	Institut Supérieur des Etudes Technologiques de Sousse **Département Génie Electrique** Examen	***Année universitaire*** *: 2017/2018* ***Semestre*** *: 1* ***Date :*** *Jan 2018* ***Durée :*** *1h30mn*
Matière *: Système Temps Réel*		***Classes*** *: AII3*
Documents *: Non autorisés*	***Enseignant*** *: Ali HMIDENE*	***Nb. Pages*** *: 2 + 1 Annexe*

Problème (12 points)

Dans l'industrie textile, l'ourdissage est une opération préparatoire qui consiste à enrouler, dans un ordre déterminé, un certain nombre de fils d'égale longueur sur une **ensouple** pour former la chaîne destinée à alimenter le métier à tisser. La figure 1 présente un modèle simplifié d'une machine à ourdir.

L'action sur la pédale rhéostat commande la rotation de l'ensouple d'ourdissage à vitesse variable.

Le codeur incrémental est utilisé pour mesurer le métrage des fils ourdis.

Ce système d'ourdissage est commandé par une carte électronique à base d'un microcontrôleur PIC18F4520 (figure2).

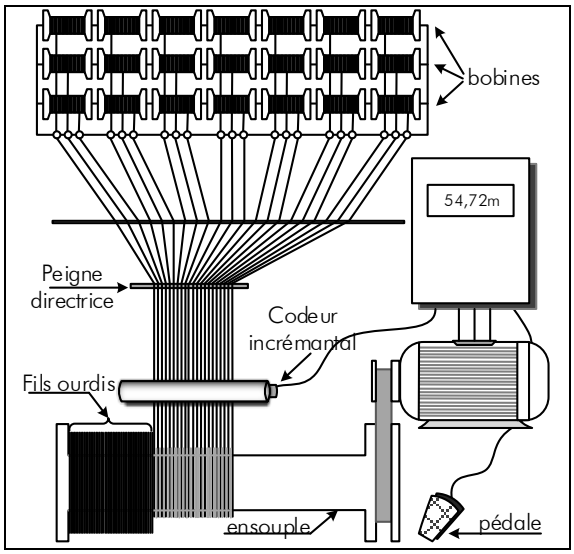


Figure 1

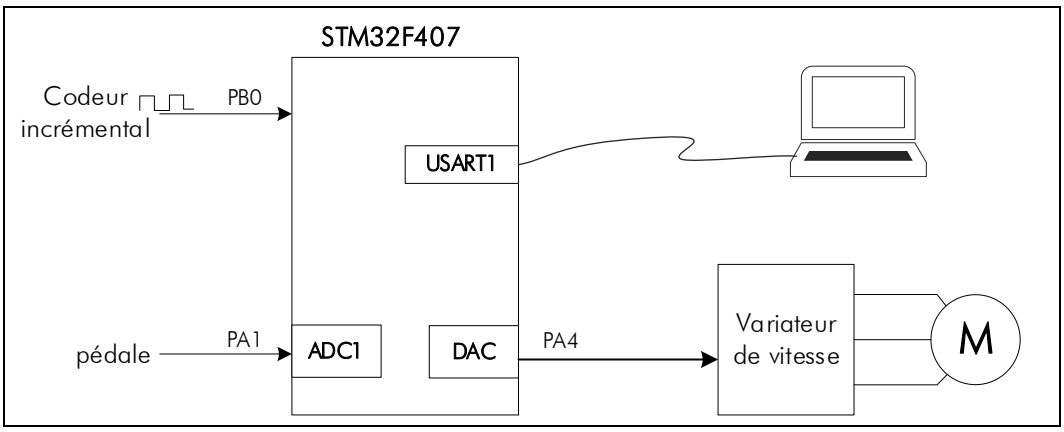


Figure 2

Le signal analogique provenant de la pédale est appliqué à l'entrée PA1 (IN1) du convertisseur analogique numérique ADC. La valeur convertie est appliquée au convertisseur numérique analogique DAC.

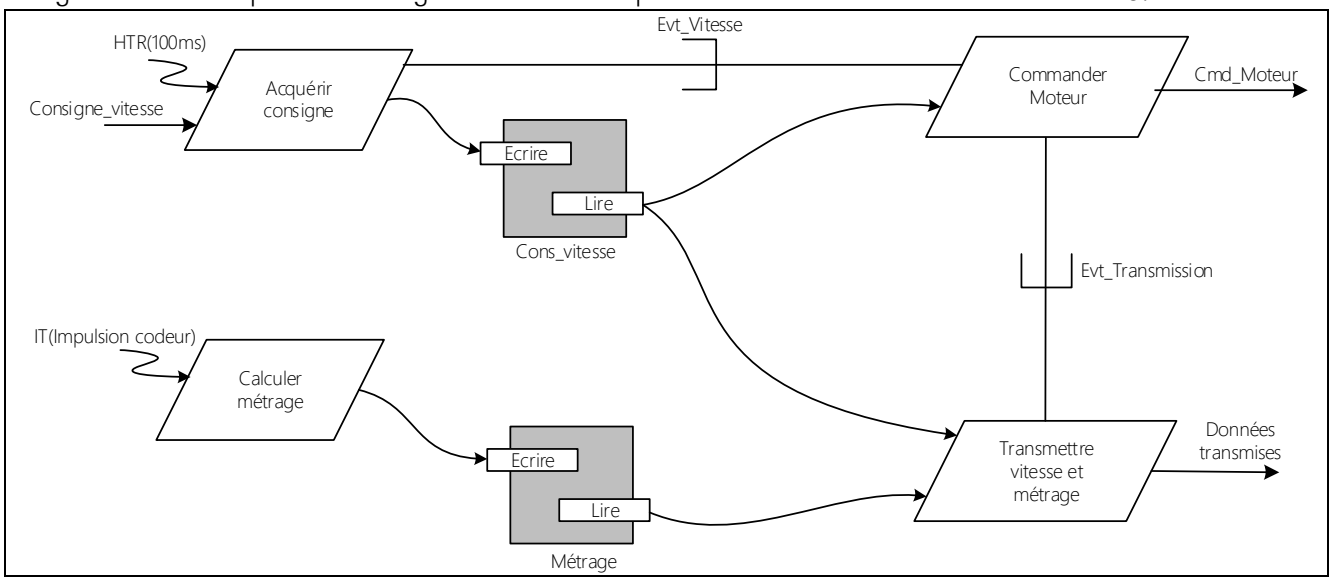
Le codeur incrémental est entrainé en rotation par le déplacement des fils. Il dévire alors un train d'impulsions dont le nombre permet de calculer la longueur des fils ourdis. Pour simplifier l'étude, on admet qu'à chaque impulsion (front montant) le métrage du fils augmente de 0,01m. La valeur du métrage est envoyée à un PC distant pour archivage.

Le système d'ourdissage peut être décomposé comme suit :

- La tâche Acquisition : convertie la tension appliquée à l'entrée PA1 toutes les 500 millisecondes.
- La tâche Commander Moteur : génère la tension de commande du variateur de vitesse.
- La tâche Calculer Métrage (ISR) : ajoute à la variable **Metrage** la valeur 0,01m. Cette tâche est activée à chaque front montant du signal généré par l'encodeur.

- La tâche transmission : envoie au format ASCII la vitesse du moteur et le métrage des fils ourdis au PC via une liaison série RS232.

La figure suivante représente le digramme de conception multitâche selon la méthode DARTS.



Pour optimiser le traitement des tâches Affichage et Transmission, l'affichage et la transmission du métrage n'aura lieu que lorsque le moteur est en rotation.

Q1. Compléter le programme proposé.

Q2. Peut-on remplacer la variable globale **Cons_Vitesse** par une boite aux lettres ? si oui comment peut on synchroniser la tache Transmission.

Exercice (8 points)

On considère une configuration **T** de trois tâches périodiques et indépendantes à échéance sur requête. Les tâches sont définies par les paramètres temporels suivants :

T1 (r0= 0, C1 = 5, P1 = 12)
T2 (r0= 0, C2 = 2, P2 = 6)
T3 (r0= 0, C3 = 5, P3 = 24)

On applique à la configuration T un ordonnancement préemptif à priorité statique selon la plus petite période (RM).

1. Donner le facteur d'utilisation U et conclure sur l'ordonnancement par RM en utilisant le test d'ordonnancement.
2. Donner la valeur de la période d'étude et tracer la séquence d'exécution correspondante.

On applique à la configuration T un ordonnancement préemptif à priorité dynamique selon l'échéance la plus proche (EDF).

3. Etant donné le facteur d'utilisation U, est-ce que le test d'ordonnancement nous permet de conclure sur la faisabilité de l'ordonnancement par EDF?
4. Tracer la séquence d'exécution et identifier les temps creux (libre).

ANNEXE

1. Création d'une tâche

```
BaseType_t xTaskCreate(     TaskFunction_t pvTaskCode,
                            const char * const pcName,
                            unsigned short usStackDepth,
                            void *pvParameters,
                            UBaseType_t uxPriority,
                            TaskHandle_t *pxCreatedTask
                          );
```

2. vTaskDelay

```
void vTaskDelay(const TickType_t  xTicksToDelay );
```

on peut diviser la paramètre par **portTICK_PERIOD_MS** pour avoir le temps en ms.

3. Sémaphore binaire

Création de sémaphore

`SemaphoreHandle_t xSemaphoreCreateBinary( void );` : renvoie l'identifiant du sémaphore binaire. Le sémaphore est créé à l'état 'vide', ce qui signifie que le sémaphore doit d'abord être libéré en utilisant la fonction API xSemaphoreGive ().

Libérer le sémaphore

```
xSemaphoreGive( SemaphoreHandle_t xSemaphore );
```

Prendre le sémaphore

```
xSemaphoreTake( SemaphoreHandle_t xSemaphore,  xTicksToWait );
```

:

`xTicksToWait` prend `portMAX_DELAY` pour une attente infinie.

4. Conversion analogique Numérique

BusyADC() : En cours de conversion

ConvertADC() : démarrer la conversion

int ReadADC() : renvoie le résultat de conversion

SetChanADC(canal) : sélectionner le canal de conversion

5. Communication série asynchrone

putcUSART(char) : envoyer un caractère

putsUSART(String) : envoyer une chaine de caractère (en ASCII)

6. Conversion des données dans une chaine de caractère ASCII

```
Sprintf(String, format, data) ;
```

String : chaine de caractère

format : s'écrit " %flag ", flag prend *u* pour entier non signé et *f* pour le type float.

Data : donnée à convertir en ASCII.

7. Condition suffisante pour l'ordonnançabilité RM

$U \leq n(2^{\frac{1}{n}} - 1)$: n : nombre de tâches