Sujet de Travaux Pratiques
Tests des logiciel 2008

1 Introduction

Ce TP consiste à réaliser les tests de 3 fonctions à l'aide de l'outil Alcotest.

1. Tests Structurels : tester la fonction `Coef_calcul_puissance` avec une couverture des chemins, une couverture des conditions et une couverture des données aux limites.
2. Tests de Validation : tester le module `Voteur` avec une couverture fonctionnelle et une couverture des données aux limites.
3. Tests Structurels : tester la fonction `Calcul_prime` avec une couverture des branches, une couverture des conditions et une couverture des données aux limites.

Modalités et résultats attendus :
- un dossier présentant les éventuelles anomalies détectées durant les tests ;
- la trace d'exécution de l'outil Alcotest ;
- à rendre pour le jour de l'examen ;
- groupe de 2 personnes.

2 CALCUL_COEF_PUISSANCE

2.1 Rôle

La fonction `Calcul_Coef_Puissance` permet de calculer le coefficient de puissance d'un moteur et de retourner l'état de la tension courante. Ce coefficient de puissance dépend de la tension courante, des conditions opérationnelles du moteur `condition_1` et `condition_2` et du mode de fonctionnement `mode`. De plus, il doit être borné entre les valeurs `MIN_COEF` et `MAX_COEF`.

2.2 Traitement

```
#define VBAT_HIGH 24000     /* 240.00 Volts */
#define VBAT_LOW  20000     /* 200.00 Volts */
#define MAX_COEF  120
#define MIN_COEF  100

typedef enum {
  FALSE = 0,
  TRUE = 1
} Bool;

typedef enum {
  NOMINAL = 0,
  INITIALISATION = 1,
```

```
  DEFAUT = 2
} T_MODE;

void Calcul_Coef_Puissance(
   Bool condition_1,
   Bool condition_2,
   T_MODE mode,
   int vbat,
   Bool * vbat_ok,
   int * coef_op)
{
   *vbat_ok = FALSE;
   *coef_op = MIN_COEF;

   if ((vbat <= VBAT_LOW) || (vbat >= VBAT_HIGH)) {          /* C1 */
      *vbat_ok = FALSE;
   } else {
      *vbat_ok = TRUE;

      /* Calcul du coefficient en fonction de la tension,
         et des conditions opérationnelles */
      if (!condition_1                                       /* C2 */
          && (condition_2
              || (mode == NOMINAL)
              || (mode == INITIALISATION)))
         *coef_op = vbat / 180;
      else
         *coef_op = vbat / 220;

     /* Saturation des coefficients  */
     if (*coef_op > MAX_COEF)                                /* C3 */
        *coef_op = MAX_COEF;
     if (*coef_op < MIN_COEF)                                /* C4 */
        *coef_op = MIN_COEF;
   }

   return;
}
```

2.3 Prototype

```
extern void Calcul_Coef_Puissance(
  Bool condition_1,
  Bool condition_2,
  T_MODE mode, int vbat,
  int * vbat_ok,
  int * coef_op);
```

3 Module VOTEUR

3.1 Rôle

Le module `Voteur` permet de consolider les valeurs reçues de 3 capteurs redondants.

3.2 Traitements

Un capteur fournit deux informations au système de vote :
- la valeur physique lue sur 10 bits ([0 .. 1023]) ;
- une information de validité donnant l'état du capteur. Cette information indique si la valeur lue est valide ou non.

Le module capteur prend en entrée les 3 capteurs et fournit une valeur physique des capteurs consolidées ainsi que la validité de cette valeur. Les règles de détermination des sorties du voteur sont :
- lorsque les 3 capteurs sont valides, le voteur retourne la moyenne des 3 valeurs physiques avec une validité correcte ;
- lorsque 2 des 3 capteurs sont valides, le voteur vérifie la cohérence des 2 capteurs. Si les capteurs sont cohérents, il retourne la moyenne des 2 valeurs physiques cohérentes avec une validité correcte sinon il retourne une validité incorrecte. Deux capteurs sont cohérents si leurs valeurs sont identiques avec une tolérance de 5 ;
- afin d'optimiser la disponibilité du système, lorsqu'un seul capteur est valide, le voteur retourne la valeur de ce capteur avec une validité correcte si le capteur est dans la gamme [70..950] sinon il retourne une validité incorrecte ;
- lorsqu'aucun des capteurs n'est valide, le voteur retourne une validité incorrecte.

Par défaut, la valeur physique retournée est 0.

3.2.1 Prototypes

%beginverbatim

```
typedef struct {
  int first;  /* Pour la valeur du capteur */
  int second; /* Pour la validité du capteur */
} tuple;

typedef enum {
  FALSE = 0,
  TRUE = 1
} Bool;

void voteur(
  tuple c1,
  tuple c2,
  tuple c3,
  Bool *valid,
  int *value);
```

4 CALCUL_PRIME

4.1 Rôle La fonction

`calcul_prime` permet de calculer la prime d'un assuré suivant 3 critères : son âge, son sexe et sa situation familliale.

4.2 Traitement

La prime par défaut est de 500. Si l'assuré est un homme de moins de 25 ans qui n'est pas marié alors appliquer une sur-prime de 1500 sinon, si l'assuré est marié ou est une femme alors appliquer une décote de 200, enfin si l'assuré est agé de plus de 45 ans et de moins de 65 ans alors appliquer une décote de 100.

4.3 Prototype

```
typedef enum {
  FALSE = 0,
  TRUE = 1
} Bool;

typedef enum {
  MALE = 0,
  FEMALE = 1
} Gender;

extern void calcul_prime(
  int age,
  Gender gend,
  Bool married,
  int *prime);
```