	Institut Supérieur des Etudes Technologiques de Sousse **Département Génie Electrique** Examen	***Année universitaire*** *: 2015/2016* ***Semestre*** *: 2* ***Date :*** ***24*** *Mai 2016*
Matière *: Digital Signal Controller*	***Classes*** *: Master EASER*	***Durée :*** *1h30mn*
Documents *: Non autorisés*	***Enseignant*** *: Ali HMIDENE*	***Nb. Pages*** *: 2 + 7 Annexes*

Problème

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** (grande bobine) pour former la chaîne destinée à alimenter le métier à tisser.

Le système comporte principalement :

- un moteur à courant continue pour l'entrainement de l'ensouple.
- Une pédale rhéostat pour la commande du moteur à vitesse variable.
- Un interrupteur (S1) Marche/Arrêt.
- Un contact (S2) de détection de la rupture du fil (interrupteur casse fil)
- Deux voyants pour la signalisation de la présence de l'alimentation (LED1) et du casse fil (LED2).

Ce système d'ourdissage est commandé par une carte électronique à base d'un microcontrôleur STM32F207 (figure1)

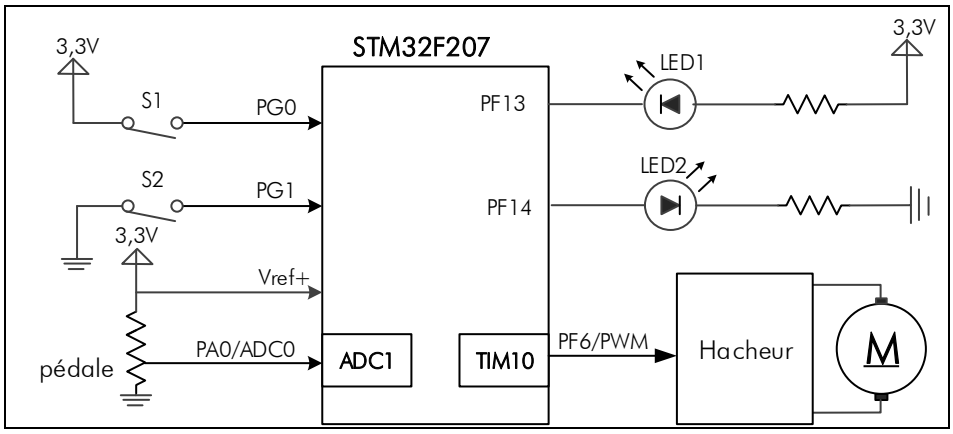

Figure 1

PARTIE 1 – GPIO –

Les entrées/sorties logiques sont connectées au microcontrôleur selon le schéma de la figure1.

1. Donner les niveaux logiques en PG0 et PG1 dans les cas suivants :
   a. S1 ouvert et S2 fermé.
   b. S1 fermé et S2 ouvert.
2. Doit-on utiliser des résistances de tirage ? préciser le type de résistance (pull-up ou pull-down) à utiliser s'il est nécessaire.
3. Quels sont les niveaux logiques à fournir aux sorties PF13 et PF14 pour allumer les LEDs ?
4. Ecrire le pseudocode de configuration des lignes PG0, PG1, PF13 et PF14.
5. Ecrire le pseudocode de commande des LEDs (LED1 et LED2) selon les états des entrées S1 et S2.

PARTIE – ADC –

La pédale est équivalente à un rhéostat dont le curseur est connecté à l'entrée PA0 (canal 0) du convertisseur analogique numérique ADC1. La valeur convertie correspond au rapport cyclique du signal de commande du moteur d'entrainement.

6. Sur combien de bits doit-on coder le signal à convertir pour garantir un pas de quantification inférieur à 3,5mV ?
7. Pour les paramètres suivants :
   - $R_{AIN} = 5K\Omega$
   - $R_{ADC} = 2K\Omega$
   - $C_{ADC} = 4pf$
   - $F_{ADC} = 10MHz$
   - Fréquence du bus APB2 = 60MHz.

   Calculer le temps d'acquisition Ts.
8. Ecrire le pseudocode de la configuration de la broche PA0 en analogique.
9. Ecrire le code de la procédure **Init_ADC1()** qui initialise le convertisseur ADC1 en mode simple conversion sur un seul canal ($F_{ADC} = 10Mhz$ et $T_S = 15cycles$).
10. Ecrire le pseudocode de l'acquisition du signal provenant de la pédale et de sauvegarder la valeur convertie dans la variable globale **Rapport_Cyclique**.

PARTIE – TIM10 –

Le timer TIM10 est utilisé pour générer le signal PWM de fréquence 10kHz. La fréquence d'incrémentation du Timer TIM10 est 120MHz.

11. Déterminer les valeurs à charger dans les registres PSC, ARR.
12. Calculer la valeur à charge dans le registre CCR1 pour commander le moteur avec un rapport cyclique de 75%.

```
typedef struct
{
  __IO uint32_t MODER;   /* GPIO port mode register,              Address offset: 0x00*/
  __IO uint32_t OTYPER;  /* GPIO port output type register,       Address offset: 0x04*/
  __IO uint32_t OSPEEDR; /* GPIO port output speed register,      Address offset: 0x08*/
  __IO uint32_t PUPDR;   /* GPIO port pull-up/pull-down register, Address offset: 0x0C*/
  __IO uint32_t IDR;     /* GPIO port input data register,        Address offset: 0x10*/
  __IO uint32_t ODR;     /* GPIO port output data register,       Address offset: 0x14*/
  __IO uint16_t BSRRL;   /* GPIO port bit set/reset low register, Address offset: 0x18*/
  __IO uint16_t BSRRH;   /* GPIO port bit set/reset high register,Address offset: 0x1A*/
  __IO uint32_t LCKR;    /* GPIO port configuration lock register,Address offset: 0x1C*/
  __IO uint32_t AFR[2]; /* GPIO alternate function registers,Address offset: 0x24-0x28*/
} GPIO_TypeDef;

typedef enum
{
  GPIO_Mode_IN   = 0x00, /*!< GPIO Input Mode */
  GPIO_Mode_OUT  = 0x01, /*!< GPIO Output Mode */
  GPIO_Mode_AF   = 0x02, /*!< GPIO Alternate function Mode */
  GPIO_Mode_AN   = 0x03  /*!< GPIO Analog Mode */
}GPIOMode_TypeDef;



typedef enum
{
  GPIO_OType_PP = 0x00,
  GPIO_OType_OD = 0x01
}GPIOOType_TypeDef;


typedef enum
{
  GPIO_Speed_2MHz   = 0x00, /*!< Low speed */
  GPIO_Speed_25MHz  = 0x01, /*!< Medium speed */
  GPIO_Speed_50MHz  = 0x02, /*!< Fast speed */
  GPIO_Speed_100MHz = 0x03  /*!< High speed */
}GPIOSpeed_TypeDef;

typedef enum
{
  GPIO_PuPd_NOPULL = 0x00,
  GPIO_PuPd_UP     = 0x01,
  GPIO_PuPd_DOWN   = 0x02
}GPIOPuPd_TypeDef;

typedef enum
{
  Bit_RESET = 0,
  Bit_SET
}BitAction;

typedef struct
{
  uint32_t GPIO_Pin;              // GPIO_Pin_x or GPIO_Pin_All
  GPIOMode_TypeDef GPIO_Mode;
  GPIOSpeed_TypeDef GPIO_Speed;
  GPIOOType_TypeDef GPIO_OType;
  GPIOPuPd_TypeDef GPIO_PuPd;
}GPIO_InitTypeDef;
```

```
typedef enum {RESET = 0, SET = !RESET} FlagStatus, ITStatus;
typedef enum {DISABLE = 0, ENABLE = !DISABLE} FunctionalState;
```

Les fonctions de manipulation des GPIOs

Lecture d'un port configuré en entrée

uint16_t GPIO_ReadInputData(port) ;

paramètre :port = GPIOA à. GPIOI

Lecture d'un port configuré en sortie

uint16_t GPIO_ReadOutputData(port) ;

paramètre : port = GPIOA à GPIOI

lecture d'une ligne configurée en entrée

uint16_t GPIO_ ReadInputDataBit (port, ligne) ;

paramètre : port = GPIOA à GPIOI
ligne = GPIO_Pin_0 à GPIO_Pin_15

Lecture d'une ligne configurée en sotie

uint16_t GPIO_ ReadOutputDataBit (port, ligne) ;

paramètre : port = GPIOA à GPIOI
ligne = GPIO_Pin_0 à GPIO_Pin_15

Ecriture dans un port

void GPIO_Write(port, valeur) ;

paramètre : port = GPIOA à GPIOI
valeur = 0 .. 65535

Ecriture dans une ligne de port

void GPIO_WriteBit(port, lignes, valeur) ;

paramètre : port = GPIOA à GPIOI
ligne = GPIO_Pin_0 à GPIO_Pin_15 ou GPIO_Pin_All
valeur : Bit_RESET ou Bit_SET

Mettre une ligne à 1 ou à 0

void GPIO_SetBits(port, lignes) ; void GPIO_ResetBits(port, lignes) ;

paramètre : port = GPIOA à GPIOI
ligne = GPIO_Pin_0 à GPIO_Pin_15 ou GPIO_Pin_All