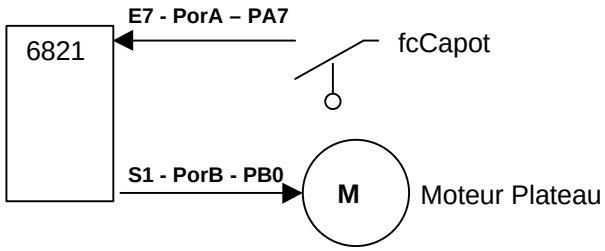
Classes Entrées/Sorties TOR

1 PRESENTATION

1.1 Architecture matérielle

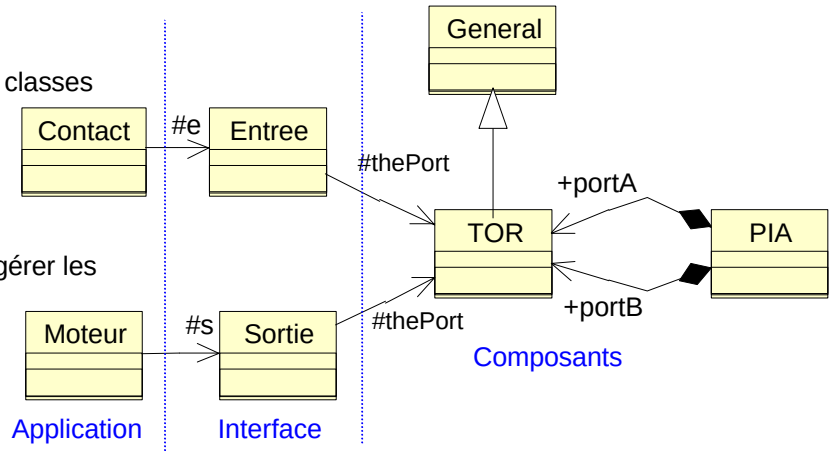
La partie opérative est couplée au système informatique par un composant 6821 (Pia) du type TOR. Le moteur est commandé par la borne 1 du port B du 6821. Le contact du fin de course connecté à la borne 7 du port A du 6821



1.2 Architecture du logicielle

Elle est construite autour de 3 couches de classes

- Les classes de l'Application : Contact et Moteur.
- Les classes d'Interface avec l'application manipulent des champs de bits : Sortie et Entree
- Les classes Composant chargées de gérer les composants périphériques : General, TOR et PIA



1.3 Eléments du programme principal

```
/* ---CONFIGURATION ------------------------ */
      PIA pia1(--);           // Initialise le PIA
      Sortie s1(&pia1.portB); // Associe la sortie 1 au Port B du PIA
      Entree e7(&pia1.portA); // Associe entrée 7 au Port A du PIA
      Moteur plateau(&s1);    // Associe le plateau à la sortie
      Contact fcCapot(&e7);   // Associe le contact fcCapot a l'entrée

/* ---Gérer le Moteur Plateau  ------------- */
                              //Si fcCapot=0
                              //Alors
                              //    Arrêter le Moteur plateau
                              //Sinon
                              //    Moteur plateau Marche
                              //Fin si
```

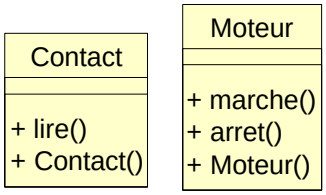
1.4 Les classes de l'Application

Corps des méthodes

```
bool Contact::lire(){
 return e->lireBit();
}
```

```
Moteur::marche() {
 s->ecrireBit(true);
}

Moteur::arret() {
 s->ecrireBit(false);
}
```



Modèle

Corps des constructeurs

```
Contact::Contact(Entree *ptrE)
                : e(ptrE){
}
```

```
Moteur::Moteur(Sortie *ptrS)
                : s(ptrS) {
}
```

2 TRAVAIL

2.1 La classe "General"

General
- etat : unsigned short - controle : unsigned short
+ lireEtat() : unsigned char + ecrireControle(val : unsigned char) + General(aEtat : unsigned short, aControle : unsigned short)

1. Quel est le rôle du constructeur ?
2. Qui (membres de la classe et instances) peut accéder aux membres « données » et « fonctions » de cette classe ?
3. Le constructeur est-il surchargé ?
4. Etablissez la déclaration de la classe `General` (General.h), en langage C++.
5. Etablissez le corps du constructeur, en langage C++.
6. Donnez le prototype du destructeur.
7. Quelle est la taille théorique d'une instance de la classe `General` ?

2.2 Classes de l'application

1. Proposez le code, en langage C++, correspond à la fonctionnalité «gérer le Plateau » du $ 1.2
2. Ces classes n'ont pas de destructeur dans leur fonction membre. Que se passe t-il lorsque l'on quitte le programme ?
3. Proposez une création dynamique d'une instance de la classe `Moteur`.
4. Quelles sont les propriétés de la relation entre la classe `Moteur` et `Sortie` ?
5. Etablissez la déclaration de la classe `Moteur` (Moteur.h), en langage C++..
6. Quelle est la taille théorique d'une instance de la classe `Moteur` ?

2.3 La hiérarchie de classe « General-TOR »

Les 2 classes sont en relation par un héritage public (Voir diagramme § 1.2)

Les données de 2 classes représentent des adresses de registres de composants périphériques.

- Le constructeur de la classe TOR réalise les opérations :

data←AdData ; etat←AdEtat ; direction←adDirection ; controle← adControle

- Les fonctions lire/ecrire opérent sur les registres dont les adresses sont données membres de ces 2 classes.

Exemple, la fonction *ecrireDirection()* réalise l'opération *[direction] ← valDirection*

Modèle de la classe TOR

TOR
- direction : unsigned short - data : unsigned short
+ lirePort() : unsigned char + ecrirePort(val : unsigned char) + TOR(adDirection : unsigned short, adData : unsigned short, adEtat : unsigned short, adControle : unsigned short) + ecrireDirection(valDirection : unsigned char)

1. A quels membres de la classe `General` peut accéder la classe `TOR` ?
2. A quels membres peut accéder une instance de la classe `TOR` ?
3. A quels membres de ces classes peuvent accéder la classe `PIA` et ces instances ?
4. A quels membres de ces classes peuvent accéder la classe `Sortie` et ces instances ?
5. A quels membres de ces classes peuvent accéder la classe `Moteur` et ces instances ?
6. Etablir un bilan sur les membres des classes qui peuvent modifier les données de ces classes.
7. Sachant que les données de la hiérarchie de classes représentent des adresses de registres, établir un bilan sur les membres des classes qui ont la possibilité modifier le contenu des registres.
8. Etablissez la déclaration de la classe `TOR` (TOR.h) , en langage C++.
9. Etablissez le corps du constructeur de la classe `TOR`, en langage C++.
10. Quelle est la taille théorique d'une instance de la classe `TOR` ?