

TP : INFORMATIQUE INDUSTRIELLE

Réseau 10TP

Emulation d'une communication entre le coffret répétiteur et le MEA3C sur le bus vent interne

1) PRESENTATION GENERALE DU RESEAU NAVIGATION

Le système de navigation du Système se décompose en sept sous-systèmes :

- Système de navigation basique,
- Système de données NAV,
- Système de radionavigation,
- Système de visualisation des données de navigation,
- Système environnement air et eau,
- Radars de navigation,

Un schéma synoptique de système de navigation est représenté ci-dessous

2) Le système environnement air et eau ()

Le système environnement air et eau se compose :

- D'un réseau (Environmental Air & Water System),
- D'un ensemble de bathymétrie.

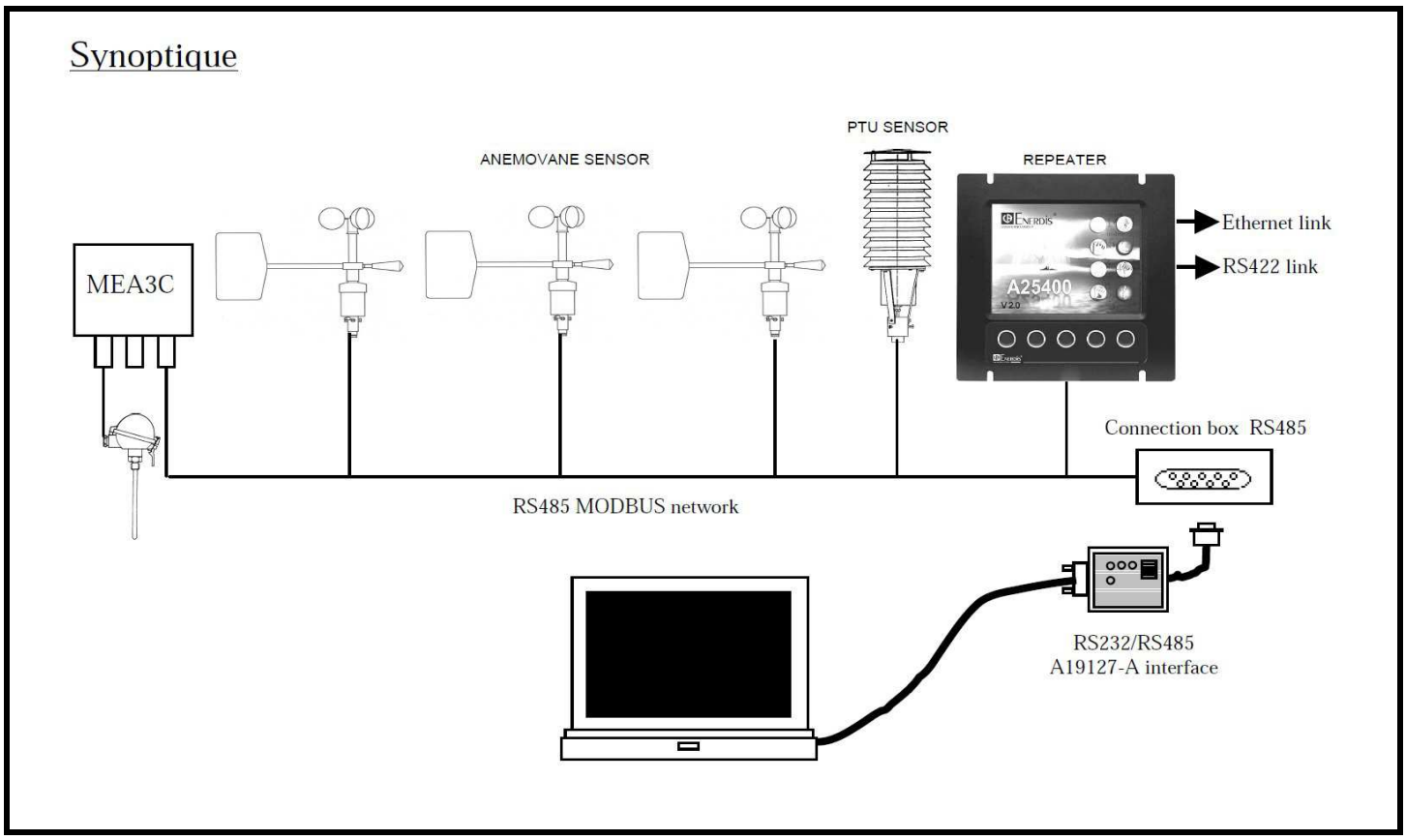
Le réseau est un système électronique permettant la mesure et l'élaboration de la situation météo. Ses fonctions essentielles sont :

- L'aide à la conduite du bâtiment,
- L'aide à la mise en œuvre
- L'aide à la mise en œuvre
- La mise à disposition de la situation météorologique (température air et mer, pression, taux d'humidité).

3) ARCHITECTURE DU RESEAU EAWS

Le réseau d'acquisition des paramètres météo et de traitement du vent se compose de 6 sous-ensembles qui se répartissent en 4 catégories (Voir planche 1 dans la schémathèque).

- Le coffret Répétiteur (séquencement des capteurs et visualisation des mesures)
- Les Capteurs Anémomètre Girouette (mesure de la vitesse et de la direction du vent)
- Le Capteur PTU (mesure de la pression, de la température, et de l'humidité de l'air)
- Le convertisseur Analogique/Digital MEA3C (mesure la température de l'eau) équipé de sa sonde PT100.



3.1) LE COFFRET REPETITEUR

Le répétiteur assure l'acquisition des données des capteurs, leur traitement, et l'élaboration d'informations nécessaires à la distribution des mesures à travers le bâtiment.

Ce sont les seuls équipements qui assurent la gestion du bus RS485.

Le répétiteur réalise 3 traitements distincts :

- La détermination du vent « relatif » manuel
- La détermination du vent « relatif » automatique
- La détermination du vent « vrai » (dit « absolu »)

En fonctionnement normal, le répétiteur interroge l'ensemble des capteurs MODBUS.
Il récupère en Multicast, via sa liaison Ethernet, les informations CAP et SOG du bâtiment venant du bus navigation et permettant de calculer le vent vrai.
A la mise sous tension du réseau, le répétiteur « Maître » séquence le réseau et le mode de fonctionnement est celui mémorisé en EEPROM.

Fonctions réalisées par l'équipement

- Gestion du bus RS485 du réseau de vent numérique
- Acquisition des mesures des capteurs
- Visualisation des mesures en face avant sous forme numérique et pseudo analogique
- Sélection automatique du réseau optimal
- Elaboration des réseaux de vent Pont / Hélicoptère
- Elaboration des réseaux de vent absolu
- Gestion de la liaison Ethernet en UDP
- Gestion de la seconde liaison série RS 422
- Réglage de luminosité

Ce coffret comporte une unité centrale équipé d'un microcontrôleur 32 bits assurant la gestion des trois liaisons de communication, le traitement de l'information, et le réglage des paramètres spécifiques de fonctionnement de l'appareil.
L'accès aux modifications du paramétrage est réalisé à partir d'un micro-ordinateur de type PC à travers une interface RS232/RS485, et du logiciel RESEAU_FREMM \ REPETV2.

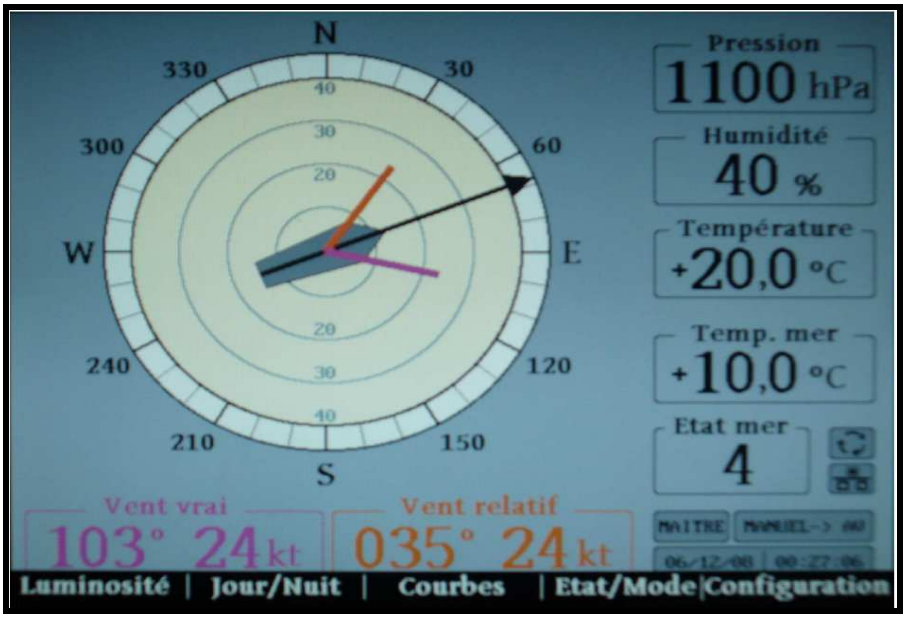
Caractéristiques d'affichage

- Affichage numérique de la vitesse
- Affichage numérique de la direction
- Affichage pseudo analogique de la vitesse et de la direction du vent
- Affichage numérique CAP / SOG
- Affichage numérique Pression / Température air
- Affichage numérique Hygrométrie
- Affichage numérique Température mer
- Affichage numérique de l'Etat de la mer

Liaisons numériques

- 1 liaison RS485 protocole MODBUS RTU maître (bus interne capteurs)
- 1 liaison RS422 protocole NMEA183
- 1 liaison Ethernet protocole UDP

L'interface homme machine



3.2) LE COFFRET CONVERTISSEUR MEA3C

Le coffret MEA3C est constitué d'un module MEA3 incorporé dans un boîtier étanche.
Il permet la mesure de 3 voies analogiques dont une seule servira à notre application.
Cette unique voie assure la mesure de la température de l'eau de mer.
Ce capteur comporte une unité centrale à microcontrôleur assurant l'acquisition des trois mesures et leur traitement, ainsi que le réglage et la modification des paramètres spécifiques de fonctionnement de l'appareil.
L'accès aux modifications du paramétrage est réalisé à partir d'un micro-ordinateur de type PC à travers une interface RS232/RS485, et du logiciel RESEAU \ MEA3. Le format des trames est le suivant :

Interrogation du boîtier MEA3

Message d'interrogation du Répétiteur

07	04	00	00	00	03	CRC	CRC

Réponse du boîtier MEA3

			Voie 1		Voie 2		Voie 3			
07	04	06	XX	XX	XX	XX	XX	XX	CRC	CRC

Avec

Information	Nb octets	Codage	Bits utilisés	Fin d'échelle	Résolution
Voie 1 (Température eau mer)	2	binaire signé Cplt 2	16 bits	59,99 °C	0,1 °C
Voie 2 (inutilisée)	2	*****	16 bits	*****	*****
Voie 3 (inutilisée)	2	*****	16 bits	*****	*****

Emulation d'une communication réseau

1) But

Emuler partiellement le réseau BUS VENT

2) Objectifs : L'étudiant devra être capable de :

-D'exploiter la documentation sur le 68000, le protocole MODBUS RTU et le SIO68564 afin d'établir un échange d'information entre le répétiteur et le coffret MEA3C

- Définir les différents mécanismes utilisés lors d'une communication entre un maitre et un esclave sous le protocole MODBUS RTU

-D'analyser et interpréter à l'oscilloscope ou à l'analyseur de trame les différents échanges sur le bus de terrain

- De développer sur le logiciel PCVUE une page de supervision sur le poste maitre MODBUS

3) Prérequis

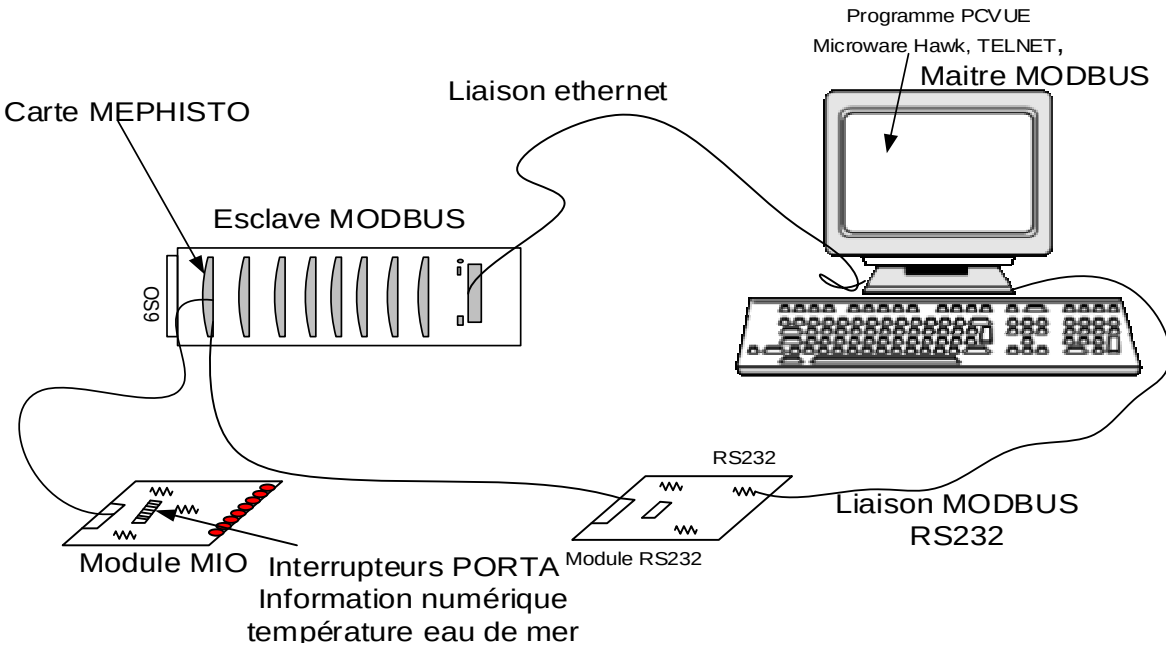
Connaissance du coupleur PIT 68230 et SIO68564 (cours).

Lecture du fascicule mode d'emploi de la carte MEFISTO.

Cour sur les bus de terrain, les liaisons séries et le langage C.

Dossier technique sur le réseau navigation

Schéma de câblage



4) Présentation

On souhaite émuler l'échange d'information entre le répétiteur et le coffret NMEA3C.

Cet échange d'information sera limité à l'acquisition de la température MER.

L'ordinateur et le logiciel PCVUE simulera le répétiteur en mode Maitre MODBUS RTU.

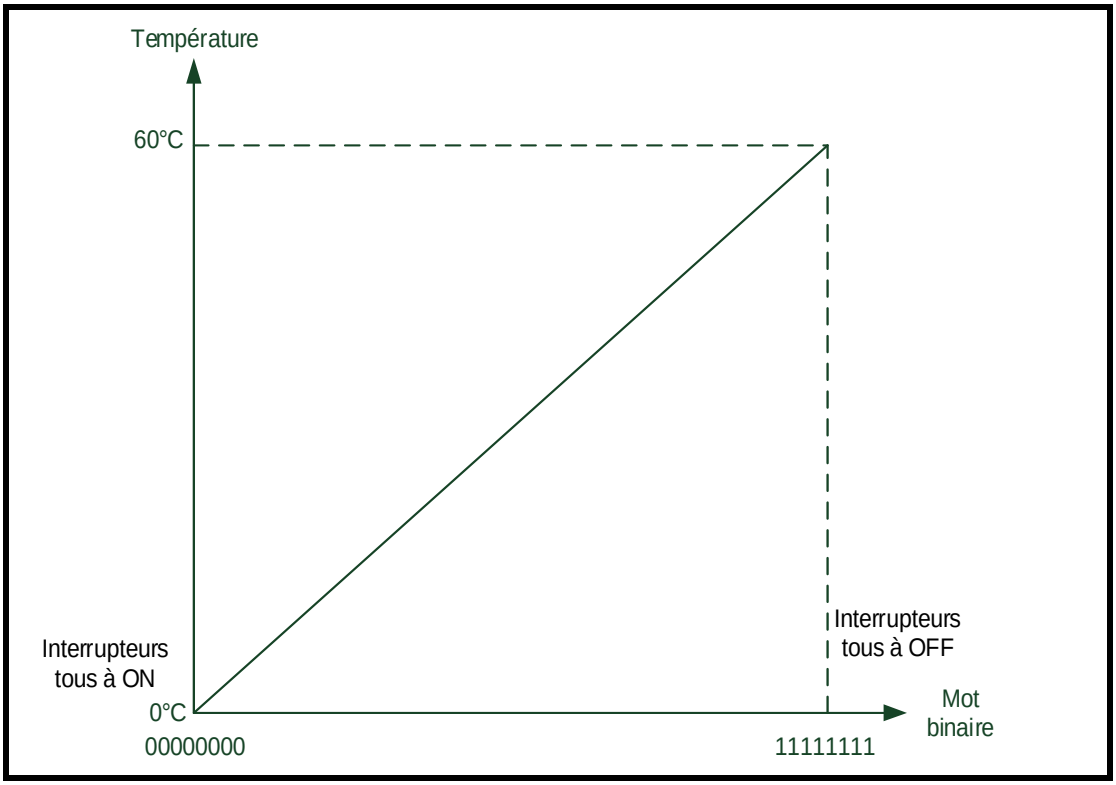
La cible OS9 simulera le coffret NMEA3C soit en mode esclave MODBUS RTU.

L'information température sera codée sur 2 octet :

Un octet de poids fort toujours à 0x00

Un octet de poids faible qui sera l'image de l'état des interrupteurs du PORTA.

La courbe qui donne la température en fonction du mot binaire est donnée ci-dessous :

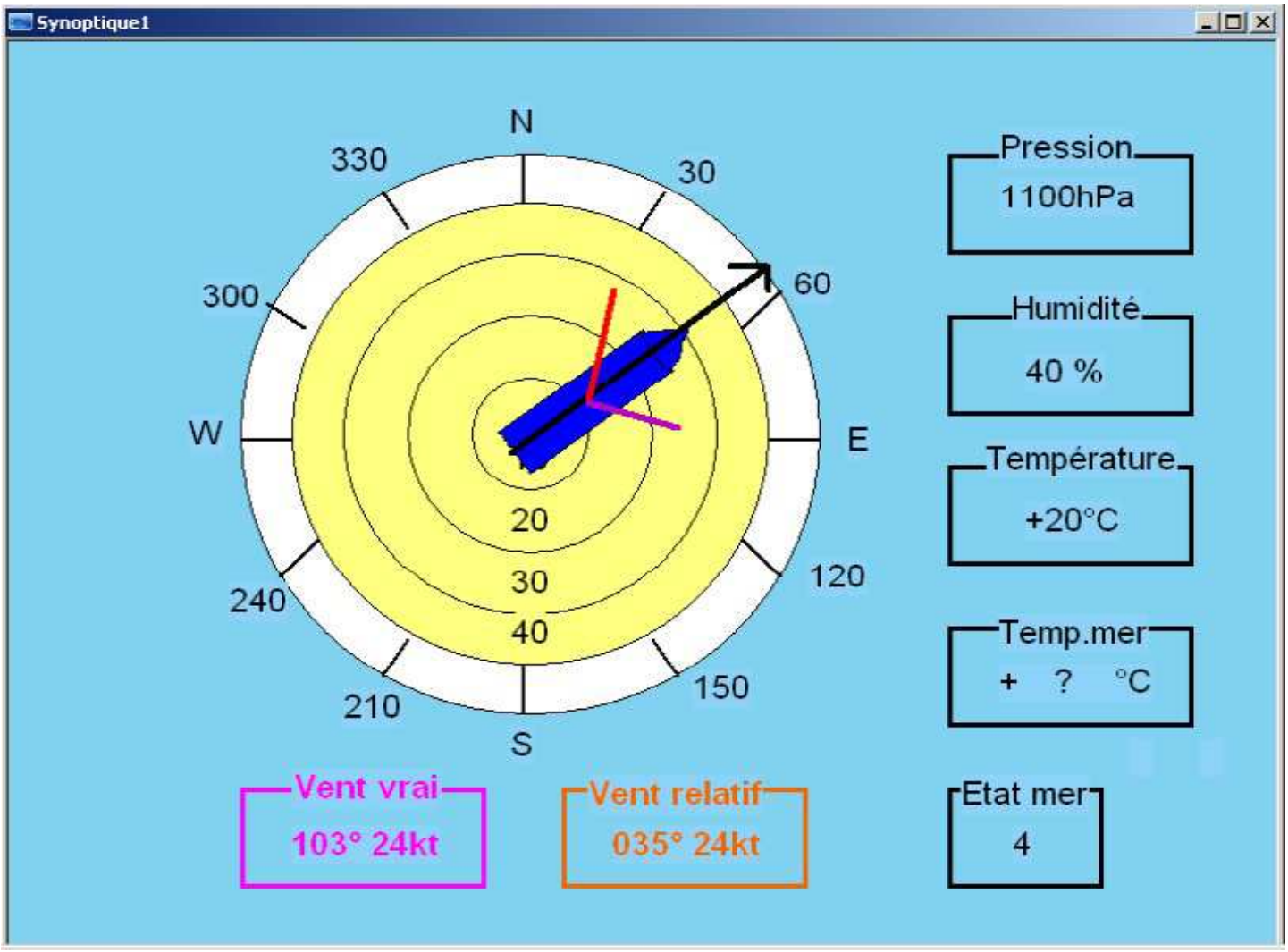


5) Questionnaire

on souhaite développer la communication en mode MODBUS RTU entre la cible OS9 (Coffret NMEA3C) et le PC(Répétiteur). Cette communication se fera en utilisant la liaison série RS232 et non la RS485

5.1) Première partie : La Communication MODBUS RTU

5.1.1) Réalisez sur l'ordinateur (Maitre MODBUS RTU) la page de supervision suivante (on s'aidera du tutorial fourni)



5.1.2) Le logiciel PCVUE envoie une seule trame MODBUS de requête auprès de la cible os9. Complétez le programme document réponse N°1 afin de réceptionner cette trame et d afficher chacun de ses octets. Testez votre programme en Lançant la communication sur le LOGICIEL PCVUE et en relevant les octets de la

trame qui circulent entre le maitre et l'esclave MODBUS RTU. Décrire les différents champs d'informations contenues dans la trame

A partir des informations visualisées en 5.1.2, On souhaite développer un programme qui permet à l'esclave de répondre à la requête température eau du maitre ou PCVUE

5.1.3) Complétez le tableau tab2 afin de fournir une réponse correcte à la requête précédente.

5.1.4) Le programme à compléter se trouve document réponse N°2. Expliquez dans ce programme la signification de la ligne de code suivante :

```
while(((sr0=get_sio_sr0(psio))&1)==0);
```

5.1.5) donnez le rôle de la fonction suivante :

```
init_pit(pointeurpit pit)
```

5.1.6) Complétez document réponse N°2 les lignes de code du programme et les commentaires afin d'émettre la trame de réponse. La valeur de la température mer sera lue sur le PORTA du pit68230 et transmise dans la trame de réponse octet de poids faible (aucun complément à 2). On n'oubliera pas de calculer les 2 Octets de CRC qu'il faut insérer en fin de trame.

int Modbuscrc16(char tab[], int longueur)

Paramètres d'entrées :	char tab[] tableau de caractère dont il faut calculer le CRC
	int Longueur Nombre de caractères utilisés pour calculer le CRC
Paramètres de sorties :	int CRC Valeur du CRC codé sur 16bits

5.1.7) Lancez la communication sur le LOGICIEL PCVUE et relevez à l'oscilloscope les trames qui circulent entre le maitre et l'esclave MODBUS RTU. En vous aidant du protocole MODBUS en déduire les différents champs d'informations qui

circulent sur la trame. Donnez le format de la transmission utilisée (débit, parité…..)

Document réponse N°1

```
/*****************************************************************
/*          Marine Nationale - Cin Saint Mandrier          *
/*      Cours BM      *
/*****************************************************************
/* Date de creation :22/02/2000
/* Fonction realisee:          gestion hyperterminal en
                               asynchrone a 9600 bauds
                          */
/*         programme permettant l'envoi de caracteres en fonction de
           l'etat de la ligne rts (entree cts du sio) et interdiction
           d'envoi de caracteres du pc->ms9 (gestion de la sortie rts
           du SIO ou cts du PC)          */
/*         pas d'it seulement par polling*/


#include <sio.h>
#include <pit.h>
#include "pointeur.h"
#include <stdio.h>
unsigned char crc_table[512];
unsigned int Modbuscrc16(unsigned char tabCRC[11],int Length);


int main()
  {

           unsigned char tab[10];
           long j=0;
           char recu='a';       /*init != de retour charriot*/
           int sr0;
           int i=0;
           int indicetab=0;
           unsigned int crc;
           pointeursio psio;
           pointeurpit pit;
           setbuf(stdin,NULL);
           setbuf(stdout,NULL);
           pit=adress_PIT;
           init_pit(pit);
           psio=adress_SIO;  /*psio pointeur sur port A du sio*/
           printf("esclave MODBUS RTU \n");
           printf("en attente de demande lecture \n");
                    while(1)                                          {
            indicetab=0;
/* récupération de la trame code commande lecture température eau */
                    for(i=0 ;_____ ;i++){
                    while(((sr0=get_sio_sr0(psio))&1)==0);
                    recu=_____;        /*lecture du caractere reçu*/
                    printf(_____) ;/* affichage du caractère */
                    }
                    printf(_____) ;/* sautez une ligne */
```

```
                }
        printf("\n");
        printf("trame bien recu \n");
        term_sio_sans(psio);
        return 0;
}

 unsigned int Modbuscrc16(unsigned char tabCRC[11],int Length)
{

int i;
int index;
unsigned char crc_Low = 0xFF;
unsigned char crc_High = 0xFF;
for (i = 0; i<Length; i++)
{
index = crc_High ^ (unsigned char)tabCRC[i];
crc_High = crc_Low ^ crc_table[index] ;
crc_Low = crc_table[index + 256];
}
return crc_High*256+crc_Low;

}

 init_pit(pointeurpit pit)
{
        P_GCR  = 0x00;     /* mode 00, H34 et H12 invalides */
        P_SRR  = 0x00;     /* Pas DMA, valide PIACK&PIRQ */
        P_ADDR = 0x00;   /* Port A en entree */
        P_ACR  = 0x80;     /* submode 1X, INIT=1 , pas d'it */
        P_BDDR = 0xff;     /* Port B en sortie */
        P_BCR  = 0x80;     /* submode 1x, impulsion, pas d'it */
        return((P_ADR)>>4);

}


void init_sio_sans(pointeursio psio)
{
        S_CR=0x48;                  /* reset du canal*/
        S_MCR=0x47;                 /* premier reg a programmer*/
        S_TCR = 0x04;     /* division horloge */
        S_BCR = 0x0D;               /*horloge interne+valide BRG*/
        S_ICR=0x00;                 /*interruption masquee*/
        S_XCR=0xC1;                 /* validation transmetteur*/
        S_RCR=0xC1;                 /*recepteur valide*/
}

void term_sio_sans(pointeursio psio)
        {
        S_ICR=0x00;                 /*interruption masquee*/
        S_XCR=(0x00);  /* invalidation transmetteur*/
        S_RCR=(0x00);     /*invalidation recepteur*/
        }

void wr_sio_sans(char var,pointeursio psio)
        {
                S_DR=var;
        }
```

```
char rd_sio_v24(pointeursio psio)
	{
		char recu;
		recu=S_DR;
		return recu;
	}



int get_sio_sr0(pointeursio psio)
	{
		return((int)(S_SR0));
	}
```

Document réponse N°2

```
/*****************************************************************
/*          *
/*      Cours BM     *
/*****************************************************************
/* Date de creation :22/02/2000
/* Fonction realisee:      gestion hyperterminal en
                           asynchrone a 9600 bauds
                                */
/*        programme permettant l'envoi de caracteres en fonction de
          l'etat de la ligne rts (entree cts du sio) et interdiction
          d'envoi de caracteres du pc->ms9 (gestion de la sortie rts
          du SIO ou cts du PC)       */
/*        pas d'it seulement par polling*/


#include <sio.h>
#include <pit.h>
#include "pointeur.h"
#include <stdio.h>
unsigned char crc_table[512];
unsigned int Modbuscrc16(unsigned char tabCRC[11],int Length);


int main()
  {

          char tab2[11]={0x07,0x04,0x06,0x00,0x80,0x00,0x00,0x00,0x00,0x23,0x32};
          unsigned char tab[10];
          long j=0;
          char recu='a';    /*init != de retour charriot*/
          int sr0;
          int i=0;
          int indicetab=0;
          unsigned int crc;
          pointeursio psio;
          pointeurpit pit;

          setbuf(stdin,NULL);
          setbuf(stdout,NULL);
          pit=adress_PIT;
          init_pit(pit); /*_____________________________________________________________*/
          psio=adress_SIO;          /*psio pointeur sur port A du sio*/
          printf("esclave MODBUS RTU \n");
          printf("en attente de demande lecture \n");
          init_sio_sans(psio);/*_____________________________________________________________*/


          while(1)                                                {
           indicetab=0;
            tab2[4]= P_ADR; /* _____________________________________________*/
             crc=Modbuscrc16(_____________________________________________);

          tab2[9]=(unsigned char)((crc&0xff00)>>8); ; /* _____________________________________________*/
          tab2[10]=(unsigned char) (crc&0x00ff); ; /* _____________________________________________*/
          /* réception de la trame de requête température eau */
          for(_____________________________________________){
                  while(((sr0=get_sio_sr0(psio))&1)==0); /*_____________________________________*/
                  recu=_____________________________;        /*lecture du caractere*/
                  tab[indicetab]=recu; /* Mémorisation du caractère reçu dans le tableau tab */
                  }
```

```
/* émission de la trame de réponse suite à la requête  température eau */

                                        for(i=0;i<11;i++)
                {
                                _____________________________________ ; /* émission d'un caractère sur la ligne RS232 */
                while(((sr0=get_sio_sr0(psio))&4)==0); /*_____________________________________________________*/

                }

                }
        printf("\n");
        printf("trame bien recu \n");
        term_sio_sans(psio);
        return 0;
}


 unsigned int Modbuscrc16(unsigned char tabCRC[11],int Length)
{

int i;
int index;
unsigned char crc_Low = 0xFF;
unsigned char crc_High = 0xFF;
for (i = 0; i<Length; i++)
{
index = crc_High ^ (unsigned char)tabCRC[i];
crc_High = crc_Low ^ crc_table[index] ;
crc_Low = crc_table[index + 256];
}
return crc_High*256+crc_Low;

}

 init_pit(pointeurpit pit)
{
        P_GCR  = 0x00;  /* mode 00, H34 et H12 invalides */
        P_SRR  = 0x00;  /* Pas DMA, valide PIACK&PIRQ */
        P_ADDR = 0x00;          /* Port A en entree */
        P_ACR  = 0x80;  /* submode 1X, INIT=1 , pas d'it */
        P_BDDR = 0xff; /* Port B en sortie */
        P_BCR  = 0x80;  /* submode 1x, impulsion, pas d'it */
        return((P_ADR)>>4);

}


void init_sio_sans(pointeursio psio)
{
        const unsigned short speed_9600=0x4;        /*vitesse 4800 bauds*/
        const unsigned short int mode_ctr=0xC4; /* mode async, 1 stop 0 parite %64*/
        const unsigned short int in_clk=0xd;        /*horloge interne+brg+/4*/

        S_CR=0x48;              /* reset du canal*/
        S_MCR=0x47;             /* premier reg a programmer*/
        S_TCR = 0x04;  /* division horloge */
        S_BCR = 0x0D;           /*horloge interne+valide BRG*/
        S_ICR=0x00;             /*interruption masquee*/
        S_XCR=0xC1;             /* validation transmetteur*/
        S_RCR=0xC1;             /*recepteur valide*/
}
```

```
void term_sio_sans(pointeursio psio)
        {
        S_ICR=0x00;             /*interruption masquee*/
        S_XCR=(0x00);  /* invalidation transmetteur*/
        S_RCR=(0x00);  /*invalidation recepteur*/
        }

void wr_sio_sans(char var,pointeursio psio)
        {
                S_DR=var;
        }

char rd_sio_v24(pointeursio psio)
        {
                char recu;
                recu=S_DR;
                return recu;
        }

int get_sio_sr0(pointeursio psio)
        {
                return((int)(S_SR0));
        }
```