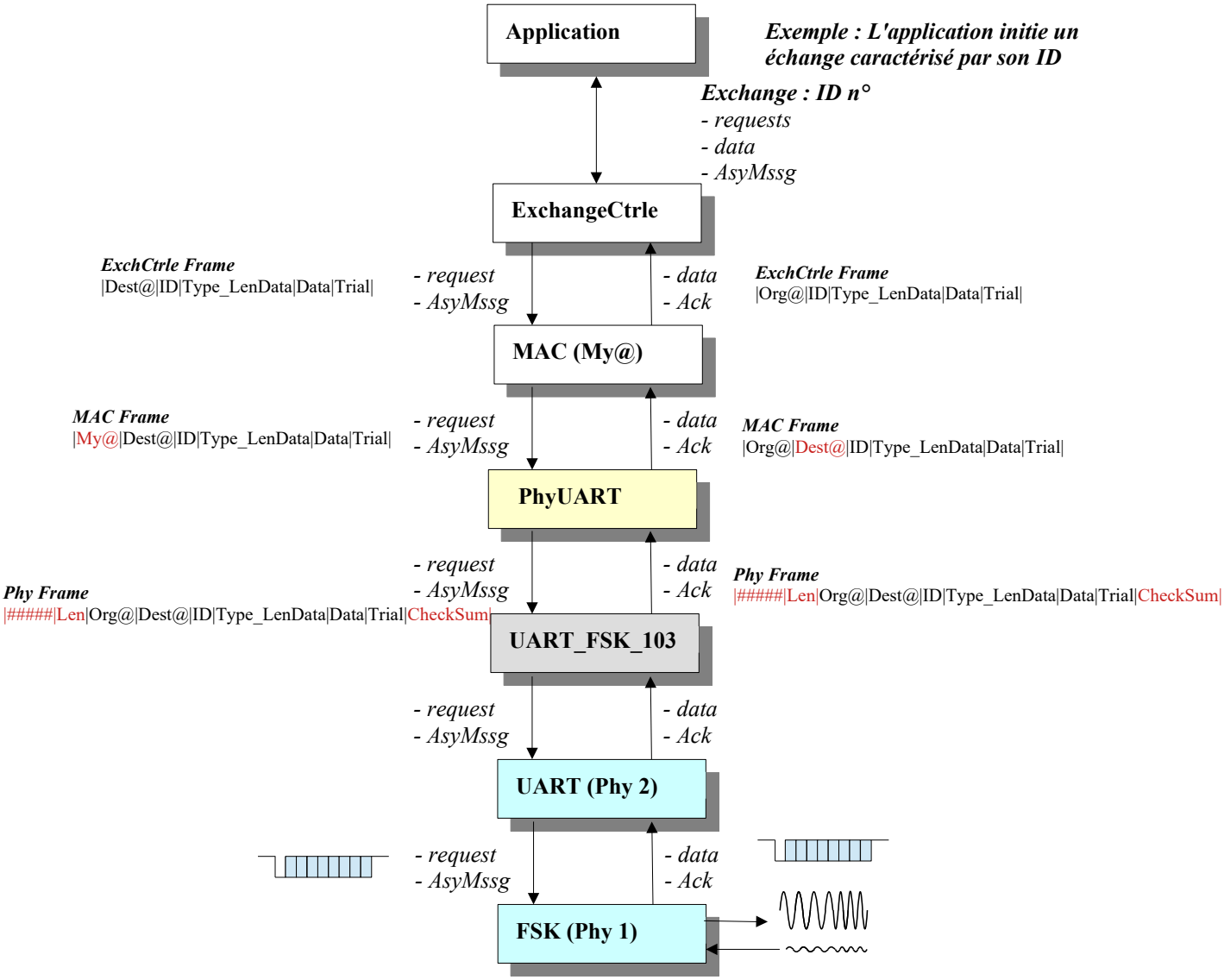
User Guide
FSK_COM_STACK

Table des matières

1. Architecture de la pile de communication

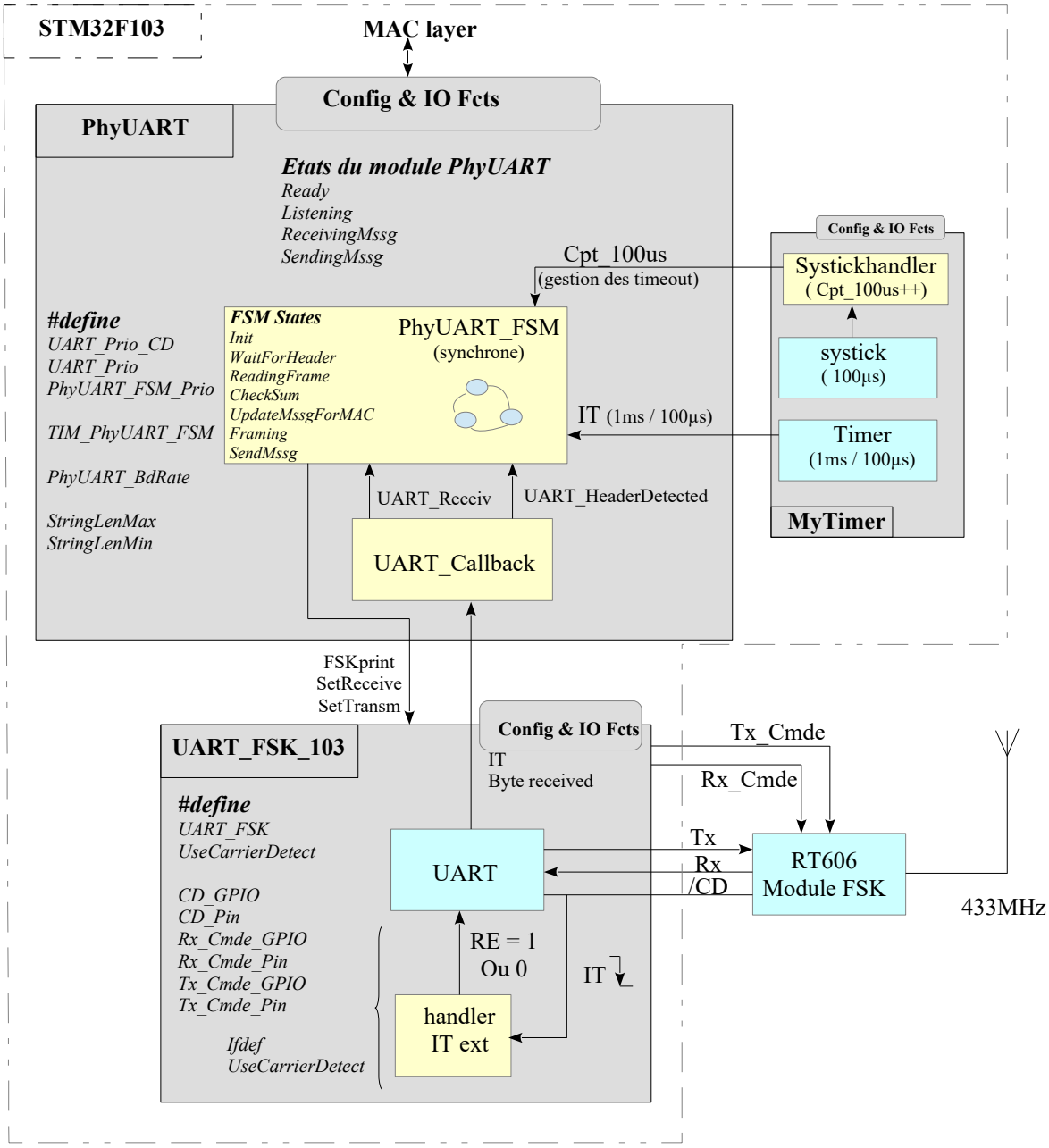


2. Couches physique

2.1. Vue d'ensemble

La couche physique regroupe :

- FSK (Phy 1) , le module FSK RT606 = matériel hertzien
- UART (Phy 2), l'UART du µC = matériel
- UART_FSK_103 , module driver UART hardware dépendant = logiciel bas niveau
- Phy_UART, module de plus haut niveau de la couche physique = logiciel



2.2. Description des fonctions

2.2.1. UART_FSK_103.c/.h

Le module dépend du hardware.

Deux interruptions sont utilisées :

- réception octet,
- détection de porteuse (optionnelle). Si le *UseCarrierDetect* est défini, l'UART est bloqué en réception en absence de porteuse. Cela permet d'éviter d'avoir des interruptions UART qui arrivent constamment en présence de bruit.

Les broches :

CD, RxCmde et Tx Cmde sont définies dans le .h.

Tx et Rx sont définies automatiquement, et configurées dans le .c en fonction de l'USART choisie (USART1 , 2 ou 3).

Les fonctions de configurations

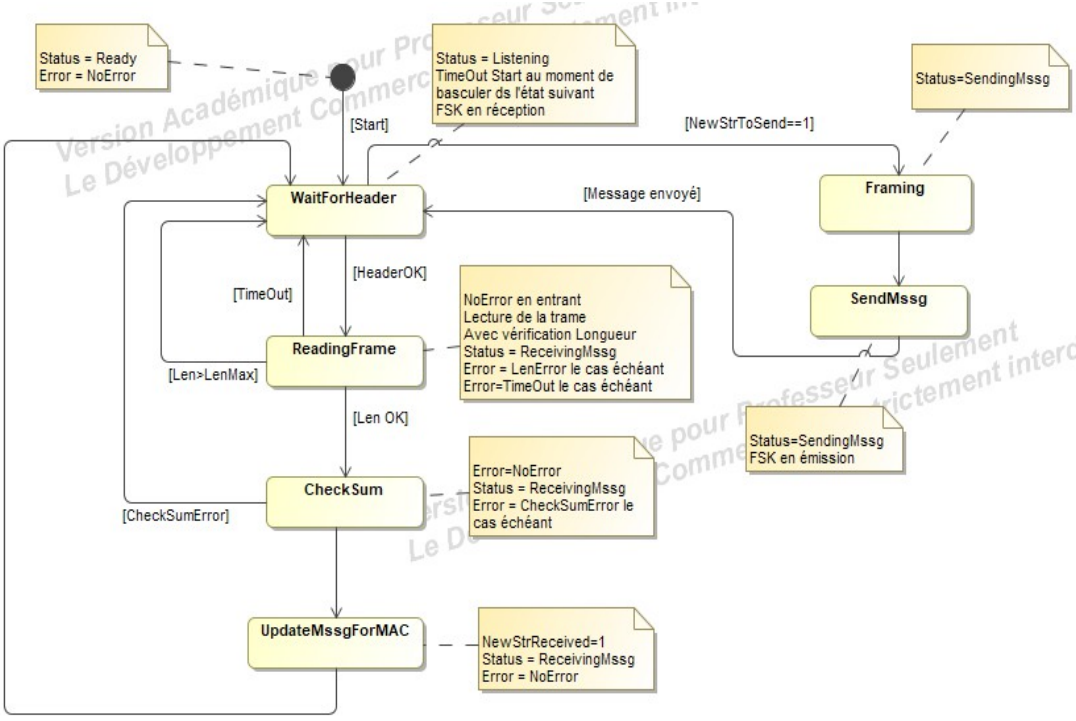
```
/*****************************************************************************************************
 Configure l'UART spécifiée en #define, avec le débit indiqué (en Baud), la priorité utilsée pour
 la détection de porteuse, la priorité d'interrution de réception, et enfin, on précise la fonction
 callback associée à l'interruption.
 Exemple :
 USART_FSK_Init(PhyUART_BdRate,UART_Prio_CD,UART_Prio,UART_Callback);
*****************************************************************************************************/
void USART_FSK_Init(int Baud_Rate_bits_par_Sec, char Prio_USART_CD, char Prio_USART, void (*IT_function) (void));
```

Les fonctions IO

```
/*****************************************************************************************************
 Renvoie le dernier caractère lu par l'UART
 Exemple :
 if (USART_FSK_GetByte()==HeaderCar)
*****************************************************************************************************/
char USART_FSK_GetByte(void);

/*****************************************************************************************************
 Envoie caractère par caractère la chaîne dont l'adresse est spécifiée. Le nombre d'octet à émettre
 est spécifiée par le paramètre Len.
 Exemple :
 USART_FSK_Print("1234",4);
*****************************************************************************************************/
void USART_FSK_Print(char* Mssg, int Len);

/*****************************************************************************************************
 Fixe le module FSK en position de réception
*****************************************************************************************************/
void USART_FSK_SetReceiveAntenna(void);

/*****************************************************************************************************
 Fixe le module FSK en position d'émission
*****************************************************************************************************/
void USART_FSK_SetTransmAntenna(void);
```

2.2.2. PhyUART.c/.h

La FSM associée est synchrone. Elle est basée sur timer dont la périodicité des interruptions peut prendre deux valeurs, selon le mode dans lequel se trouve la FSM (cf ci-dessous) :



Dans l'état *WaitForHeader* (majorité du temps), l'échantillonnage de la FSM est de 1ms tant que le *Header* n'est pas détecté. Dès que celui-ci est détecté (dans le callback UART), le timing descend à 100µs ainsi que pour tous les autres états. Le retour à l'état *WaitForHeader* se fait toujours après avoir modifié le timing à 1ms.

Les #define

#define StringLenMax 30

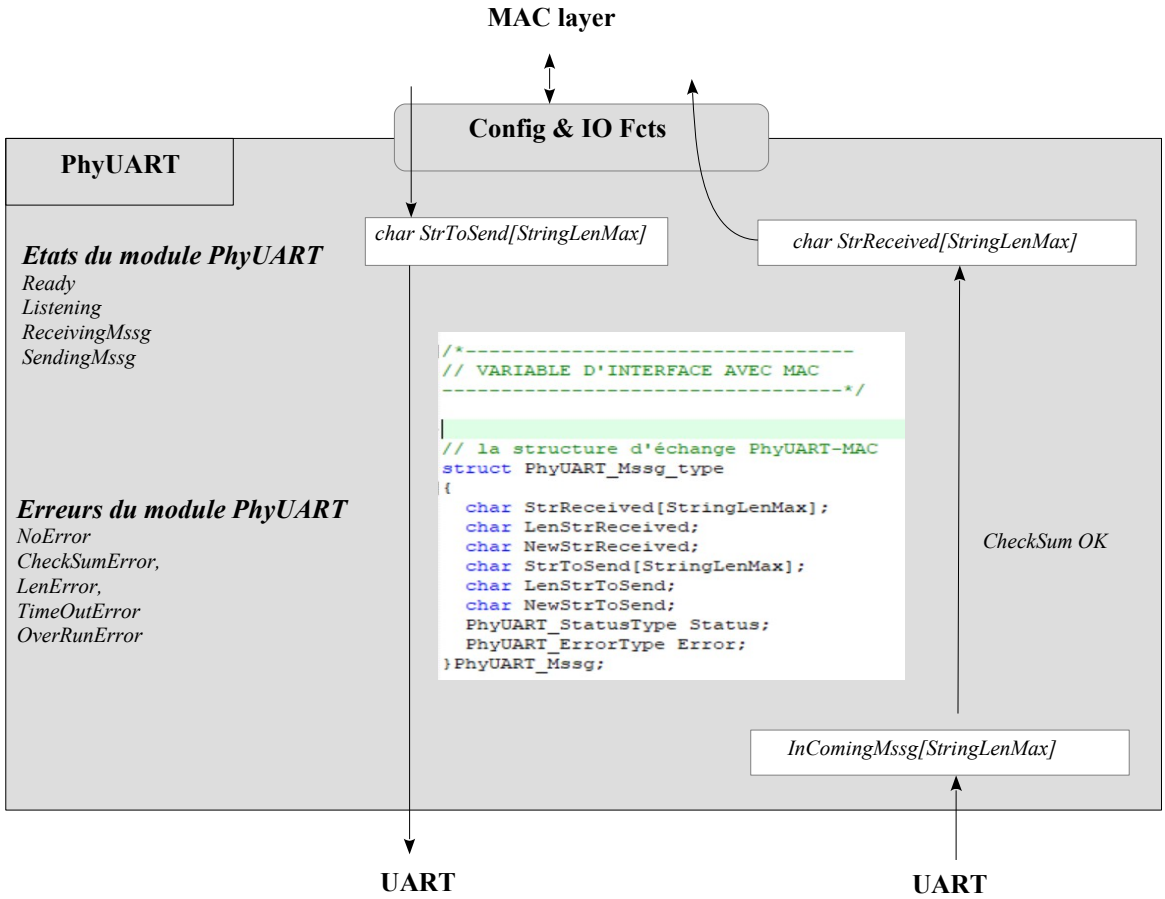
#define StringLenMin 7

Ce sont les longueurs qui définissent la chaîne de réception et la celle d'émission.

Les fonctions

```
/*****************************************************************************************************
 Configure le module PhyUART (Timer, UART, état initial de la FSM...
*****************************************************************************************************/
void PhyUART_Init(void);

/*****************************************************************************************************
 Lance la FSM en validant la transition vers l'état WaitForHeader
*****************************************************************************************************/
void PhyUART_StartFSM(void);


char PhyUART_IsNewMssg(void);
PhyUART_StatusType PhyUART_Get_Status(void);
PhyUART_ErrorType PhyUART_Get_Error(void);

/*****************************************************************************************************
 Permet de recopier la donnée validée par la couche PhyUART.
 Len est la longueur Maximale pour cette recopie.
 ATTENTION ! la chaîne samplée par l'UART est entièrement recopiée à partir de l'adresse pointée.
 Un test est donc fait pour savoir si la longueur du tableau recevant est suffisant.
 La fonction répond -1 en cas d'erreur sur ce test ou 0 si tout est OK
*****************************************************************************************************/
int PhyUART_GetNewMssg (char * AdrString, int Len);
int PhyUART_SendNewMssg (char * AdrString, int Len);
```

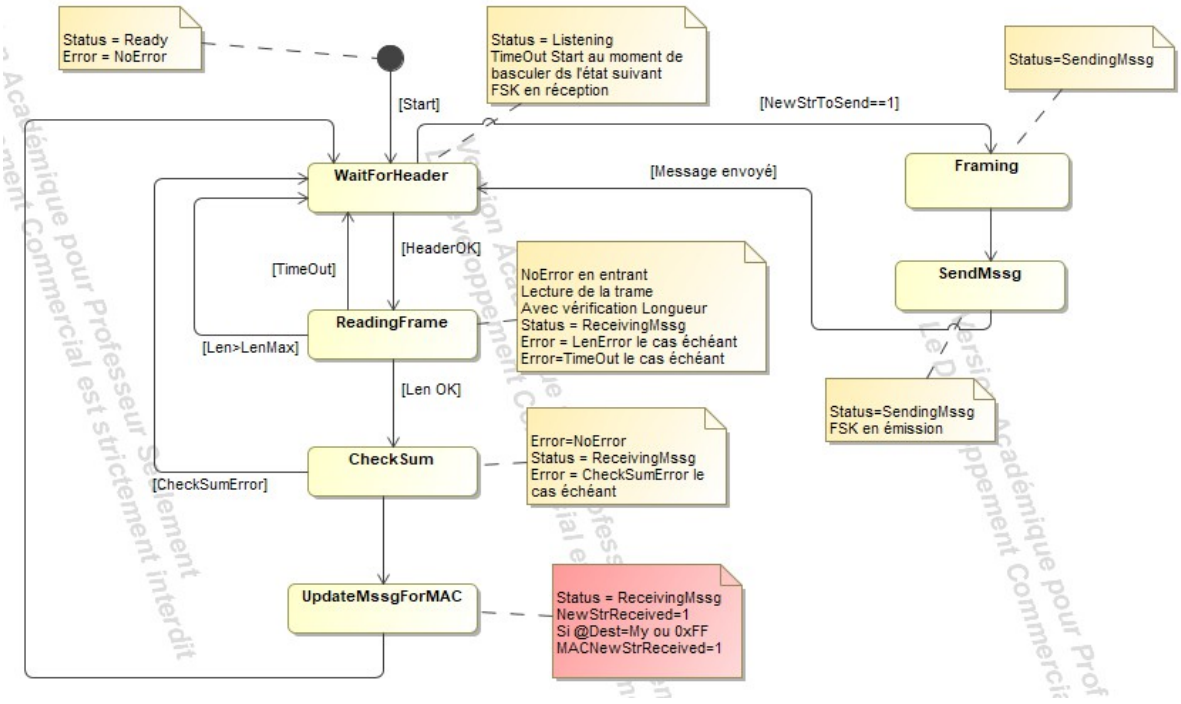
2.3. Le flux de données



3. La couche MAC

3.1. Vue d'ensemble

Elle est greffée au module PhyUART, le module est renommé : MACPhyUART.
La machine à état est très légèrement modifiée (partie UpdateMssgForMAC), voir ci-dessous.



3.2. Les fonctions de la couche MAC

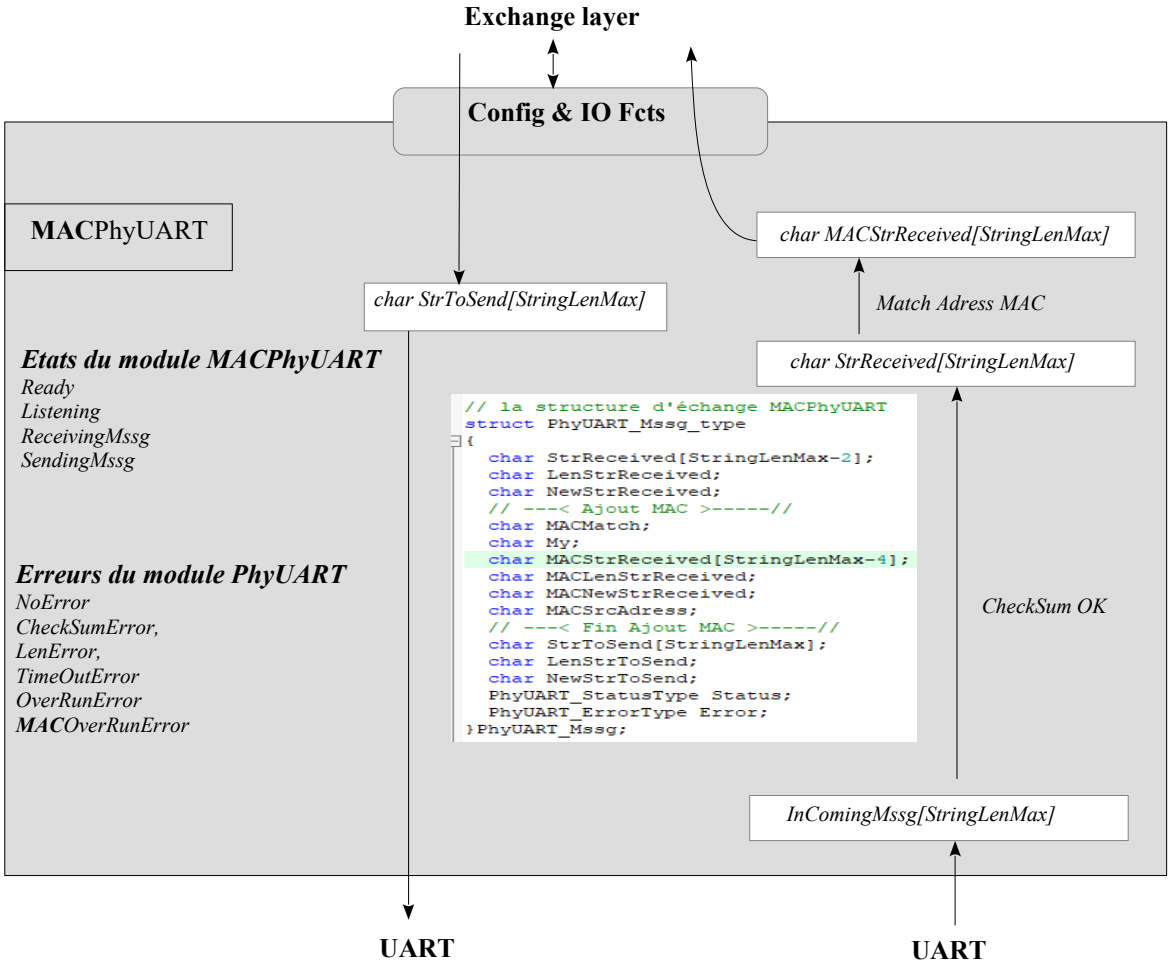
```
//*****************************************************************************
//*****************************************************************************
//                  COUCHE MAC
//*****************************************************************************
//*****************************************************************************

void MACPhyUART_Init(char My);
#define MACPhyUART_StartFSM PhyUART_StartFSM
char MACPhyUART_IsNewMssg(void);
int MACPhyUART_GetNewMssg (char * AdrString, int Len);
char MACPhyUART_GetSrcAdress(void);
char MACPhyUART_GetLen(void);
int MACPhyUART_SendNewMssg (char DestAdr, char * AdrString, int Len);
```

NB : Si l'utilisateur décide d'exploiter la couche MAC, il utilisera uniquement, exclusivement les fonctions démarrant par MAC. C'est la raison d'être du #define de démarrage de la FSM. Il s'agit effectivement de la même fonction qu'on soit en MAC ou en Phy.

NB : la taille maximale des datas fournies par la couche MAC et livrées à la couche MAC ont une taille maximale qui dérive de la taille maximale fixée au niveau du message UART (plus basse couche physique). Cette taille est définie par *StringLenMax* dans le .h, initialisée par défaut à 30.

Dans ces conditions, le maximum des datas MAC est de **26**.

3.3. Les flux de donnée



3.4. Les erreurs

Niveau Phy

- *NoError :* tout va bien,
- *LenError* : Test de Len (premier octet reçu) , la longueur de la chaîne reçue au niveau UART est soit :
  - Trop grande, précisément, supérieure strictement à *StringLenMax* (define .h),
  - Trop petite, précisément, inférieure strictement à *StringLenMin* (define .h). En effet, dans notre cas, la chaîne doit contenir au moins *Len, @src, @dest,ID, Type_LEN,Trial, CheckSum*, soit 7 otctets.

  Aucun échantillonnage ne se fait dans *IncomingMssg*.
- *CheckSumError :* la détection du *header* est passée, l'échantillonnage dans *IncomingMssg* a eu lieu, mais la vérification du checksum a échoué,
- *TimeOutError :* la trame a commencée à être reconstituée dans *IncomingMssg* mais elle n'a pas été remplie dans le temps imparti. Le *timeout* est automatiquement ajusté en fonction du débit de la liaison série. Il est calculer à +10% de la longueur maximale pouvant être reçue.
- *OverrunError :* cette erreur survient au moment de la mise à jour du flag de réception : si ce dernier était déjà à 1, ce code erreur est activé. Cela signifie que le dernier message Phy n'a pas été lu.

Niveau MAC

- *MACOverrunError :* cette erreur survient si le message MAC précédent n'a pas été lu.