Faculté des Sciences
Aix*Marseille Université

Année universitaire 2017/2018

janvier 2018 *Licence SPI AGE – Semestre 5*

Aix-Marseille Université

Contrôle des connaissances

Automatisme et informatique industrielle

CORRIGE

Documents: seul le polycopié de cours est autorisé
Calculatrices autorisées

Cet examen de 2 heures est composé de 2 parties :

- partie *'Informatique industrielle'* (10 points / approximativement 60min).
- partie *'Automatisme'* (8 points / approximativement 45min),

La *présentation* et le soin apporté à la rédaction de façon générale comptent pour 2 points.

Partie Informatique industrielle

Questions de cours (3 pts)

Question 1:

Le vecteur RESET pointe vers la case mémoire contenant la première ligne du programme à exécuter. On tombe sur le vecteur RESET à la mise sous tension ou en appuyant sur le bouton RESET par exemple.

Question 2: Le vecteur d'INTERRUPTION placé à l'adresse 0x0008 pointe vers la première ligne du programme à exécuter en cas d'interruption de priorité haute.

Question 3: Pour un PIC18F4520 : 8*16^3 = 32000 cases.
Pour un PIC18F4420 : 4*16^3 = 16000 cases .

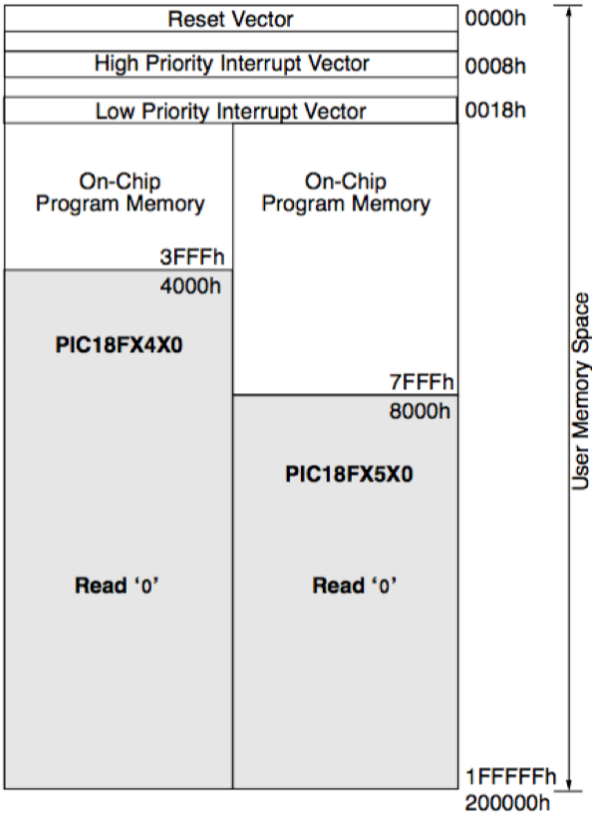

Figure 1 : *organisation de la mémoire programme du PIC 18F4520.*

Exercice 1 : programmation C (3 pts)

On souhaite configurer de façon simplifiée une interruption de débordement du TIMER0. Si le bit `TMR0ON` est à 1, il faut donner à `TMR0H` et `TMR0L` les valeurs `0x0B` et `0xDB` respectivement.

```
//------------------------Debut du programme-----------------------
int TMR0ON = 1;
void main{


int TMR0H = 0x00;
int TMR0L = 0x00;


int* p_TMR0H = &TMR0H;
int* p_TMR0L = &TMR0L;


Config_Timer0(p_TMR0H, p_TMR0L, TMR0ON);
printf(TMR0H);
printf(TMR0L);
}
void Config_Timer0(int* Valeur1, int* Valeur2, int Valeur3) {


if (Valeur3 ==1)
{
*Valeur1 = 0x0B ;
*Valeur2 = 0xDB ;
}
}
//------------------------Fin du programme-----------------------
```

Ce programme affiche deux valeurs qui sont `TMR0H` et `TMR0L`, après l'appel de la fonction `Config_Timer0`.

Question 1: Variable globale dans ce programme : `TMR0ON`

Variables locales dans ce programme : `TMR0H et TMR0L`

Question 2: Les valeurs de `TMR0H` et `TMR0L` affichées par printf sont 0 et 0 : ces valeurs sont modifiées dans la fonction `Config_Timer0` mais la modification n'est pas prise en compte en dehors de la fonction `Config_Timer0`.

Question 3: pour que la modification de `TMR0H` et `TMR0L` soit prise en compte dans le programme principal : il faut passer par des pointeurs (voir modifications en rouge).

Exercice 2 : Programmation d'un microcontrôleur en C (4 pts)

Considérons les fonctions suivantes, qui s'inspirent du programme chronomètre vu en TP :

La fonction bouton_appui, qui boucle tant qu'on n'a pas appuyé sur un bouton poussoir puis levé le doigt du bouton; la fonction d'initialisation du vecteur d'interruption de priorité haute, et la fonction TMR0_overflow qui est exécutée lorsqu'il y a débordement du TIMER0, qui permet de gérer l'incrément du chronomètre et son affichage. Ces fonctions sont présentées dans la page ci-après.
La valeur de MSB est 0x0B, la valeur de LSB est 0xDB.

Question 1: Ce programme comporte plusieurs erreurs. Sur l'énoncé, barrez les lignes correspondantes, et si nécessaire réécrivez une solution correcte, en justifiant brièvement votre choix.

Question 2 : La période *T* du Timer0 est définie de la façon suivante:

$T=T_{Cy} * (max-min) * Prescaler$

où $T_{Cy} = 10^{-6}$ sec. *Prescaler* vaut 16, et *max* vaut $(65535)_d$.

Sachant que $(0BDB)_h = (3035)_d$ quel est le temps écoulé entre chaque interruption de débordement du Timer0 ?

Question 3 : Faites un algorigramme pour la fonction TMR0_overflow.

Fonction d'appui sur bouton

```
void bouton_appui (void)
{
    // Tant que PORTXbits_bouton=1 (ie. bouton poussoir non actionne), on boucle...
    while( ! PORTXbits_bouton);

    // Tant que PORTXbits_bouton=0 (ie. bouton poussoir actionne), on boucle...
    while(PORTXbits_bouton);
}
```

```
While(PORTXbits_bouton) ;

While( ! PORTXbits_bouton) ;
```

Vecteur d'interruption et routine d'interruption

```
#pragma code Vecteur_interruption_priorite_haute = 0x256
// Directive pour spécifier l'adresse memoire ou debute la fonction

void Vecteur_interruption_priorite_haute (void)
{
  // va a la routine de gestion des interruptions de priorite haute
  _asm
    goto TMR0_overflow // instruction assembleur
  _endasm
}
#pragma code

#pragma interrupt TMR0_overflow
void TMR0_overflow(void)
{
    // RAZ flag it venant de TMR0
    INTCONbits.TMR0IF    = 1;

    // RAZ TIMER 0
    TMR0H = MSB;
    TMR0L = LSB;

    // Increment d'une seconde et ajustement des minutes
    SEC ++;

    if(SEC == 65535){
        MIN ++;
        SEC=0;
    }

    lcd_affiche_chrono();
}
```

0X0008 au lieu de 0x256

```
INTCONbits.TMR0IF=0 ; // RAZ flag
```

if (SEC == 60) Quand on arrive à 60 secondes on incrémente les minutes

PARTIE Automatismes

Exercice 1 (4 pts)

On considère la photographie en *figure 2*. Il s'agit de la sonde Voyager 2. Son composant le plus proéminent est son antenne de 3.7 mètres de diamètre.

Figure 2: sonde Voyager 2

Les principaux instruments de mesure de Voyager 2 sont les suivants

ISS	Caméras
IRIS	Spectromètre infrarouge
UVS	Spectromètre ultraviolet
CRS	Analyse rayons cosmiques
LECP	Particules à faible énergie

Ces instruments sont montés sur une plateforme orientable, ce qui leur permet de viser une cible particulière. Par exemple, l'étude de Titan, un satellite de Saturne, a été un objectif majeur de Voyager 2.

Question 1: Dans ce système :
-la partie opérative avec:
-actionneurs : la plateforme orientable

-capteurs : les caméras, spectromètres et analyseurs

- la partie relation : l’antenne

Question 2: De quoi peut être constituée la partie commande ?
Un microcontrôleur ou un automate qui va coder les données reçues par les instruments de mesure et les envoyer vers la Terre.

Exercice 2 (4 pts)

Cet exercice est fondé sur la maquette de domotique vue en TP. On s'intéresse à la boîte 'chauffage'. Elle comporte une ampoule chauffante qui peut être alimentée par le biais d'une PWM (pulse width modulation, c’est-à-dire modulation de largeur d'impulsion), de rapport cyclique alpha. Ce coefficient est à valeurs entre 0 et 1.

La partie commande de la maquette de domotique est un automate crouzet. Il travaille sur des paquets de données de 10 bits, qui codent des valeurs entières (notées ici par exemple NUM).

Question 1:

Le programme présenté en figure 3 ne permet pas d'utiliser le mode d'alimentation PWM. Il faut placer un bloc XA PWM sur OF XA pour cela, en le reliant à un NUM avec une valeur souhaitée entre 0 et 1023.

Question 2:

Trigger : les ordonnées ne prennent pas la même valeur quand les abscisses augmentent et quand les abscisses diminuent. La sortie du trigger est connectée à O3, le ON/OFF du gradateur.

Question 3:

NUM = alpha * 1023

Question 4:

On a pu utiliser par exemple : Première montée alpha =0.6 et NUM=environ 600

Deuxième montée alpha =1 et NUM=1023.

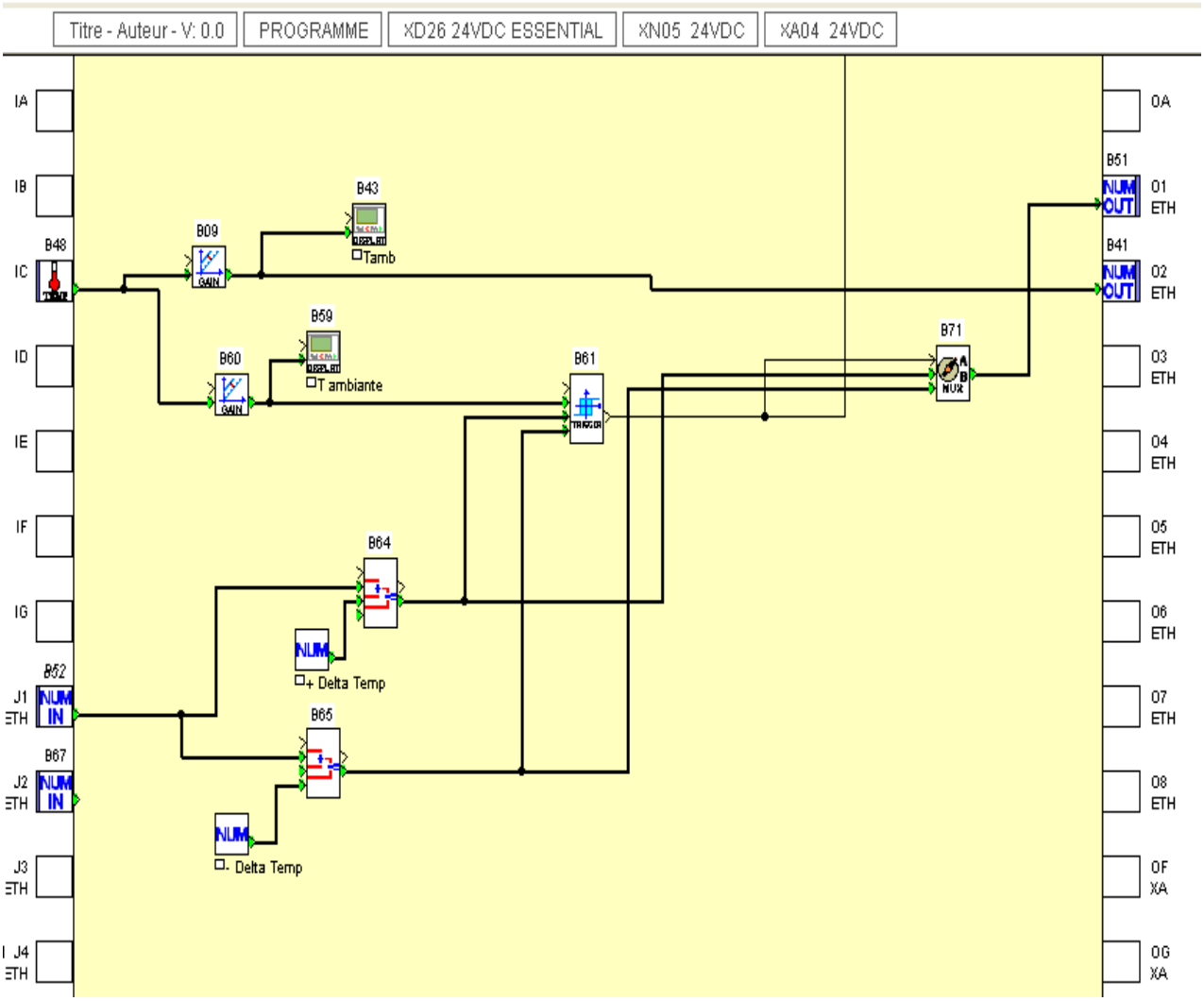

Figure 3: programme FBD pour régulation TOR

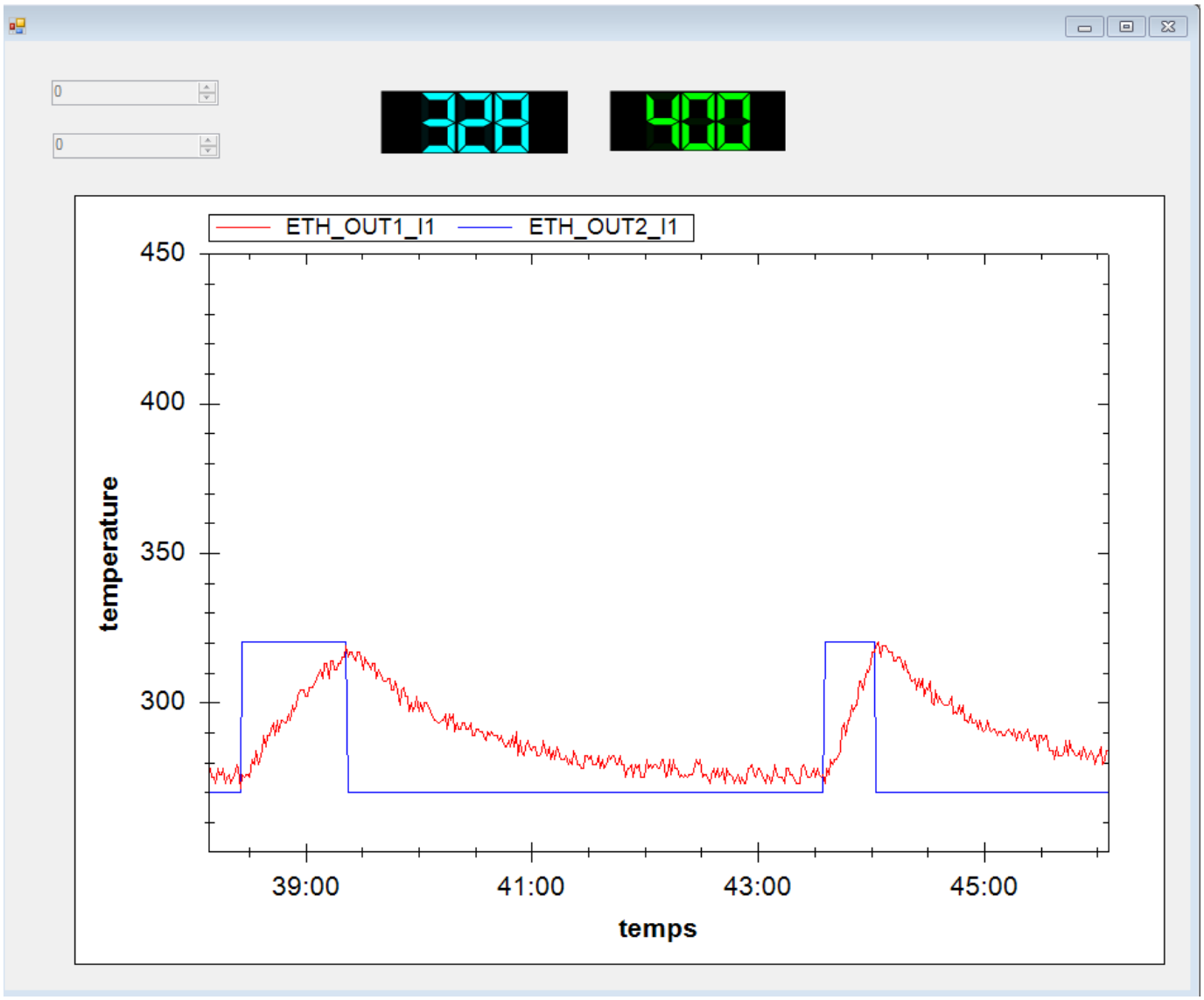

Figure 4: deux montées en température