User Guide
UARTStack

Table des matières

1. Présentation de la pile de communication UART Stack

1.1. Généralités

Le notion de pile ici est un peu galvaudée dans le sens où il n'y a pas de notion d'adresse origine, destination. Il s'agit juste d'une communication UART entre deux entités en full duplex (STM32F103 et Raspberry pi).

Toutefois, sont implémentés :

- un timeout,
- un checksum basique

Le principe repose sur la réception d'une trame prédéfinies fondées sur un nombre d'octets à lire :

Format de trame :

|Nb Octets| Bytes de données | CheckSum|

Nb Octets : C'est le nombre d'octets à venir que l'UART doit lire. Ce nombre correspond donc au nombre de bytes de la payload + 1 byte de checksum.

Bytes de données : Les données effectives en bytes. C'est la payload.

CheckSum : C'est la somme de tous les bytes de la payload, y compris le tout premier octet correspondant au nombre d'octets de la trame. Il s'agit de la troncature de la somme pour ne conserver que l'octet de poids faible.

La gestion du *timeout* nécessite un timer dédié.

Au final, la chaîne remontée à l'utilisateur n'est que la *Payload.* La longueur est à récupérer avec une fonction dédiée.

Exemple :

Chaîne transmise : |5|'T'|'O'|'T'|'O'| checkSum|

0 1 2 3 4 5

Longueur remontée : 4

Chaîne remontée : |'T'|'O'|'T'|'O'|

1.2. Réception de trame

Le module contient un tableau de char de longueur 256. En effet, la longueur de trame peut valoir 255 au maximum, ce qui donne une longueur de Payload de 254 maximum. On marge de deux octets. Cette chaîne se nomme *char HMISting[256]*.

1.2.1. Principe

L'UART choisie fonctionne en interruption. Lors de la première interruption, la longueur est enregistrée dans le premier caractère de *HMISting*. Les interruptions suivantes complètent la construction de *HMISting.*

Dès la réception du premier octet (le nombre d'octets à recevoir), le timer dédié au timeout est lancé. Quelque soit l'issue de la réception, l'interruption timeout a lieu. C'est à ce moment que le code détermine si la réception s'est bien passée, ou bien si elle est incomplète.

Il n'y a pas de machine à états pour cette « pile » étant donné sa simplicité.

Pour que l'application puisse exploiter les données de réception, 3 fonctions sont utilisées, l'une indiquant qu'une trame valide est à lire, l'une précisant la longueur de la payload, l'autre enfin donnant l'adresse du premier octet de la payload.

1.2.2. Gestion des erreurs en réception

Un type erreur est fourni dans l'API : *UARTStack_ErrorType*.

Trois statuts erreurs sont identifiables :

- *NoError* : vaut '1' a l'initialisation et uniquement à réception de la chaîne, si pas d'erreur *Checksum,*
- *CheckSumError* : vaut '1' à la réception complète de la chaîne, si erreur *Checksum,*
- TimeOut : vaut '1' si le timer est arrivé à échéance et si la chaîne est incomplète

1.2.3. Gestion du flag de réception

Le flag de réception est *HMIStringComplete*. Il est accessible via la fonction *UARTStack_IsHMIMssg(void)*.

HMIStringComplete est mis à 1 :

- lorsque le nombre de caractères est atteint.

NB : l'UART est dès lors bloquée.

HMIStringComplete est mis à 0 :

- Lors d'nue lecture du string de réception,
- au début du processus de réception (lecture du nombre d'octets à saisir)

1.2.4. Gestion de l'activité de l'UART dédiée

L'UART est validée :

- à l'initialisation,
- après un timeout (défini en multiple de 100ms),

L'UART est invalidée :

- Dès de la réception de n octets.

1.2.5. Gestion du timeout

Le timer dédié est réglé pour un cycle de 100ms. Le paramètre *TimeOut_x100ms* permet de spécifier le temps d'attente en multiple de 100ms. Par défaut 5, soit 500ms.

1.2.6. Aspect Temps réel

Deux périphériques fonctionnent en interruption. Les priorités sont les suivantes :

- UART : priorité 0
- Timer dédié au Timoute : priorité 1

1.3. Emission de trame

Se fait en mode scrutatif.

2. Les fonctions API du module

2.1. Fonction d'initialisation

void UARTStack_Init (void) :

- lance la configuration de l'UART retenue. Le BaudRate est par défaut à 9600Bds, il peut être changé dans le fichier *RessourcesPeriph.h*.
- Initialise le Timer dédié (choix fait dans *RessourcesPeriph.h*) pour le TimeOut. Le Timer n'est pas lancé.

2.2. Fonctions de réception

char UARTStack_IsHMIMssg(void);

- Renvoie '1' si une chaîne est à lire

char UARTStack_GetLen(void);

- Renvoie la longueur de la chaîne brute (la payload)

*char * UARTStack_GetHMIMssg(void);*

- Renvoie l'adresse du premier octet de data à lire (payload)

UARTStack_ErrorType UARTStack_GetErrorStatus(void);

2.3. Fonction d'émission

void UARTStack_SendNewMssg (char * AdrString, int Len);

- Envoie la chaîne de caractères spécifiée (avec sa longueur) en respectant la trame (donc avec premier octet de longueur, puis à la fin un checksum
- travaille en scrutation donc bloquant (pas d'IT)

3. Dépendances