# Manuel de maintenance

Ce guide présente les diverses modifications à faire si l’on veut modifier le comportement du banc de test.

Actuellement le programme de carte intermédiaire est paramétré pour fonctionner avec 2 ports UART. Si vous voulez modifier ce comportement ou si vous utilisez une carte intermédiaire différente, il faudra modifier les portions de code suivantes :

**Ajout de buffers circulaires**

Il est conseiller d’ajouter premièrement les nouveaux ports UART via l’outil de configuration CubeMX afin de ne pas avoir à créer les fonctions d’initialisations à la main.

*l. 84*

#define MAX\_UART\_PORT 2

Modifier le nombre de port ici

*l. 90*

#define CIRC\_BUFFER\_MAX\_SIZE 550

En fonction des besoins, agrandir ou réduire la taille des buffers en modifiant la valeur précédente

*L. 140*

CIRC\_BBUF\_DEF(cbuf1, CIRC\_BUFFER\_MAX\_SIZE);

CIRC\_BBUF\_DEF(cbuf2, CIRC\_BUFFER\_MAX\_SIZE);

Rajouter ici les définitions des nouveaux buffers.

*l. 161*

circ\_bbuf\_t \*bbuf\_table[MAX\_UART\_PORT] = {

&cbuf1,

&cbuf2

};

Rajouter les nouveaux buffers définit dans le tableau de pointeurs vers les buffers.

*l. 571*

if(huart == &huart2)

{

circ\_bbuf\_push(&cbuf2, rxBuff2);

HAL\_UART\_Receive\_IT(huart, &rxBuff2, 1);

}

else if(huart == &hlpuart1)

{

circ\_bbuf\_push(&cbuf1, rxBuff1);

HAL\_UART\_Receive\_IT(huart, &rxBuff1, 1);

}

Modifier la fonction de callback de l’UART pour ajouter les fonctions de « push » dans les nouveaux buffers circulaires.

**Améliorations envisageables**

Même si nous sommes arrivés à un produit fonctionnel, certaines portions de code auraient, éventuellement, pu être différentes au niveau de la réalisation. Les lignes qui suivent proposent des voies qui n’ont pas été suivi lors de ce stage et qui pourraient, peut-être, mieux convenir au fonctionnement voulu par l’entreprise.

*Programme carte intermédiaire*

* Utiliser un seul buffer circulaire
  + Recevoir d’un UART jusqu’à la fin de ligne, puis changer de port UART.
* Modifier le fonctionnement du programme pour le changement du port UART
  + Actuellement le programme change de port UART lorsque les 2 ports UART ont commencé à recevoir des données et lorsque que le caractère transmis par I²C correspond à un \r ou \n. Peut être que le changement de port UART pourrait s’effectuer d’une autre façon plus appropriée (par exemple supprimer l’attente d’au moins un caractère sur chaque port à la ligne 278 du programme)

*Programme Raspberry*

* Supprimer la structure « deviceInfo\_t »
  + Cette structure n’est peut être pas indispensable au fonctionnement du programme.
* Supprimer la fonction i2cParse
  + A la réception de la trame I²C, nous utilisons actuellement une fonction permettant d’assigner les 2 caractères reçues aux variables correspondantes. La fonction faisant uniquement 2 lignes, il n’est peut-être pas indispensable et le code aurait pu éventuellement être directement mis dans le « main ».
* Exploiter ou Supprimer le nombre de ports UART
  + Lors de la configuration, le nombre de port UART par cartes intermédiaires et demandé à l’utilisateur afin d’afficher, via un calcul, le nombre total de capteur testés. Peut-être que l’on pourrait exploiter de manière plus utiles ce paramètre de configuration.

**Bugs éventuels**

Il est possible que certains bugs surviennent lors de l’utilisation du programme. Il est expliqué ici comment les résoudre.

* Non réception des logs
  + Lors du lancement du banc de test, il est possible que les logs ne soient pas reçus dans les cas suivants :
    - Le capteur n’est pas en mode DEBUG.
    - Il n’y a pas de capteur branché sur le 1er port UART
  + Solutions
    - Passer le capteur en mode debug
    - Toujours mettre le nombre maximum de capteurs sur une carte intermédiaire (ou modifier le système de changement de port comme suggérer précédemment)