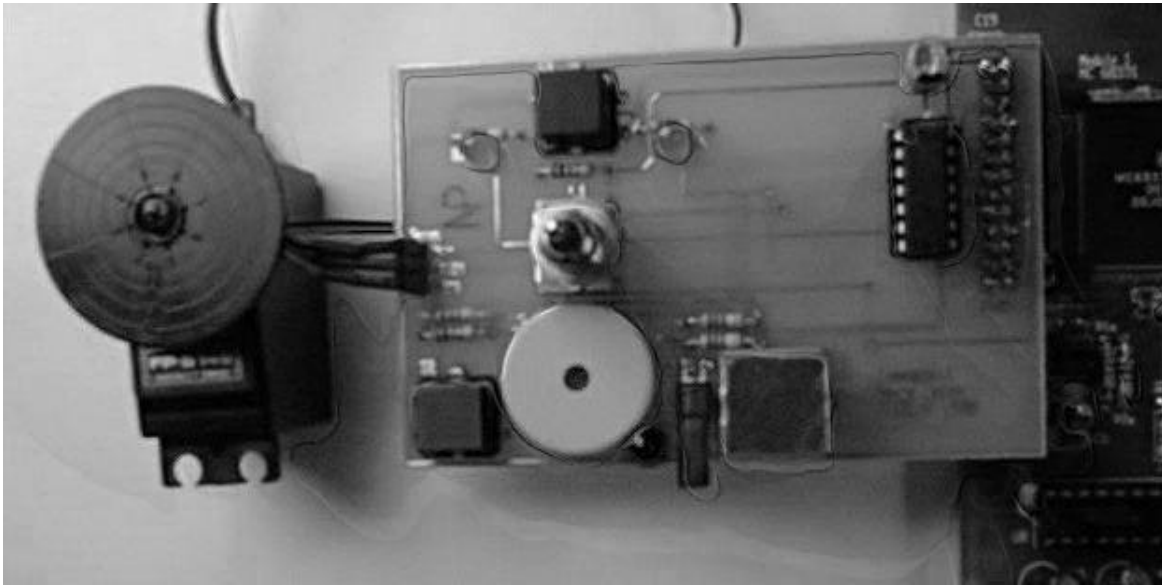


*Interface logicielle pour le TamagoKit*  
GEI-455 - Systèmes en temps réel



## Les fichiers

Afin d'interfacer le montage du TamagoKit à vos logiciels fonctionnant sur votre carte de développement 68331, vous avez besoin des deux fichiers suivants :

tamaki t. h	définitions des fonctions et constantes;
tamaki t. c	fonctions d'interface en C.

Assurez-vous qu'ils se trouvent dans le même répertoire que vos fichiers sources.

## Leur inclusion

Vous devez ajouter la directive

```
#include tamaki t. h
```

dans vos fichiers faisant appel aux routines d'interface et vous devez compiler tamaki t. c avec les fichiers constituant votre programme.

## Le fonctionnement interne

Le TamagoKit est relié, via le port P7, au General Purpose Timer (GPT) de votre carte de développement 68331. Le TamagoKit n'utilise que les lignes d'entrées/sorties et les horloges du GPT.

Les routines d'interface sont toutes en C et n'ont recours à aucune interruption. Il n'y a donc pas de danger qu'elles affectent le fonctionnement de votre noyau temps réel, à moins que vous n'utilisiez déjà le GPT. Toute la génération des signaux de contrôle de l'élément piézo-électrique et du détecteur de présence se font par scrutation. Le servomoteur est quant à lui branché sur une ligne pouvant générer sans l'aide du CPU les impulsions nécessaires à son contrôle. Le reste des lignes utilisées ne sont que de simples entrées/sorties digitales.

# Les fonctions

Les routines d'interface pour le TamagoKit sont les suivantes :

## **void init\_tamagokit(void);**

**Init\_tamagokit()** est la première fonction que vous devez appeler avant d'utiliser le TamagoKit. Cette routine initialise le GPT afin de permettre l'interface entre la carte 68331 et le TamagoKit.

## **void test\_tamagokit(void);**

Cette routine ne sert qu'à tester le bon fonctionnement des éléments composant le TamagoKit. Au moment de l'appel, **test\_tamagokit()** initialise la carte, émet trois sons de 1 seconde chacun, puis tombe dans une boucle sans fin scrutant les entrées de la carte et retournant leur état sur les sorties :

- le détecteur de présence active l'oeil droit;
- le bouton pour caresser active l'oeil gauche;
- le nez active le servomoteur;
- le bouton pour nourrir active la langue.

## **short get\_pet\_button(void);**

Retourne l'état du bouton S3 pour caresser. **get\_pet\_button()** retournera une valeur non-nulle si le bouton est enfoncé et zéro s'il n'est pas enfoncé.

## **short get\_feed\_button(void);**

Retourne l'état du bouton S2 pour nourrir. **get\_feed\_button()** retournera une valeur non-nulle si le bouton est enfoncé et zéro s'il n'est pas enfoncé.

## **short get\_nose\_button(void);**

Retourne la position du nez (interrupteur S1). **get\_nose\_button()** retournera trois valeurs distinctes, toutes définies dans tamaki t. h :

NOSEUP	lorsque le bouton est poussé vers le haut;
NOSECENTER	lorsque le bouton est au repos au centre;
NOSEDOWN	lorsque le bouton est poussé vers le bas.

**short get\_presence(void);**

Active la diode infrarouge D1 et vérifie si le détecteur U1 perçoit ou non un objet devant lui. `get_presence()` retournera une valeur non-nulle s'il y a quelqu'un ou un objet devant le détecteur U1 ou zéro s'il n'y a rien. NOTE : Lorsque le servomoteur est en cours de positionnement, `get_presence()` peut se mettre à détecter des objets alors qu'il n'y en a pas. Cet effet est provoqué par les fluctuations sur les alimentations, fluctuations auxquelles le détecteur U1 est très sensible. `get_presence()` ne devrait donc pas être utilisé tant que le servomoteur n'est pas immobile. De plus, si U1 persiste à percevoir des objets inexistant, il se peut qu'en fait il détecte la table sur laquelle repose votre montage. Soulevez D1 (la diode avec la gaine) de façon à ce qu'elle pointe plus vers le haut.

**void left\_eye(short status);**

Modifie l'état de l'oeil gauche, la LED D2. `left_eye()` allumera l'oeil si *status* est non-nul et l'éteindra si *status* est égal à zéro.

**void right\_eye(short status);**

Modifie l'état de l'oeil droit, la LED D3. `right_eye()` allumera l'oeil si *status* est non-nul et l'éteindra si *status* est égal à zéro.

**void tongue(short status);**

Modifie l'état de la langue, la LED D4. `tongue()` allumera la langue si *status* est non-nul et l'éteindra si *status* est égal à zéro.

**void servomotor(unsigned short position);**

Positionne le servomoteur selon la valeur spécifiée. `servomotor()` nécessite comme argument une valeur entre 0 et 100. NOTE : Lorsque le servomoteur est en cours de positionnement, `get_presence()` peut se mettre à détecter des objets alors qu'il n'y en a pas. Cet effet est provoqué par les fluctuations sur les alimentations, fluctuations auxquelles le détecteur U1 est très sensible. `get_presence()` ne devrait donc pas être utilisé tant que le servomoteur n'est pas immobile.

**void buzzer(unsigned long freq, unsigned short millisec);**

Cette fonction permet de générer des sons à partir de l'élément piézo-électrique U3. La fréquence, en Hertz, est spécifiée avec l'argument *freq* et la durée du son, en millisecondes, est spécifiée avec l'argument *millisec*. La plage de fréquence que `buzzer()` accepte va de 16 Hertz à 50 000 Hertz, bien que votre oreille cessera de percevoir les sons à partir de 16-17kHz...