bits en un *Frame* = 12000 bits
Període de Tx. d'un SOF = 1 mseg.

a) Per fer una transacció BULK es transmet un paquet IN, DATA i un ACK, que fan un total de 34 + 18 + 8+ 8 + 64*8+16+2 = 598 bits. En un frame hi ha 12000 bits menys 34 bits del SOF, llavors en el temps d'un frame es podran realitzar 20 transaccions de tipus BULK. Com que hi ha 1000 frames en un segon, la tx. efectiva màxima serà de 64*20*1000 Bytes/Segon.

b) Per fer una transacció Isòcrona es transmet un paquet IN, DATA sense ACK, que fan un total de 34 + 8+ 8 + 128*8+16+2 = 1092 bits. En un frame hi ha 12000 bits menys 34 bits del SOF, llavors en el temps d'un frame es podran realitzar 10 transaccions de tipus BULK. Com que hi ha 1000 frames en un segon, la tx. efectiva màxima serà de 128*10*1000 Bytes/Segon.

FIB, Interfícies dels Computadors
Tercer parcial 12-1-2015 (1h30', Full 2/2)

COGNOMS:	
NOM:	**DNI:**

Responeu en aquest mateix full. Cal justificar totes les respostes.
Respostes sense un mínim text explicatiu no es tindran en consideració.

5) Per un PIC18F4550 amb una Fosc de 32 MHz, (1.5 punts)

a) Quina és la màxima velocitat de transmissió en bits/segons (bps) que es pot assolir amb la interfície SPI ?

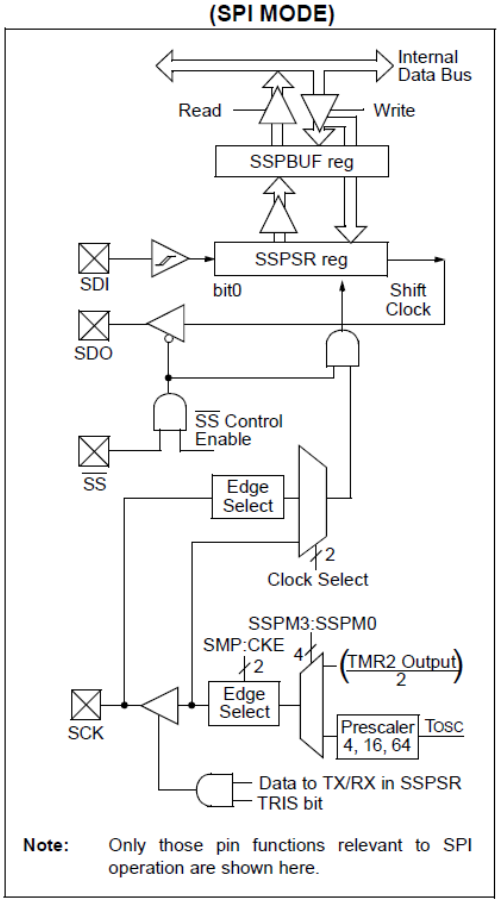
b) Es pot generar un *bit rate* de 1 Mbits/s ?

a) La màxima velocitat s'obté seleccionant **Fosc/4** com a origen del *clock* pel registre SSPSR. En aquest cas el *bit rate* màxim resultant seria 8 MHz.

b) El *clock* del registre SSPSR es pot generar a partir del Timer 2. Per generar un *bit rate* de 1 Mbps cal configurar el Timer 2 per a que activi la seva sortida a la freqüència de 2 MHz.

Per tant caldria configurar el Timer2 per a que activi la seva sortida cada 1µs, tenint present que Fosc= 32MHz, i assignar el valor als bits SSPM3:SSPM0 (=0011) per a que el clock del mòdul SPI sigui TMR2 output/2.

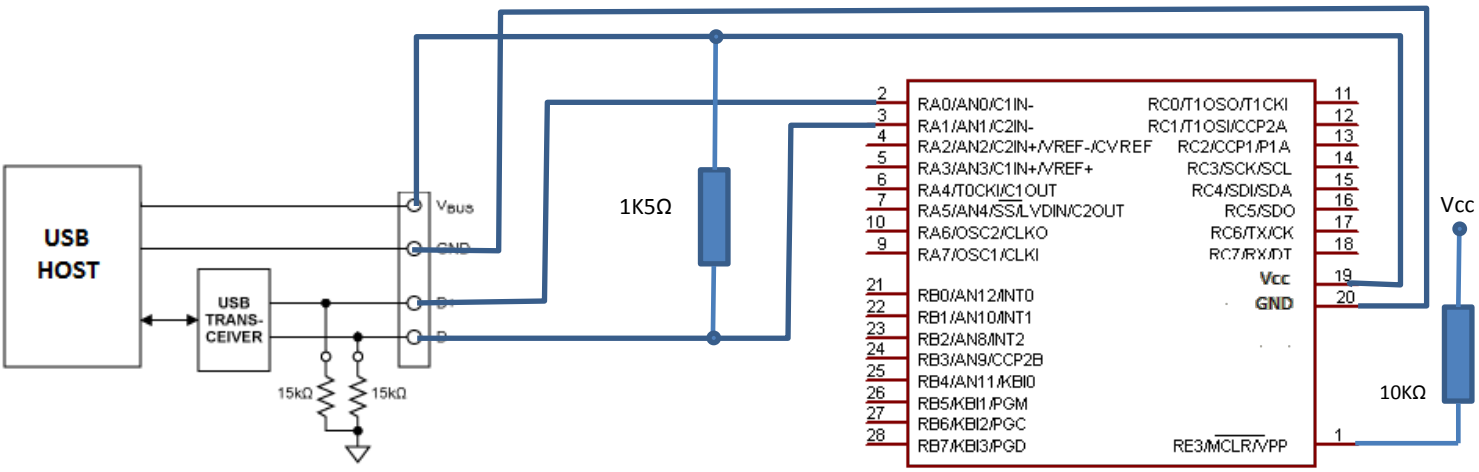
FIGURE 19-1: MSSP BLOCK DIAGRAM (SPI MODE)



Note: Only those pin functions relevant to SPI operation are shown here.

6) El microcontrolador de la següent figura no disposa d'un perifèric USB integrat, i per realitzar la funció de comunicació USB Low-Speed en mode *slave* s'ha optat per utilitzar la tècnica de *bit-banging* (emulació per programa). Es demana: (3 punts)

a) Completeu el diagrama elèctric següent per a la connexió del nostre sistema amb un host USB.



b) Implementeu la rutina SendPackectUSB(byte *pbody, unsigned int nbits), on *pbody* és un punter on es troba la informació a transmetre (PID, ADDR, ENDP, DATA i CRC), i *nbits* és el nombre de bits a transmetre. La informació està disposada de forma consecutiva bit a bit en M bytes (M = $\lceil nbits/8 \rceil$) i en l'ordre PID-ADDR-ENDP-DATA-CRC.

Nota: Cal tenir present que un paquet USB s'inicia amb un SOP, i es finalitza amb un EOP, i per tant la rutina SendPackectUSB, entre d'altres coses, també s'ha d'encarregar de l'enviament dels bits que formen el SOP i el EOP. En implementar la rutina de transmissió podeu ignorar el temps que triga la CPU en executar el codi (Tosc ≈ 0).

Per enviar informació en el protocol USB LS, cal tenir present:
- La informació s'envia bit a bit en format diferencial (excepte els 2 bits de l'EOP)
- En l'estat IDLE D+=0 i D+=1, l'estat J (es vol transmetre '1') D+=0 i D+=1 i l'estat K (es vol transmetre '1') D+=1 i D+=0

..- El bit rate és de 1.5Mbps, i per tant la durada de 1 bit a la línia és de 0,666 µs.

..- Cal implementar el procediment de bit stuffing i codificar la informació en NRZI.

.. - El paquet complert serà: SOP, PID, ADDR, ENDP, DATA, CRC i EOP (i per tant s'envien nbits+10 bits en total –que corresponen al SOP i EOP)

Un possible codi que realitza l'enviament d'un paquet segons el protocol USB LS en mode slave.

```
#include <xc.h>

typedef unsigned char BYTE;

void waitbittime(); // Wait to complete 1 bit period (active polling on TimerXIF)
void startbittime(); // Start a new bit period (1.5 Mbps in USB LS)

void SendPacketUSB(BYTE *pbody, unsigned int nbits) {
    char sop= 0x01;
    char aux, currentbit, prevbit, count, m, pending, value, i;
    unsigned int bitcount;

    ADCON= 0x0F;  // Set PORTA as digital output
    CMCON1= 0x07;
    TRISA= 0x00;  // Bit RA0 is D+  Bit RA! Is D-

    pending= 1; // Flag=> 1= Send SOP; 2= Send Information 0= Send EOP
```

```
  aux= 0x01;  // SOP value
  prevbit= 0; // D+ idle value in USB LS
  bitcount=0; // sended bit count
  count=0;    // counter for bit stuffing
  PORTA= 0x02; // Idle state for D+ and D- in USB LS
  startbittime(); // Start a new bit period (1.5 Mbps in USB LS)

  while ( pending ) {
    for (i=0; i<8; i++) {
        if (bitcount>=nbits) break; // All bits of packet information sent
        currentbit= aux & 0x80; // Test current MSB
        if (currentbit) count++; else count=0;
        if (count>6) { // Add extra bit - bit stuffing
            waitbittime(); // Wait to complete 1 bit period
            startbittime(); // Start a new bit period
            if (prevbit) value=0; else value=1;   // NRZI encoding
            if (value)          PORTA= 0x02; // J state in USB LS
            else                PORTA= 0x01; // K state in USB LS
            prevbit=value;
            count=1;
        };

        waitbittime(); // Wait to complete 1 bit period
        startbittime(); // Start a new bit period
        // NRZI encoding
        if (currentbit==0) {   // Switch level
            if (prevbit) value=0; else value=1;
            if (value)          PORTA= 0x02; // J state in USB LS
            else                PORTA= 0x01; // K state in USB LS
        };
        aux= aux <<1; // shift 1 bit left
        bitcount++;
    };

    if (pending==1) { pending=2; bitcount=0;};// First byte is SOP , now send information
    if (bitcount >= nbits) // All bit sent
          pending= 0;
    else {  aux= *pbody; // Get next byte to send
            pbody++;
    };
  }
  // Send EOP - 2 bits bit D+=0 and D-=0
  waitbittime();     // Wait to complete 1 bit period
  startbittime();   // Start a new bit period
  PORTA= 0x00;
  waitbittime();    // Wait to complete 1 bit period
  startbittime();  // Start a new bit period
  PORTA= 0x00;
  waitbittime(); // Wait to complete 1 bit period
}
```