A. Base de PHP

- Le code source PHP se trouvant sur le serveur, il n'est jamais vu par l'utilisateur
- Le script PHP doit avoir comme extenstion .php
- Le code doit être compris entre **<?php et ?>**
- Chaque instruction se termine par un point virgule « ; »

(1) Les Syntaxes de base

- Séparation des instructions : par « ; » comme en C.
- Commentaire : « // » comme en C++
- Les types de variables : booleen, entier, chaîne de caractères …
- MAIS, le type d'une variable n'est pas en général, déclaré par le programmeur. Il est décidé au moment de l'exécution par PHP : $foo="essai" ; est un chaîne de caractères

(2) Structure d'un script PHP

- Délimité par les balises d'échappement **<?php** et **?>**
- Composé d'instructions terminées par un point-virgule **;**
- Bloc d'instructions délimité par des accolades {}
- Associé à des commentaires :

// Commentaires de fin de ligne
/* Commentaires
longs */

(3) Les Variables PHP

les variables sont représentées par un signe dollar "$"

Contrairement à de nombreux langages de programmation, en PHP il ne faut pas prédéclarer une variable mais celle-ci est typée lors de son instanciation.

```
$x=1 indique que $x est un entier
$mot='test' indique que $mot est une chaîne
```

(4) Types PHP

- booléen TRUE FALSE (i.e. **0**,"","**0**")
- entier
- flottant
- chaîne de caractères (entre " ou ')
- tableau

(5) Les structures de contrôle

- if

```
if(condition)
{       instruction(s);}
```

else permet d'exécuter une autre instruction si la condition **if** n'est pas réalisée
else if permet d'enchaîner une série d'instructions et évite d'avoir à imbriquer des instructions **if**
Les boucles

- **while** permet d'exécuter une série d'instructions tant que la condition est vérifiée.

```
while (condition)
{       instructions(s);          }
```

- **for** permet d'exécuter une série d'instructions un nombre déterminé de fois.

```
<?php
for (Compteur; Condition; Incrémente)
{…}
```

- Compteur: nom de la variable qui sert de compteur.
- Condition: condition pour laquelle la boucle s'arrête
- Incrémente: instruction qui incrémente (ou décrémente) le compteur.

Exemple

```
<?php
for ($i=0; $i<10; $i++)
{…
}
```

(6) Les Tableaux

Exemple :

```
<?php
    $prenoms = array();
//on déclare le tableau de prénoms, remarquez que cela se fait avec la fonction array()
 /* On va initialiser les valeurs pour les indices 0, 1, 2  et 3
    */

    $prenoms[0] = 'Maurice';
    $prenoms[1] = 'Jean';
    $prenoms[2] = 'Pierre';
    $prenoms[3] = 'Paul';

    echo $prenoms[0]; //Va afficher "Maurice"
    echo $prenoms[1]; //Va afficher "Jean"

    ...
?>
```

B. Accès à la base de données mysql

- L'accès s'effectue en 4 étapes :
  1. Se connecter
  2. Exécuter des Requêtes
  3. Exploiter les résultats et traiter les erreurs
  4. Fermer de la connexion

(1) Connexion à un serveur MySQL

- **int mysql_connect(string serveur,string utilisateur,string secret)**
  - ✓ Etablir une connexion avec *serveur* (MySQL), pour un compte *utilisateur*, et de mot de passe secret.
  - ✓ Renvoie l' **identifiant de connexion** qui sera utilisé ensuite pour dialoguer avec la bdd.

Exemple :

```
$con=mysql_connect("192.168.109.10","user1","gigngtrrl");
```

(2) Sélectionner une Bdd

- **bool mysql_select_db(string base ,int identifiant de connexion)**
  - Permet de se placer dans une base de données.

1. **Exécuter une requête (1)**

- **int mysql_query(string code[,int identifiant de connexion])**
  - Adresse une requête SQL au serveur MySQL.
  - Requête de texte code
  - Renvoie un identificateur de résultat
    - ✓ Vrai si la requête est valide

Dans le cas d'une clause SELECT, MySQL retourne un identifiant de résultat à utiliser ultérieurement dans les opérations de consultation.

(3) Exploiter les résultats

- **array mysql_fetch_row(int result)**
  - Récupère une des lignes du résultat, et positionne le curseur sur la ligne suivante.
  - La ligne est représentée sous forme d'un *tableau* (une liste de valeurs).
  - Accès aux données :
    - ✓ **$Tableau[Indice]** où les indices correspondent aux champs utilisés dans la requête
  - NB: result= Résultat de requête
  -

- **array mysql_fetch_array(int result)**
  - transforme la ligne courante en un tableau associatif (clé/valeur) à indices numériques ou nommés
  - accès aux données : **$Tableau[Indice/Nom_Indice]** où les indices correspondent aux champs utilisés dans la requête
  -

Exemple dans le cas où la requête est "SELECT NumAvion,Type,Nom FROM avion";

```
while ($ligne = mysql_fetch_row($resultat)) {
echo "NumAvion : $ligne[0]<br>";
echo "Type : $ligne[1]<br>";
echo "Nom : $ligne[2]<br>";
```

(4) Récupérer TOUTES les lignes du résultat

- Problème :
  - **mysql_fetch_row, mysql_fetch_array,** récupèrent uniquement la première ligne du résultat
  - Il faut les appeler plusieurs fois
- Solution :
  - Utiliser une boucle **(while ou for)**

Exemple :

```
while ($ligne = mysql_fetch_row($resultat)) {...}
```

C.La gestion des fichiers avec PHP

(1) Ouverture et fermeture

```
int fopen(chaine nomdufichier, chaine mode);
```

Cette fonction renvoie un identifiant.
Le mode indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture. Il s'agit d'une lettre (en réalité une chaîne de caractères) indiquant l'opération possible :
r (comme *read*) indique une ouverture en lecture seulement
w (comme *write*) indique une ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
a (comme *append*) indique une ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
Lorsque le mode est suivi du caractère + celui-ci peut être lu et écrit. Enfin, le fait de faire suivre le mode par la lettre b entre crochets indique que le fichier est traité de façon binaire.

```
void fclose(indentifiant) ;
```

(2) Lecture et écriture

Une fois que le fichier a été ouvert avec le mode désiré, il est possible de lire son contenu et d'y écrire des informations grâce aux fonctions :
fputs() (aussi parfois appelée fwrite(), les deux noms sont équivalents, on parle d'alias) permettant d'écrire une chaîne de caractères dans le fichier

```
entier fputs(entier Etat_du_fichier, chaine Sortie);
```

La fonction *fputs()* renvoie le nombre de caractères effectivement écrits dans le fichier
fgets() permettant de récupérer une ligne du fichier

```
chaîne fgets(entier Etat_du_fichier, entier Longueur);
```

Le paramètre *Longueur* désigne le nombre de caractères maximum que la fonction est sensée récupérer sur la ligne. La fonction *fgets()* renvoie 0 en cas d'échec, la chaîne dans le cas contraire

1. Functional description

The 10/100BASE-T(X) ports of an RS2-5TX(/FX) represent a terminal connection for the connected LAN segment. You can connect single devices or complete network segments.

1.1 FRAME SWITCHING FUNCTIONS

Store and Forward
All data received by the RS2-5TX(/FX) from the system bus or at the ports is stored and checked for validity. Invalid and defective frames (> 1,518 bytes or > 1,522 bytes VLAN frames or CRC error) as well as fragments (< 64 bytes) are discarded. The RS2-5TX(/FX) forwards the valid frames.

Multi address capability
An RS2-5TX(/FX) learns all source addresses per port. Only packets with
– unknown addresses
– addresses learnt at this port
– a multi/broadcast address
in the destination address field are sent to this port.
An RS2-5TX(/FX) learns up to 1,000 addresses. This becomes necessary if more than one terminal device is connected to one or more ports. In this way several independent subnetworks can be connected to an RS2-5TX(/FX).

Learnt addresses
An RS2-5TX(/FX) monitors the age of the learned addresses. The RS2-5TX(/FX) deletes address entries from the address table which exceed a certain age (300 seconds).
Note: Restarting deletes the learned address entries.

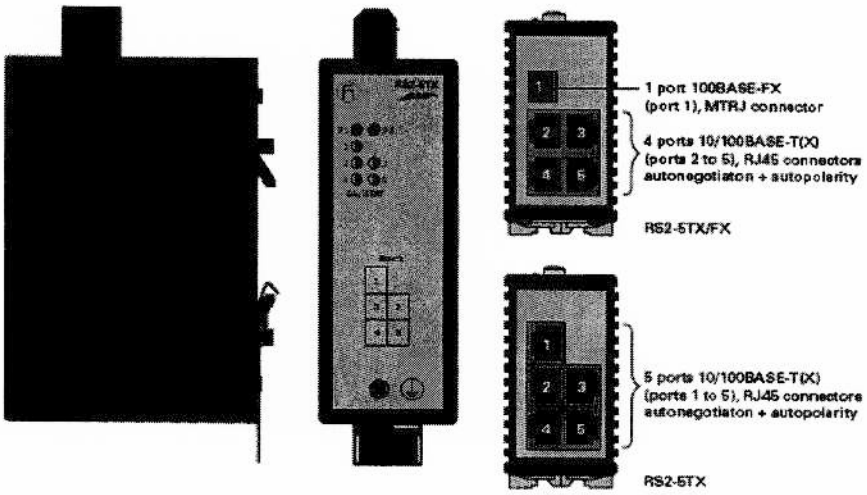
Tagging (IEEE 802.1Q)
The IEEE 802.1 Q standard designates the VLAN tag to be included in a MAC data frame for the VLAN and prioritizing functions. The VLAN tag consists of 4 bytes (2 bytes tag protocol identifier TPID, 2 bytes tag control information TCI). It is inserted between the source address field and the type field. Data packets with a VLAN tag are transmitted unchanged by the RS2-5TX(/FX).

1.2 SPECIFIC FUNCTIONS OF THE TP/TX INTERFACE

Link control
The RS2-5TX(/FX) monitors the connected TP/TX line segments for short-circuits or interrupts using regular idle signals in accordance with IEEE standard 802.3 10/100BASE-T/TX. The RS2-5TX(/FX) does not transmit any data to a TP/TX segment from which it does not receive an idle signal.

Note: A non-occupied interface is assessed as a line interrupt. The TP/TX line to terminal equipment which is switched off is likewise assessed as a line interrupt as the de-energised bus coupler cannot transmit idle signals.

Auto polarity exchange
If the receive line pair is incorrectly connected (RD+ and RD- switched) polarity is automatically reversed.



Fiig. 1: Overview display elements and interfaces of the RS2-5TX(/FX)

1.3 SPECIFIC FUNCTIONS OF THE F/O INTERFACE

Link control (RS2-5TX/FX)
The RS2-5TX/FX monitors the connected F/O line for interrupts using idle signals during frame pauses in accordance with IEEE standard 802.3 100BASE-FX. The RS2-5TX/FX transmits no data to an F/O line from which it is receiving no idle signal.
Low Light Detection: If the optical input power decreases below the low light threshold the transmit and receive path will be disabled for data and the idle signal will be transmitted.

1.4 FURTHER FUNCTIONS AND FEATURES

Reset
The RS2-5TX(/FX) will be reset by the following action:
– input voltages fall below a threshold

After a reset the following action is carried out:
– initialization

1.5 DISPLAY ELEMENTS

Equipment status
These LEDs provide information about statuses which affect the function of the entire RS2-5TX(/FX).
P1 – Power 1 (green LED)
– lit: – supply voltage 1 present
– not lit: – supply voltage 1 less than 9.6 V

P2 – Power 2 (green LED)
– lit: – supply voltage 2 present
– not lit: – supply voltage 2 less than 9.6 V

Port Status
These LEDs display port-related information.
DA/STAT 1 to 5 – Data, Link status (green LED)
– not lit: – no valid link
– flashes: – data activity
– lit: – valid link