Faculté des Sciences
Aix*Marseille Université

Année universitaire 2015/2016

Site : ☐ Luminy ☐ St-Charles ☐ St Jérôme ☐ Cht-Gombert ■ Aix-Montperrin ☐ Aubagne-SATIS

Sujet session de : ☐ 1er semestre - ■ 2ème semestre - ☐ Session 2

Durée de l'épreuve :1h30……..

Examen de : ☐L1/■L2/☐L3 - ☐M1/☐M2 - ☐ LP - ☐ DU Nom diplôme : L2 SPI

Code Apogée du module : SPI42AA Libellé du module : Informatique industrielle

Document autorisé : ■ OUI - ☐ NON (Poly de cours) Calculatrices autorisées : ■ OUI - ☐ NON

Vendredi 13 Mai 2016 13H00-14H30 *Licence 2 – SPI*

Aix Marseille Université

Informatique industrielle

Le polycopié de cours est autorisé.
Une calculatrice de type collège est autorisée.

Cet examen de 1 heure 30 est composé de 2 parties :

- *des questions de cours et exercices* (approximativement 50 mn),
- *un problème* (approximativement 30 mn).

Notes :

Vous rendrez votre sujet d'examen avec votre copie.

C'est sur le sujet que vous répondrez à une partie des questions.

Une annexe comporte des données utiles aux deux parties de l'examen.

Questions de cours

Attention

Vous êtes notés sur clarté de vos réponses. Pour avoir le maximum de points, elles devront être justes, courtes, et précises. Utilisez vos brouillons avant de vous jeter sur vos copies d'examen.

[A] Questions générales

1. Quels sont les principaux critères de performance d'un microcontrôleur ? Comment ces critères ont-ils évolué ces dernières années ?
2. Pour une instruction en assembleur, à quoi correspond l'opcode ? A quoi correspond l'opérande ? Donnez un exemple.
3. Quelle est la différence entre l'adressage direct et l'adressage indirect ? Dans quel cas utilise-t-on le registre FSR (File Select Register) ?

[B] Questions « architecture »

1. Identifier les éléments ci-dessous sur le schéma bloc de la *Figure 1* :

   -le compteur programme,
   -le bus de données,
   -le registre File Select Register,
   -l'unité arithmétique et logique,
   -la mémoire programme,
   -le module Timer 0.

2. Le microcontrôleur présenté *Figure 1* est-il de structure *Von-Neumann* ou *Harvard* (justifiez votre choix)? Est-ce un microprocesseur 8 bits, 16 bits ou 32 bits ?
3. Combien de ports entrée-sortie comporte ce microcontrôleur ?

[C] Question « codage binaire, hexadécimal et décimal »

1. Convertissez a=0x3A en binaire
2. Convertissez b=0x5C en binaire
3. En utilisant les questions 1 et 2, effectuez l'opération a-b en passant par le calcul du complément à deux de b. Donnez le résultat sous forme hexadécimale.

4. Effectuez l'opération *logique* (binaire) suivante : c xor w où c=0101 et w=0001 (xor : ou exclusif).
5. Effectuez l'opération *logique* (binaire) suivante : d ior w où d=1010 et w=0011 (ior : ou inclusif).

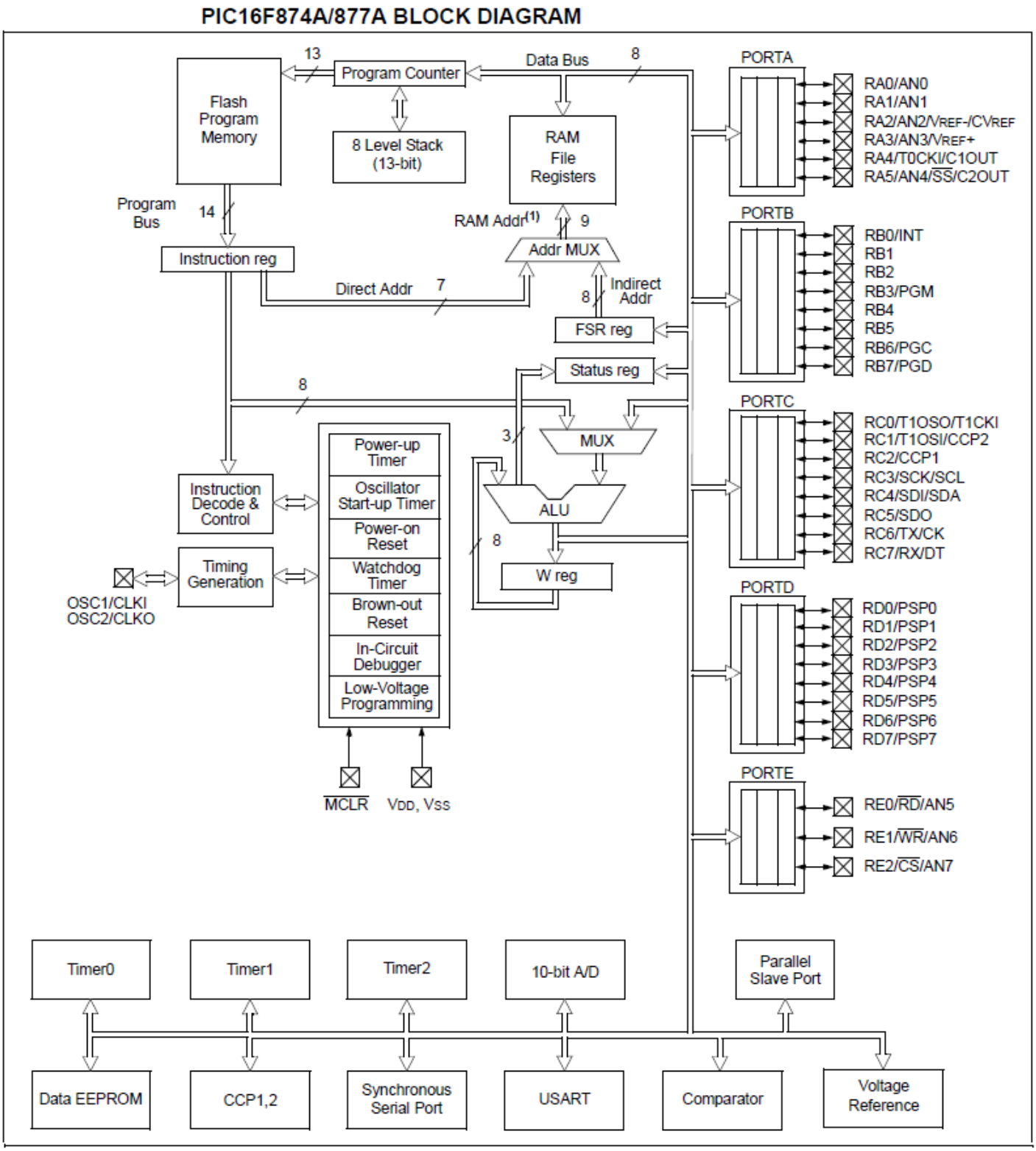


Figure 1 : Schéma de principe du PIC 16f87.

[D] Question « Complexité »

Considérez le programme de temporisation assembleur suivant :

```
            movlw       t0
            movwf       T
decrement   decf        T
            tstfsz      T
            goto        decrement
            movlw       0
            cpfseq      T
            goto        decrement
            return
```

Où t0 est un entier naturel > 2.

1. Réalisez un algorigramme qui décrit le fonctionnement de ce programme assembleur;
2. Quelle est la durée du programme, en nombre de cycles et en fonction de t0 ? On négligera le temps d'appel au programme.
3. Pour quelle valeur entière de t0 le programme a-t-il une durée la plus proche de 100 micro-secondes ?
4. Ce programme comporte des lignes surnuméraires, que l'on peut supprimer tout en conservant son caractère de temporisation. Barrez-les sur le sujet d'examen.
5. Une fois ces lignes supprimées, pour quelle nouvelle valeur entière de t0 le programme simplifié que vous obtenez a-t-il la durée la plus proche de 100 micro-secondes ?

Pour répondre aux questions, vous pouvez consulter le tableau suivant :

Instruction	Signification	Nombre de cycles
tstfsz	"test f, skip if 0"	Si l'instruction suivante n'est pas sautée : 1 cycle; Si l'instruction suivante est sautée : Si c'est une instruction sur un mot : 2 cycles; Si c'est une instruction sur deux mots : 3 cycles.
cpfseq	"Compare f with WREG, skip if 0"	Idem ci-dessus
movlw		1
movwf		1
decf	"decrement f"	1
goto		2
return		2

Problème :

La période T du TIMER0 est donnée par l'expression suivante :

$$T = 2^n * prescaler * T_{cy}$$

Où T_{cy} est la durée d'un cycle d'instruction, n le nombre de bits sur lesquels le Timer compte, et $prescaler$ un nombre entier.
Le tableau 1 donne les valeurs de T pour plusieurs valeurs de $prescaler$ (entre 1 et 256) et de n (8 et 16).

	8	*16*
1	2,56E-004	6,55E-002
2	5,12E-004	1,31E-001
4	1,02E-003	2,62E-001
8	2,05E-003	5,24E-001
16	4,10E-003	1,05E+000
32	8,19E-003	2,10E+000
64	1,64E-002	4,19E+000
128	3,28E-002	8,39E+000
256	6,55E-002	1,68E+001

Tableau 1 : valeurs de T en sec., en fonction du $prescaler$ et de *n*

Considérons le programme assembleur en page suivante.

On l'a créé pour suivre ce cahier des charges:
-on inverse l'état des LEDs associées aux première et deuxième broches du PORT B (RB0, et RB1) chaque seconde.
-pour compter pendant 1 seconde, on détecte le débordement du TIMER0, qui doit être correctement configuré. Une interruption est déclenchée à chaque débordement du TIMER0.

1) Dans le programme présenté en page suivante, repérez avec des accolades les étapes principales d'une interruption.
2) D'après le Tableau 1, Quelles sont les valeurs de $prescaler$ et de n qui permettent de respecter au mieux le cahier des charges ?
3) Le programme étant inséré dans le microcontrôleur et celui-ci mis sous tension, on se rend compte qu'aucune LED ne clignote. Quel type d'interruption a-t-on en fait autorisé ? Barrez les instructions fausses et réécrivez les instructions adéquates en face.
4) Une fois cette erreur corrigée, certaines LEDs clignotent, mais pas forcément RB0 et RB1. Quelles sont les LEDs qui clignotent ? Barrez l'instruction concernée et remplacez-là par une instruction qui répond au cahier des charges.

```
        LIST P=18F4520
        #include <P18F4520.inc>
        #include <CONFIG.inc>

;----- Déclaration de variables
        CBLOCK 0x00
                W_TEMP  : 1
                STATUS_TEMP : 1
                BSR_TEMP : 1
        ENDC


;----- Programme
        org     h'00'
        goto    init

        org     h'08'
        goto    routine_int

init    clrf    PORTB
        movlw   h'00'
        movwf   TRISB
        movlw   h'83'
        movwf   T0CON
        rcall   tmr0_init
        movlw   h'90'
        movwf   INTCON

boucle  nop
        goto    boucle

tmr0_init
        movlw   h'0B'
        movwf   TMR0H
        movlw   h'DB'
        movwf   TMR0L
        return

;----- Routine d'interruption
routine_int
        movwf   W_TEMP
        movff   STATUS, STATUS_TEMP
        movff   BSR, BSR_TEMP
        btfsc   INTCON,1
        rcall   tmr0_overflow
        movff   BSR_TEMP, BSR
        movff   W_TEMP, WREG
        movff   STATUS_TEMP, STATUS
        retfie

;----- Interruption de débordement du TIMER0
tmr0_overflow
        bcf     INTCON,1
        rcall   tmr0_init
        movlw   h'DB'
        xorwf   PORTB
        return

        END
```

Annexe

Dans tout l'énoncé, les caractéristiques du PIC utilisé sont telles que la période de l'horloge est TOSC= $0.25\ 10^{-6}$ sec. Il y a quatre coups d'horloge par cycle.

Voici quelques instructions et le nombre de cycles qui leur est associé :

movlw : 1 cycle

movwf : 1 cycle

goto : 2 cycles

return : 2 cycles

nop : 1 cycle

REGISTER 9-1: INTCON 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
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

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

Faculté des Sciences
Aix*Marseille Université

Année universitaire 2016/2017

Site : ☐ Luminy ☐ St-Charles ■ St Jérôme ☐ Cht-Gombert ☐ Aix-Montperrin ☐ Aubagne-SATIS

Sujet session de : ☐ 1er semestre - ■ 2ème semestre - ☐ Session 2

Durée de l'épreuve :1h30……..

Examen de : ☐L1/■L2/☐L3 - ☐M1/☐M2 - ☐ LP - ☐ DU Nom diplôme : L2 SPI

Code Apogée du module : SPI42ATJ Libellé du module : Informatique industrielle

Document autorisé : ■ OUI - ☐ NON (Poly de cours) Calculatrices autorisées : ■ OUI - ☐ NON

Jeudi 11 Mai 2017 8H30-10H00 *Licence 2 – SPI*

Aix Marseille Université

Informatique industrielle

Le polycopié de cours est autorisé.
Une calculatrice de type collège est autorisée.

Cet examen de 1 heure 30 est composé de 2 parties :

- *des questions de cours et exercices* (approximativement 50 mn),
- *un problème* (approximativement 30 mn).

Notes :

Vous rendrez votre sujet d'examen avec votre copie.

C'est sur le sujet que vous répondrez à une partie des questions.

Une annexe comporte des données utiles aux deux parties de l'examen.

Questions de cours

Attention

Vous êtes notés sur clarté de vos réponses. Pour avoir le maximum de points, elles devront être justes, courtes, et précises. Utilisez vos brouillons avant de vous jeter sur vos copies d'examen.

[A] Questions générales

1. Pour une instruction en assembleur, à quoi correspond l'opcode ? A quoi correspond l'opérande ? Donnez un exemple.
2. Quels sont les principaux critères de performance d'un microcontrôleur ? Comment ces critères ont-ils évolué ces dernières années ?
3. Considérons le mode d'adressage indirect: sur un schéma, représentez le mode de fonctionnement de ce mode d'adressage, en faisant apparaître le registre FSR, et la mémoire donnée avec ses 'pages', appelées 'Banks' en anglais.

[B] Questions « architecture »

1. Identifier les éléments ci-dessous sur le schéma bloc de la *Figure 1* :

   -le compteur programme,
   -le bus de données,
   -le registre File Select Register,
   -le registre d'instruction,
   -la mémoire programme,
   -le module Timer 0.

2. Le microcontrôleur présenté *Figure 1* est-il de structure *Von-Neumann* ou *Harvard* (justifiez votre choix)? Est-ce un microprocesseur 8 bits, 16 bits ou 32 bits ?
3. Combien de ports entrée-sortie comporte ce microcontrôleur ?

[C] Questions « codage binaire, hexadécimal et décimal »

1. Convertissez a=0x22 en binaire
2. Convertissez b=0x11 en binaire
3. En utilisant les questions 1 et 2, effectuez l'opération a-b en passant par le calcul du complément à deux de b. Donnez le résultat sous forme hexadécimale.

4. Effectuez l'opération *logique* (binaire) suivante : c xor w où c=1101 et w=1001 (xor : ou exclusif).
5. Effectuez l'opération *logique* (binaire) suivante : d & w où d=1110 et w=0111 (& : et logique).

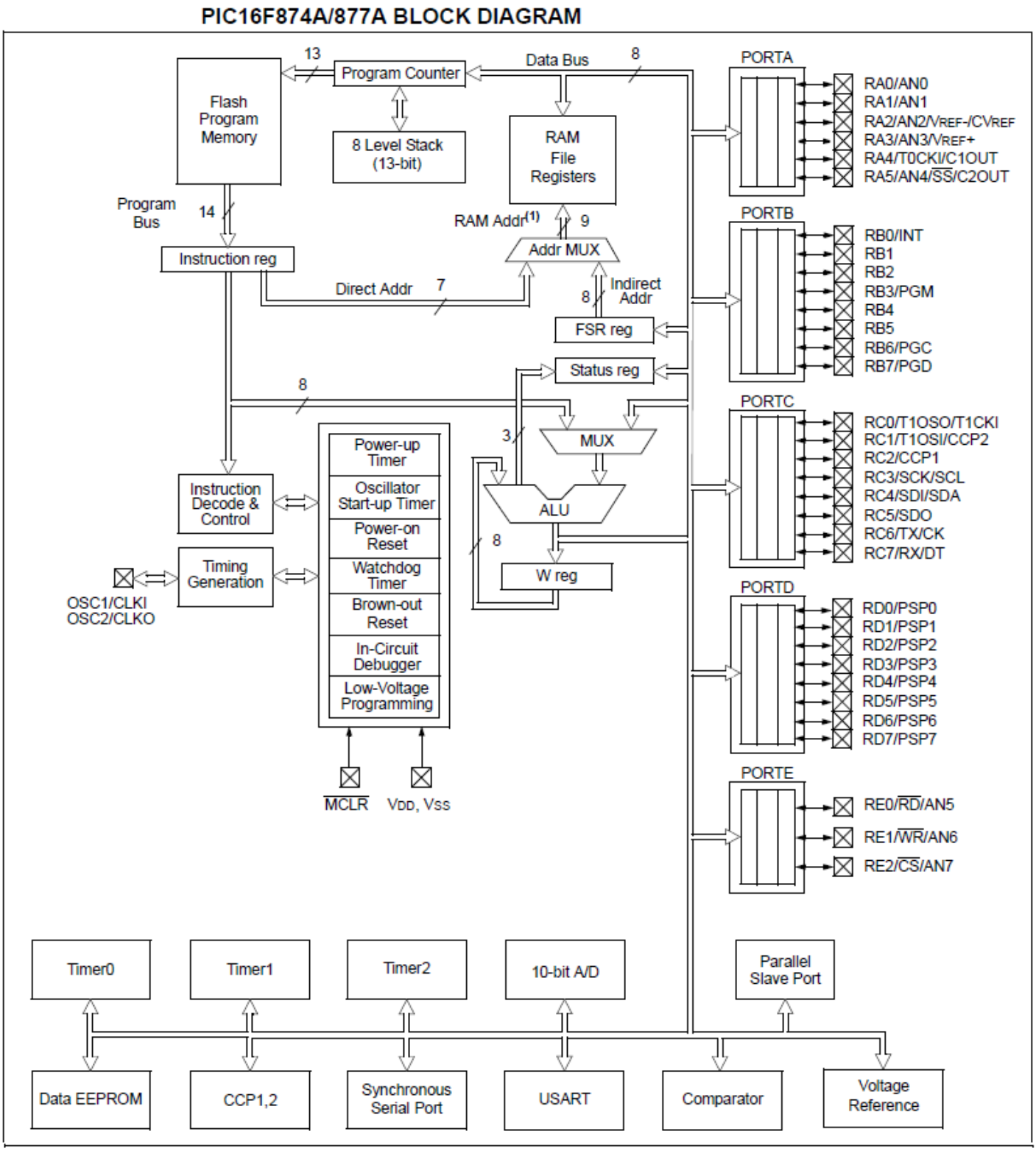


Figure 1 : Schéma de principe du PIC 16f87.

[D] Question « Algorigrammes et complexité numérique»

Le programme assembleur ci-dessous doit respecter le cahier des charges suivant:
il vise à allumer les quatre LEDs d'une carte de test successivement, à intervalle de temps régulier, de la première à la quatrième, et ceci à l'infini. Il contient un sous-programme de temporisation qui définit l'intervalle de temps.

```
main
        movlw d' _ _ '
        movwf PORTB
        call tempo

        movlw d' _ _ '
        movwf PORTB
        call tempo

        movlw d' _ _ '
        movwf PORTB
        call tempo

        movlw d' _ _ '
        movwf PORTB
        call tempo

        goto main


tempo
        movlw d'255'
        movwf t1
label1 decf t1
        tstfsz    t1
        cpfseq t1
        goto    label1
        return
```

Pour répondre aux questions qui suivent en page suivante, vous pouvez consulter le tableau suivant :

Instruction	Signification	Nombre de cycles
tstfsz	"test f, skip if 0"	Si l'instruction suivante n'est pas sautée : 1 cycle; Si l'instruction suivante est sautée : Si c'est une instruction sur un mot : 2 cycles; Si c'est une instruction sur deux mots : 3 cycles.
cpfseq	"Compare f with WREG, skip if 0"	Idem ci-dessus
decf	"decrement f"	1

Questions:

1. Complétez le programme aux emplacements prévus ' _ _ ' afin qu'il respecte le cahier des charges
2. Faites un algorigramme incluant le programme main et le sous-programme tempo
3. Il y a une ligne en trop dans le sous-programme tempo, rayez-là. Une fois cette ligne de code supprimée, quelle est la complexité du programme tempo en nombre de cycles ? Et donc, en seconde, la durée de l'intervalle de temps entre l'allumage successif des LEDs ?

Problème :

On utilise le logiciel MPLAB pour créer un programme assembleur, que l'on implante dans un microprocesseur PIC18F4520. Ce microprocesseur est ensuite placé sur une plaque de test comportant notamment des LEDs connectées au PORTB du microprocesseur.

Le programme est conçu selon ce *cahier des charges :*

On veut inverser l'état de la LED L1 connectée à la broche RB0 du PORTB, de façon à ce qu'elle reste à l'état haut pendant 1 seconde, et à l'état bas pendant 1 seconde.

1) Dans l'algorithme présenté pages suivantes, repérez et nommez les étapes principales d'une interruption (avec des accolades pour les étapes constituées de plusieurs lignes).

2) Quelle est la durée d'une itération de la boucle d'attente infinie du programme principal ? Servez-vous des données présentées en annexe pour répondre.

3) Quel registre faut-il considérer dans chaque cas pour :

   (a) configurer le TIMER0 (on/off, prescaler, etc.) ?

   (b) configurer et mettre en œuvre les interruptions (autorisation, etc.) ?

   Complétez le programme en conséquence aux emplacements désignés par _ _.

4) En faisant le test sur une plaquette PICDEM 2 PLUS, on constate que le programme ne fait pas ce qu'on voulait : qu'observe-t-on ? Pourquoi ? Que faudrait-il faire pour que le *cahier des charges* soit respecté ?

5) Dessinez deux algorigrammes succincts qui décrivent le fonctionnement du programme principal, et du programme d'interruption. Indiquez par un point d'exclamation l'endroit où le programme comporte une erreur.

```
; Programme Clignotant avec Interruption
        LIST P=18F4520
        #include <P18F4520.inc>
        #include <CONFIG.inc>

;----- Déclaration de variables
        CBLOCK 0x00
                W_TEMP  : 1
                STATUS_TEMP : 1
                BSR_TEMP : 1
                VARIABLEX : 1
        ENDC
;----- Programme principal
        org     h'0000'
        goto    init
        org     h'0008'
        goto    prog_int

init    clrf    PORTB       ;Remise à zéro du port B
        movlw   h'00'
        movwf   TRISB               ;Le port B est défini en sortie

        movlw   h'83'
        movwf   _ _             ;TIMER0 On, 16bits, Prescaler 16

        rcall   tmr0_init       ;Init TMR0 pour 1s pile
        movlw   h'A0'
        movwf   _ _               ;Autorisation Générale des interruptions et des interruptions du TIMER0.

        movlw   h'01'
        movwf   VARIABLEX

boucle
        incf    VARIABLEX
        decf    VARIABLEX
        goto    boucle          ;Boucle d'attente infinie
```

```
;----- Programme d'interruption
prog_int
        movwf  W_TEMP
        movff  STATUS, STATUS_TEMP
        movff  BSR, BSR_TEMP

        btfsc       _ _      ,2

        rcall    tmr0_overflow

        movff  BSR_TEMP, BSR
        movff  W_TEMP, WREG
        movff  STATUS_TEMP, STATUS
        retfie

;----- Init du registre TMR0
tmr0_init
        movlw  h'0B'
        movwf  TMR0H
        movlw  h'DB'
        movwf  TMR0L
        return

;----- Interruption de débordement du TIMER0
tmr0_overflow
        bcf         _ _      ,2
        rcall    tmr0_init
        movlw    h'08'
        xorwf  PORTB
        return

        END
```

Annexe

Dans tout le problème, les caractéristiques du PIC utilisé sont telles que la période de l'horloge est TOSC= 0.25 10^{-6} sec. Il y a quatre coups d'horloge par cycle.

Voici quelques instructions et le nombre de cycles qui leur est associé :

movlw : 1 cycle

movwf : 1 cycle

goto : 2 cycles

return : 2 cycles

nop : 1 cycle

call : 2 cycles

le descriptif complet des instructions incf et decf est donné ci-dessous:

INCF — **Increment f**

Syntax:	INCF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) + 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	0010 | 10da | ffff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `INCF CNT, 1, 0`

DECF — **Decrement f**

Syntax:	DECF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	(f) – 1 → dest
Status Affected:	C, DC, N, OV, Z
Encoding:	0000 | 01da | ffff | ffff
Description:	Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: `DECF CNT, 1, 0`

Le descriptif du registre INTCON est donné ci-dessous:

REGISTER 9-1: INTCON 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
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

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

Faculté des Sciences
Aix*Marseille Université

Année universitaire 2017/2018

Site : ☐ Luminy ☐ St-Charles ■ St Jérôme ☐ Cht-Gombert ☐Aix-Montperrin ☐ Aubagne-SATIS

Sujet session de : ☐ 1er semestre - ■ 2ème semestre - ☐ Session 2

Durée de l'épreuve :1h30……..

Examen de : ☐L1/■L2/☐L3 - ☐M1/☐M2 - ☐ LP - ☐ DU

Nom diplôme : L2 SPI

Code Apogée du module : SPI42ATA

Libellé du module : Informatique industrielle

Document autorisé : ■ OUI - ☐ NON (Poly de cours)

Calculatrices autorisées : ■ OUI - ☐ NON

Mardi 15 Mai 2018 10H45-12H15

Licence 2 – SPI

Aix Marseille Université

Informatique industrielle

Le polycopié de cours est autorisé.

Une calculatrice de type collège est autorisée.

Cet examen de 1 heure 30 est composé de 2 parties :

- *des questions de cours et exercices (14 points)*
- *un problème en langage Assembleur (6 points)*

Note :

Des extraits de la documentation technique nécessaires à cet examen vous sont donnés en annexe.

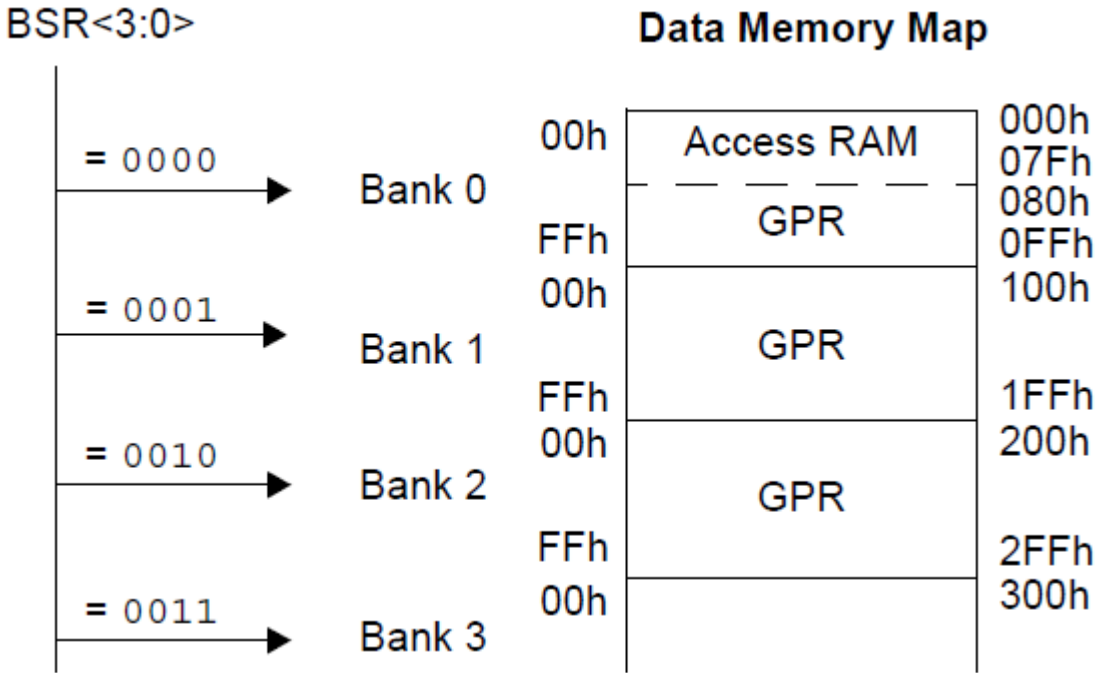
Questions de cours

Attention

*Vous êtes notés sur clarté de vos réponses. Pour avoir le maximum de points, elles devront être justes, **courtes**, et précises. Utilisez vos brouillons avant de vous jeter sur vos copies d'examen.*

[A] Questions générales

1. Lorsque l'on étudie le mode de fonctionnement du cerveau, on parle de 'potentiels d'action'. Quel est le parallèle que l'on peut faire entre les potentiels d'action et le terme de 'bit'que l'on utilise en informatique ?
2. Pour une instruction en assembleur, à quoi correspond l'opcode ? A quoi correspond l'opérande ? Donnez un exemple.
3. Cette question concerne les modes d'adressage : quel est le mode d'adressage utilisé quand on utilise l'adresse des données dans la RAM ? Quel est le rôle du registre BSR (voir figure ci-dessous) dans ce contexte ?

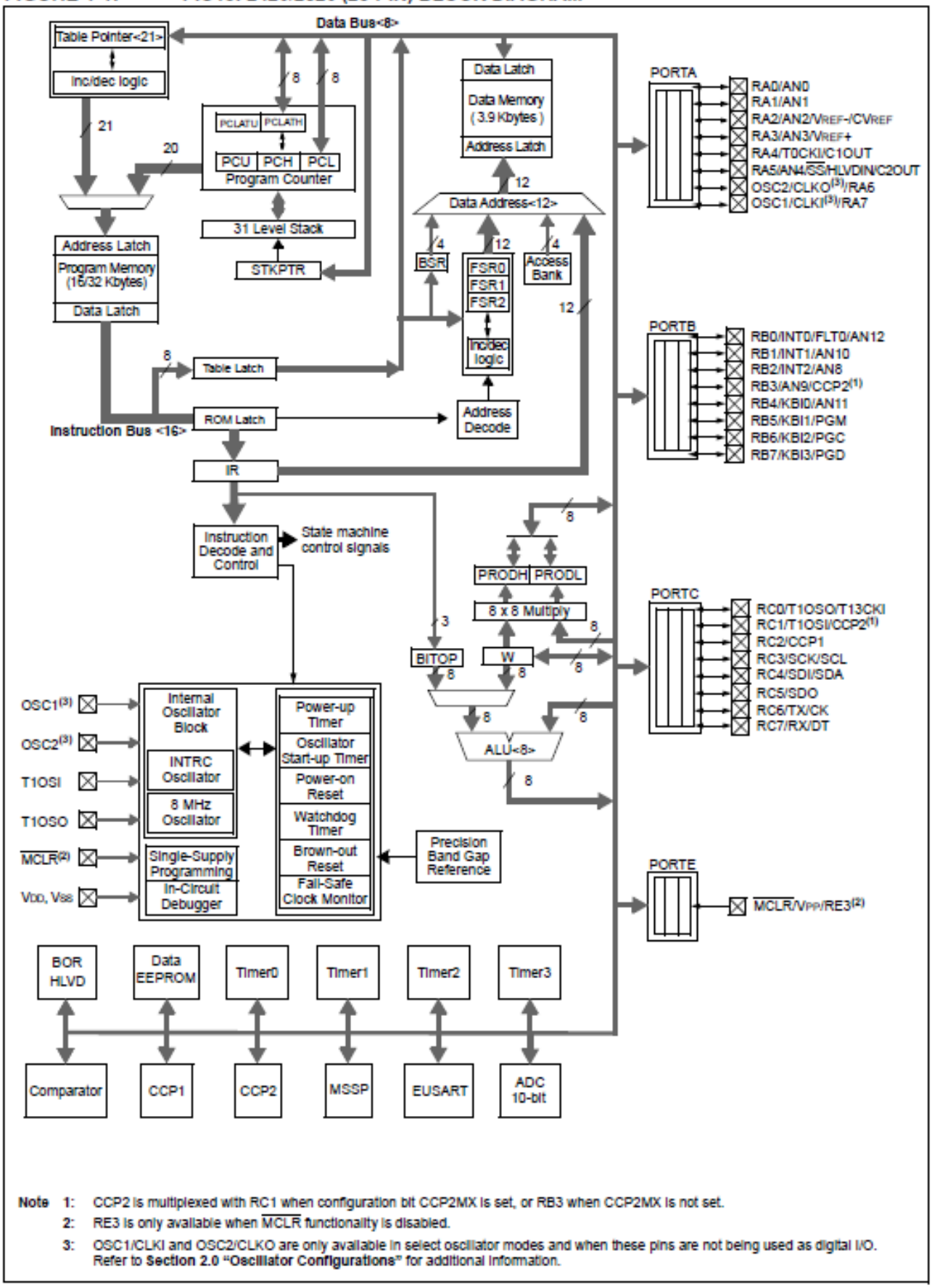


[B] Questions « architecture »

Identifier les éléments ci-dessous sur le schéma bloc de la figure en page suivante :

-le module de décodage et de contrôle,
-le module TIMER0,
-le bus d'instructions,
-l'unité arithmétique et logique,
-la mémoire programme,
-le port entrée-sortie A.

FIGURE 1-1: PIC18F2420/2520 (28-PIN) BLOCK DIAGRAM



Note 1: CCP2 is multiplexed with RC1 when configuration bit CCP2MX is set, or RB3 when CCP2MX is not set.

2: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

3: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

[C] Question « codage binaire, hexadécimal et décimal »

1. Convertissez a=0x4C en binaire
2. Convertissez b=0xBD en binaire

3. Effectuez l'opération *logique* (binaire) suivante : a xor b (xor : ou exclusif).

[D] Question « Temporisation »

Les programmes 1 et 2 sont deux programmes de temporisation. L'annexe contient des informations utiles à la résolution de cet exercice.

On considère le programme 1 suivant, où t10 est une valeur fixe 'initiale' de t1, telle que t10>1 :

```
        movlw   t10
        movwf   t1
comp1   decf    t1
        movlw   0
        cpfseq  t1
        goto    comp1
        return
```

On considère le programme 2 suivant, où t20 est une valeur fixe 'initiale' de t2, telle que t20>1 :

```
        movlw   t20
        movwf   t2
comp2   dcfsnz  t2
        return
        goto comp2
```

1. Dessinez les algorigrammes correspondant à ces deux programmes.
2. Que peut-on dire de ces deux algorigrammes ?
3. Quel est le nombre de cycles du programme 1, en fonction de t10 ? On négligera le temps d'appel à ce programme.
4. Quel est le nombre de cycles du programme 2, en fonction de t20 ? On négligera le temps d'appel à ce programme.

5. Existe-t-il des valeurs entières de t10 et t20 pour lesquelles les durées des programmes 1 et 2 sont égales ? Si oui, quelles sont ces valeurs, et la durée correspondante en micro-secondes ?

Problème

De même qu'un programme de temporisation, il existe un moyen de compter le temps qui s'écoule avec un microcontrôleur. Il s'agit des interruptions de débordement du TIMER0. L'annexe contient des informations utiles à la résolution du problème.

Dans ce problème on souhaite configurer le TIMER0 pour qu'une interruption se produise toutes les 1 sec, pour changer l'état de la LED RB0, connectée à la première broche du PORTB.

La période du TIMER0 est donnée par l'expression suivante :

$$T = 2^n * prescaler * T_{cy}$$

Où T_{cy} est la durée d'un cycle d'instruction, n le nombre de bits sur lesquels le timer compte.

Le tableau 1 donne les valeurs de T pour plusieurs valeurs de prescaler et de n

	8	*16*
1	2,56E-004	6,55E-002
2	5,12E-004	1,31E-001
4	1,02E-003	2,62E-001
8	2,05E-003	5,24E-001
16	4,10E-003	1,05E+000
32	8,19E-003	2,10E+000
64	1,64E-002	4,19E+000
128	3,28E-002	8,39E+000
256	6,55E-002	1,68E+001

Tableau 1 : valeurs de T, en fonction du prescaler et de *n*

1) Quelle est la valeur de T_{cy} ?

2) Quel registre faut-il considérer dans chaque cas pour :
   (a) configurer le TIMER0 (on/off, prescaler, etc.) ?
   (b) configurer et mettre en œuvre les interruptions (autorisation, etc.) ?
   Complétez les extraits de programme page suivante en conséquence aux emplacements désignés par _ _. <u>Vous rendrez cette page avec votre copie d'examen.</u>

3) Dans les extraits de programme présentés page suivante, repérez avec des accolades les étapes principales d'une interruption.
4) Ce programme comporte une erreur: barrez la ligne de code correspondante et réécrivez une ligne de code qui permet de respecter les spécifications.

```
        ;;;;;;;;;;;;;;;;;;; Programme principal
init    clrf    PORTB;Remise à zéro du port B
        movlw   h'00'
        movwf   TRISB           ;Le port B est défini en sortie

        movlw   h'83'
        movwf   _ _             ;TIMER0 On, 16bits, Prescaler 16

        rcall   tmr0_init       ;Init TMR0 pour 1s pile
        movlw   h'A0'
        movwf   _ _             ;Autorisation des interruptions

boucle
        nop
        goto    boucle          ;Boucle d'attente infinie

;;;;;;;;;;;;;;;;;;; Programme d'interruption
prog_int
        movwf   W_TEMP
        movff   STATUS, STATUS_TEMP
        movff   BSR, BSR_TEMP

        btfsc       _ _     ,2

        rcall   tmr0_overflow

        movff   BSR_TEMP, BSR
        movff   W_TEMP, WREG
        movff   STATUS_TEMP, STATUS
        retfie

tmr0_overflow
        bcf         _ _     ,2
        movlw   h'0B'
        movwf   TMR0H
        movlw   h'DB'
        movwf   TMR0L
        movlw    h'02'
        xorwf   PORTB
        return
```

Annexe

Dans tout l'énoncé, les caractéristiques du PIC utilisé sont telles que la période de l'horloge est TOSC= $0.25\ 10^{-6}$ sec. Il y a quatre coups d'horloge par cycle.

Voici quelques instructions et le nombre de cycles qui leur est associé :

movlw : 1 cycle

movwf : 1 cycle

goto : 2 cycles

return : 2 cycles

nop : 1 cycle

dcfsnz : 1 cycle si on ne saute pas, 2 cycles si on saute une instruction sur un mot, 3 cycles si on saute une instruction sur 2 mots.

decf : 1 cycle

cpfseq : 1 cycle si on ne saute pas, 2 cycles si on saute une instruction sur un mot, 3 cycles si on saute une instruction sur 2 mots.

Le descriptif complet des instructions cpfseq, decf, et dcfsnz est donné ci-dessous:

DCFSNZ	Decrement f, skip if not 0
Syntax:	DCFSNZ f {,d {,a}}
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – 1 → dest, skip if result ≠ 0
Status Affected:	None
Encoding:	0100 11da ffff ffff
Description:	The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

CPFSEQ — Compare f with W, skip if f = W

Syntax: CPFSEQ f {,a}

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?
After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

DECF — Decrement f

Syntax: DECF f {,d {,a}}

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (f) – 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0000	01da	ffff	ffff

Description: Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank (default).
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example:
```
DECF    CNT,  1, 0
```

Before Instruction
CNT = 01h
Z = 0
After Instruction
CNT = 00h
Z = 1