

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve Etude d'un Système Informatisé

Système d'acquisition de température de bobinages

Sujet – première partie : analyse d'éléments du système

Durée : 1h30

Coefficient 1,5/5

*"Calculatrice autorisée (conformément à la circulaire n° 99-186 du 16 novembre 1999)."
"Le candidat peut utiliser tous les documents personnels qu'il estime nécessaire."*

Ce document comprend 8 pages composées de :

Sujet : pages 1 à 6 (papier blanc)

Documents réponses : pages 7 à 8 (papier de couleur)

à rendre obligatoirement (même vierge) avec la copie

A. Principe de la mesure

On s'intéresse au principe de la mesure tel qu'il a été présenté dans les paragraphes 1-3-1 à 1-3-3 du dossier technique.

Chaque multimètre a la charge de la mesure de la résistance de 4 enroulements (voir figure 4 du dossier technique). Chaque point de mesure tel que présenté dans le paragraphe 1-3-3 est en réalité un nuage composé de cinq points.

Le poste de mesure accomplit donc, lorsque les enroulements sont arrivés à une température stable, les actions suivantes :

- coupure de l'alimentation des enroulements ;
- commutation d'un relais ;
- attente de la stabilisation ;
- relevé rapide de cinq mesures ;
- commutation du relais suivant...

Le relevé des valeurs de résistance R est effectué en utilisant un mode d'acquisition rapide du multimètre. On peut considérer les cinq valeurs acquises par le multimètre comme simultanées.

Question A.1.1

Proposez un type de données structuré susceptible de recevoir les informations de résistance et de temps relatives à cet ensemble.

Après coupure de l'alimentation des relais, la loi de variation en fonction du temps de la valeur de la résistance mesurée est, comme exposé dans le dossier technique, une exponentielle décroissante d'équation $R = A * e^{-t/\tau} + R_i \dots$

Question A.1.2

En utilisant les rappels mathématiques du dossier technique et en supposant pour τ la valeur de 3600 secondes suggérée par le dossier technique, et pour A la valeur 3600 (pour avoir $A/\tau=1$), calculez la pente de cette exponentielle au bout de 0 secondes, de 120 secondes, de 300 secondes .

On assimile cette exponentielle décroissante sur les 120 premières secondes à la droite d'équation $R = -(A/\tau) * t + (A + R_i)$ tangente à l'origine, comme cela avait été proposé dans le dossier technique.

Question A.1.3

Quelle erreur (en %) commet-on sur la mesure à $t=120$ secondes, et à $t=300$ secondes en utilisant l'équation de la droite au lieu de l'exponentielle ?
Justifiez alors la simplification proposée ?

Au cours de ces 120 secondes, chacun de ces bobinages fait l'objet de trois séries de mesures. De ces trois couples (R_0, t_0) , (R_1, t_1) et (R_2, t_2) on détermine l'équation de la droite $R=f(t)$.

Question A.1.4

En utilisant les formules explicitées dans le dossier technique, exprimez la valeur de la résistance à chaud A de l'enroulement en fonction de R_0, R_1, R_2, t_0, t_1 et t_2 .

B. Analyse du système

B.1 Modélisation UML de la mesure : diagramme de classe

En utilisant la diagramme de classes (figure : 10 page 13) du dossier technique répondre aux questions suivantes.

Question B.1.1

Que signifie la relation entre la Classe « CIEEE » et la classe « CGPIB » ?

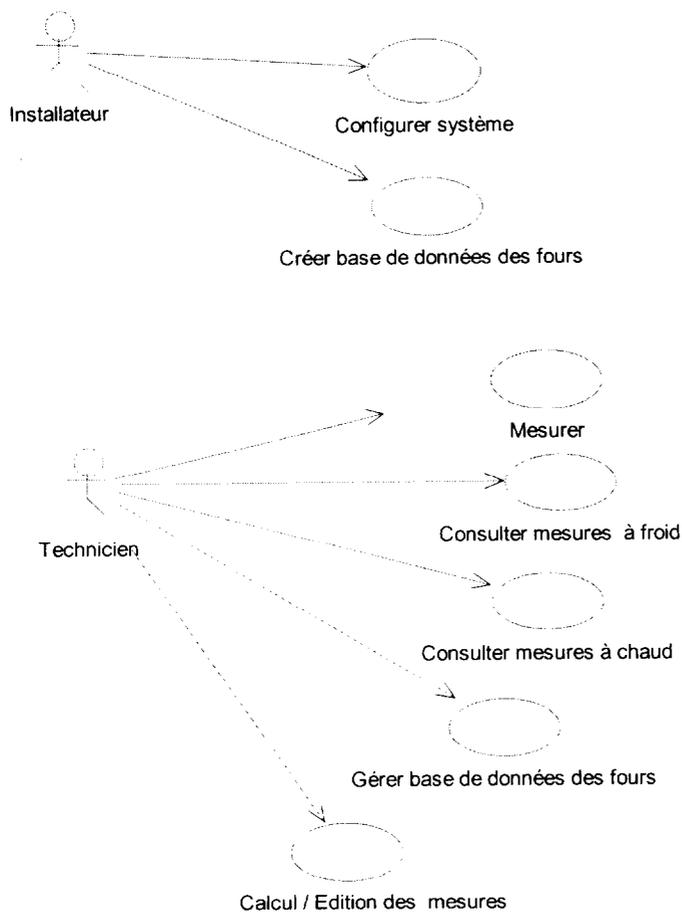
Question B.1.2

fd est l'identificateur de périphérique qui permet de communiquer avec la carte IEEE.
Pourquoi l'attribut « fd » de la classe « CIEEE » est-il déclaré « protégé » ?

Question B.1.3

La Classe « CIEEE » est une classe abstraite. Qu'est ce que cela implique au niveau de ses méthodes ?

Plusieurs cas d'utilisations (extraits) peuvent-être définis :



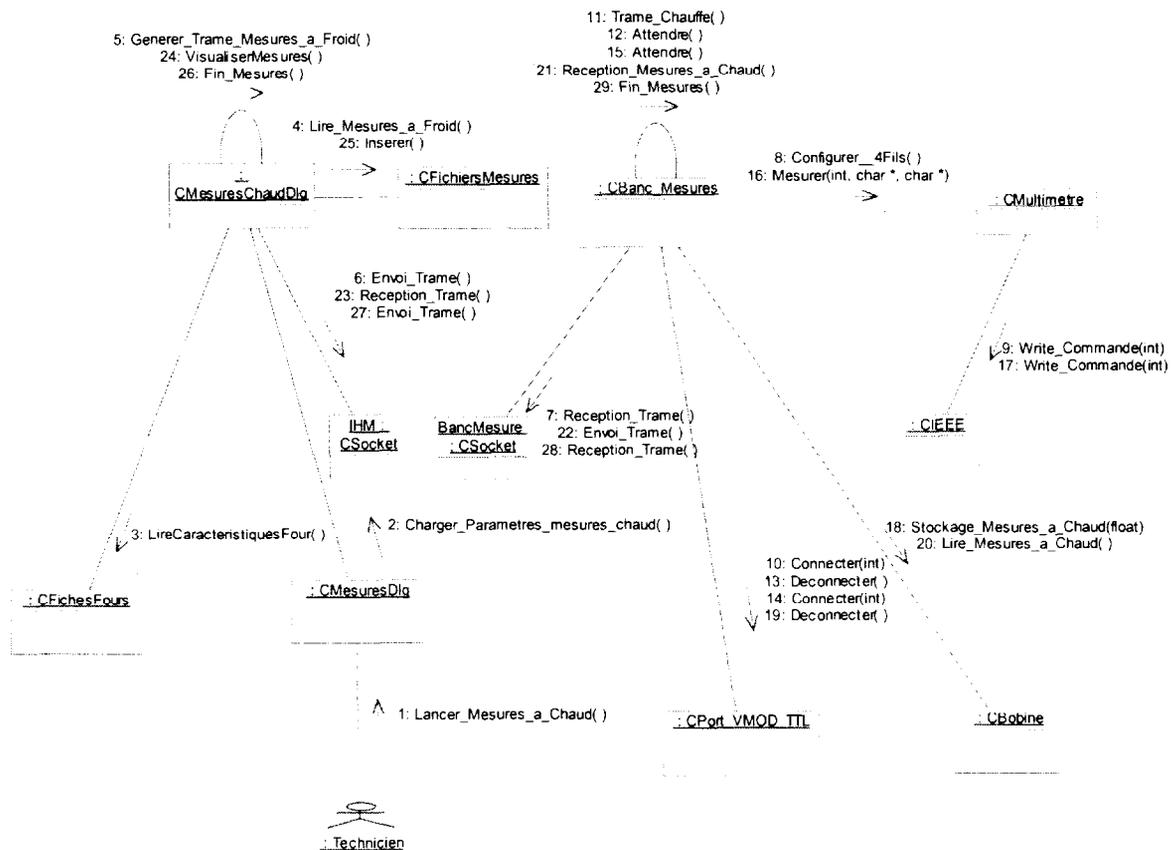
Le cas d'utilisation « Mesurer » induit plusieurs scénarii concrétisés par les diagrammes de séquence suivants :

- « Mesurer_a_Froid »
- « Mesurer_a_Chaut »

Nous allons étudier le scénario « Mesurer_a_Chaut ».

Un scénario est modélisé avec un diagramme de collaboration et un diagramme de séquence.

Celui-ci après analyse a permis de réaliser le diagramme de collaboration suivant :



Question B.1.4 – Répondez directement sur le document Réponse.
 A partir du diagramme de classe (figure : 10 page 13) du dossier technique et du diagramme de collaboration précédent, compléter le diagramme de séquence du scénario « Mesurer_a_Chaut » (Indiquer les messages manquants sur les flèches du diagramme de séquence, ils sont au nombre de 8 et signalés par des < ? >)

Question B.1.5
 Sachant qu'un bobinage se comporte comme un circuit **RL**, dont on veut contrôler la valeur de la résistance **R**, justifiez le message n° 15 du diagramme de collaboration.

C. Analyse des ressources

L'ensemble de l'étude ne s'est intéressé par souci de simplification qu'à un banc de mesure. Cependant, le système est capable de prendre en charge jusqu'à quatre bancs de mesure.

C.1 Ressources partagées

Question C.1.1 – Répondez directement sur le document Réponse.

Parmi les matériels ci dessous, signalez ceux qui sont dédiés à chaque banc de mesure, ceux qui sont partagés entre les bancs, en indiquant le nombre de bancs qui y accèdent ?

	Dédié	Partagé	Nombre de bancs
Sonde PT100			
Multimètre IEEE 488			
VMOD-RL			
VMOD-TTL/O			
VMOD-GPIB			
Carte VMOD-IO			
Carte VMOD-BAB			
Port RS-232			
Port Ethernet			

C.2 Temps réel

Le diagramme des classes présenté dans le dossier technique et revu dans la première partie du sujet fait apparaître que la classe **CBanc_Mesures** dérive de la classe **CThread**, qui possède toutes les méthodes nécessaires à la gestion et à l'exécution d'un thread.

Question C.2.1

Quel est l'intérêt de cette héritage (dérivation) ? En quoi son utilisation facilite-t-elle la gestion simultanée de plusieurs bancs de mesure ?

Question C.2.2

Quelles sont d'après vous les ressources qui devront être protégées d'un accès concurrent par plusieurs bancs de mesure ? Quelle technique proposez vous pour assurer la protection de ces ressources ?

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve Etude d'un Système Informatisé

Système d'acquisition de température de bobinages

Sujet – première partie : Document Réponse N°1

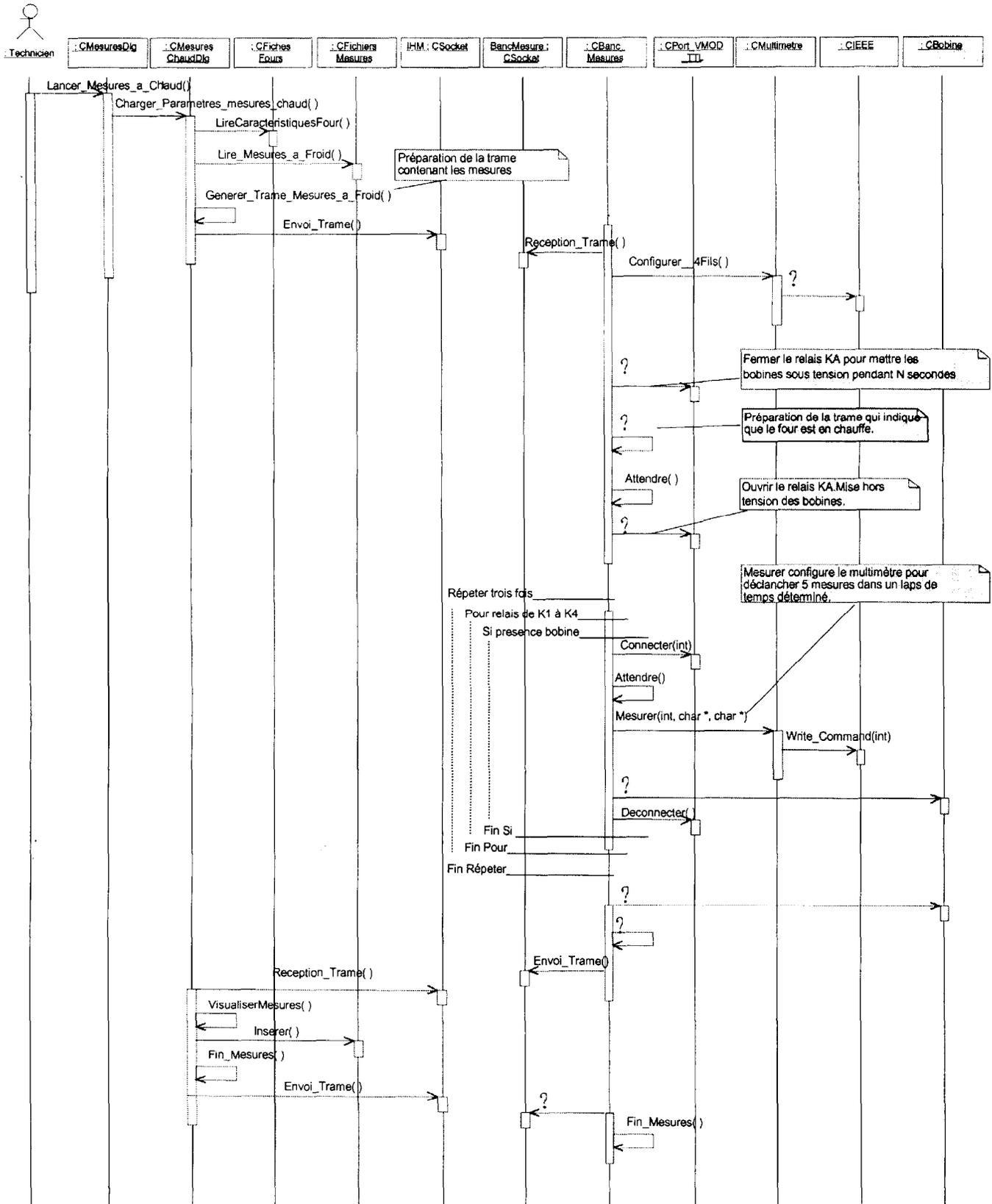
Réponse C.1.1

	Dédié	Partagé	Nombre de bancs
Sonde PT100			
Multimètre IEEE 488			
VMOD-RL			
VMOD-TTL/O			
VMOD-GPIB			
Carte VMOD-IO			
Carte VMOD-BAB			
Port RS-232			
Port Ethernet			

Epreuve Etude d'un Système Informatisé

Système d'acquisition de température de bobinages

Sujet – première partie : Document Réponse N°2



BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve Etude d'un Système Informatisé

Système d'acquisition de température de bobinages

Sujet – deuxième partie : conception d'éléments du système

Durée : 4h30

Coefficient 3,5/5

*"Calculatrice autorisée (conformément à la circulaire n° 99-186 du 16 novembre 1999)."
"Le candidat peut utiliser tous les documents personnels qu'il estime nécessaire."*

Ce document comprend 12 pages composées de :

Sujet : pages 1 à 10 (papier blanc)

Documents réponses : pages 11 à 12 (papier de couleur)

à rendre obligatoirement (même vierge) avec la copie

Toutes vos réponses doivent être accompagnées d'une justification.

D. Architecture matérielle

D.1 Etude des caractéristiques du bus IEEE 488

Une documentation partielle du bus est fournie en annexe D4.

Question D.1.1 – Répondez directement sur le document Réponse.

Complétez le tableau de configuration donnant les fonctions et les adresses de chacun des appareils.
(de nombreuses réponses sont possibles pour les adresses)

	Rack de commutation	Multimètre 1	Multimètre 2	Multimètre 3	Multimètre 4
Fonction	Ecouteur ou parleur				
Adresse	7				
Code Ecouteur	\$27				
Code Parleur	\$47				

Nota : On supposera le bit de parité toujours à 0

La chaîne de caractères *IDN? est envoyée sur le bus à partir du rack de commutation vers l'un des multimètres.

Question D.1.2

Quelles sont les valeurs binaires présentes sur les lignes de données au moment de l'envoi du caractère * (code ASCII 42 décimal) ?

Question D.1.3

Quelle sont les informations contenues dans la chaîne de caractères renvoyée par le multimètre concerné ?

D.2 Adressage des cartes

L'organisation de l'espace adressable est fixée par la carte CPU BAB40. L'annexe S1 donne une documentation partielle de la carte BAB40.

Question D.2.1

Quelle est dans cet espace la plage d'adresses disponible pour le bus VME (première et dernière adresse) en accès court (données sur 8 ou 16 bits) et la capacité (nombre d'octets) correspondante ?

La carte d'extension VMOD-IO est connectée sur le bus VME (voir Figure 8 du dossier technique).

Un extrait de la documentation de la carte VMOD-IO est fourni en Annexe S2.

Le Jumper J4 est fixé en position 2-3. Les commutateurs S2 et S1 sont positionnés respectivement à C et 8.

Question D.2.2

Quelles sont les lignes d'adresse (A15, A14...) fixées par les commutateurs S2 et S1 ?

Question D.2.3

Quelle est la plage d'adresses relatives utilisées par la carte VMOD-IO et la capacité correspondante ?

Question D.2.4

Quel est l'espace d'adresse absolu utilisé par la carte VMOD-IO dans le plan mémoire fixé par la carte CPU BAB 40 ?

Le module VMOD GPIB intègre un circuit contrôleur de bus IEEE 488 (bus GPIB) μ PD7210 dont une documentation partielle est donnée en annexe S4.

Question D.2.5

Quelle est l'espace d'adresse absolu utilisé par les 16 registres internes du contrôleur μ PD7210 ?

D.3 Etude du module VMOD-RL

Les 12 sorties du module VMOD-TTL/O sont reliées aux 2 modules VMOD-RL (6 sorties par module) de la carte d'extension VMOD-IO (voir Figure 8 du dossier technique).

Ces 12 sorties correspondent aux 8 lignes du Port B et aux 4 lignes du port C du CIO Z8536 intégré au module VMOD-TTL/O et sont accessibles par un connecteur 20 broches (voir page 3 de

l'annexe S3). Elle permettent de programmer les commutations du 230V, des bobines du four, de la sonde de température et du multimètre connecté (voir Figure 9 du dossier technique).

Le schéma structurel du module VMOD-RL est donné annexe S6.

Question D.3.1

Comment doit-on configurer les cavaliers J1 à J6 pour que le module VMOD-RL soit relié au port B du CIO Z8536 ?

Question D.3.2

Quel est le rôle du circuit ULN2003 (Documentation en Annexe S7)?

Question D.3.3

Quel niveau logique doit être présent en sortie du circuit ULN2003 pour alimenter les bobines des relais.

Question D.3.4

Quel est le niveau logique présent en entrée des circuits 74LS04 en cas de déconnexion de la carte ?
Dans quel état sont alors les relais ?
Quel est le rôle des circuits 74LS04 ?

Question D.3.5

Sachant que les commutateurs sont représentés au repos, quelle valeur doit t-on écrire sur le port B pour commuter la bobine 3 du four sur le multimètre ?

E. Programmation

E-1 Utilisation du matériel IEEE

Les questions qui suivent s'intéressent à la manipulation de la carte VMOD-GPIB. Ce module enfichable conforme à la norme IEEE 488-2 permet le pilotage des multimètres qui mesurent les résistances des bobines, et effectue la récupération des données de ces multimètres.

Lors de la mise en route de l'application, on devra initialiser le driver et créer le périphérique (device) correspondant au module GPIB (voir Annexe S5). Une fois ces opérations effectuées, l'application, pour accéder au module GPIB doit obtenir un descripteur de fichier de niveau 1, en utilisant la fonction `open()` de la bibliothèque standard avec comme paramètre le nom utilisé lors de la création du périphérique (device). Cette ouverture se fait en mode lecture seule, bien que les opérations d'écriture ultérieures soient autorisées.

Question E.1.1

En vous référant à la documentation fournie en annexe S8, et au schéma décrit dans le dossier technique, proposez un prototype pour la méthode `CGPIB ::Open()` qui :

- initialise le driver GPIB,
- crée le device correspondant,
- obtient le descripteur de fichier nécessaire pour la suite des opérations.

Vous tiendrez compte pour établir ce prototype des variations possibles d'implantation du module et de la carte support ?

Nota : Le vecteur d'interruption est fixé à **0x80** et le niveau d'interruption à **5**. Ces valeurs sont indicatives, et ne sont jamais utilisées dans le reste du sujet.

Question E.1.2

Proposez une solution complète en C++ pour cette méthode ?

Question E.1.3

Quels sont d'après le dossier technique et les questions D.3.1 à D.3.5 les valeurs des paramètres qu'il faudra passer à cette méthode?

Une fois le descripteur de fichier obtenu, l'envoi de la chaîne ***IDN?** vers l'adresse présumée d'un multimètre comme vu plus haut, permet de s'assurer de la présence d'un multimètre.

On souhaite pouvoir réinitialiser un multimètre et vider éventuellement la file d'attente des messages d'erreur de l'appareil.

Question E.1.4

Quelle fonction en langage C de la bibliothèque GPIB peut-on utiliser pour envoyer cette commande au multimètre ? A l'aide de cette fonction, quelle chaîne de caractères doit on envoyer au multimètre ? Quel sera l'effet de l'ajout de la commande ***OPC?** à la fin de cette chaîne ?

E-2 Analyse des résultats

Soit la structure :

```
#define NB_FASTACQUIRE 5

struct Measure
{
    long    m_Time ;
    float   m_Res[NB_FASTACQUIRE] ;
} ;
```

Cette structure reçoit les données brutes liées à une acquisition du multimètre. On souhaite ajouter à cette structure une information susceptible de recevoir la moyenne de ces mesures, en mettant de côté systématiquement la plus petite et la plus grande des 5 mesures, pour ne garder que la moyenne des trois valeurs centrales.

Question E.2.1

Proposez une modification de la structure Measure qui prend en compte cet ajout.

Question E.2.2

Proposez un codage en C++ de la fonction **float CBobine::GetCentralValue(...)** qui prend en argument une référence à une structure Measure, met à jour cette structure, et renvoie la valeur centrale.

Question E.2.3

Quel est l'intérêt de la constante **NB_FASTACQUIRE** ?

F. Communication et réseau

F.1 Liaison série RS232

La liaison RS232 est seulement utilisée en phase d'initialisation pour configurer différents paramètres réseau du rack.

Le constructeur donne les indications suivantes pour le raccordement d'un terminal ou d'un ordinateur à la carte CPU BAB40 :

- liaison sur 3 fils
- 9600 bit/s
- 8 bits de données
- parité paire
- 1 bit d'arrêt
- contrôle de flux Xon/Xoff

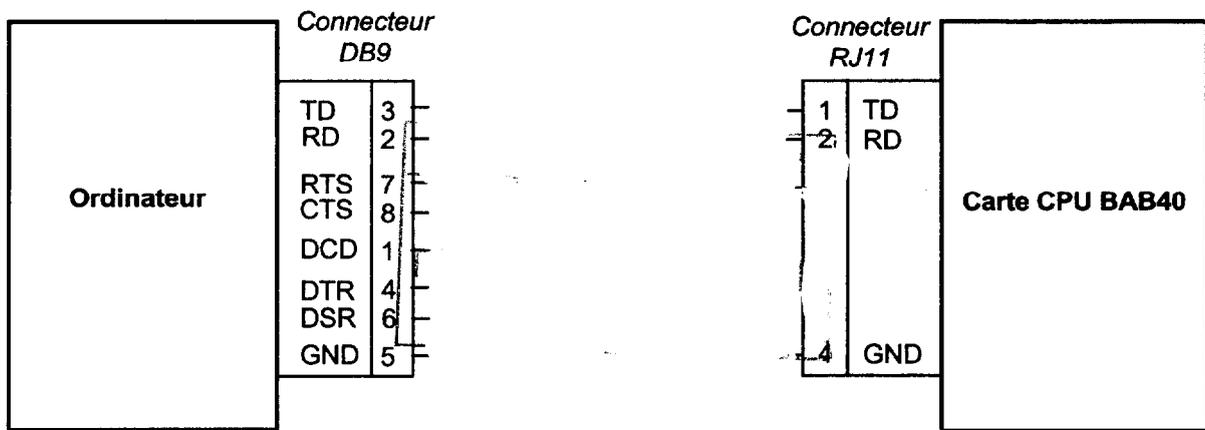
Question F.1.1

La liaison est-elle de type asynchrone ou synchrone ?

Physiquement, la liaison est réalisée à l'aide d'un câble muni d'un connecteur DB9 côté ordinateur et d'un connecteur RJ11 côté carte CPU.

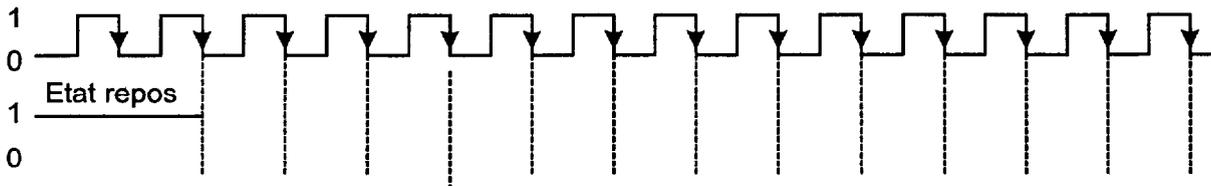
Question F.1.2 – Répondez directement sur le document Réponse.

Complétez le schéma de connexion entre l'ordinateur et la carte CPU.



Question F.1.3 – Répondez directement sur le document Réponse.

Représenter le chronogramme correspondant à l'émission du caractère 'A' (code ASCII 41H) en faisant figurer les différents bits de la trame (bit de départ, bit de données...) à partir d'un état de repos au niveau haut.



Question F.1.4

Quel est le temps minimum nécessaire pour transmettre, lors de mise sous tension du rack, la chaîne de caractères «VxWorks System Boot»?

Question F.1.5

Expliquez le principe du contrôle de flux utilisé sur cette liaison.

F.2 Organisation du réseau

Le réseau mis en œuvre est à base d'une architecture Ethernet 10baseT. On utilise le service FTP de TCP/IP pour le transfert de fichiers (voir Figure 2 du dossier technique).

Question F.2.1 – Répondez directement sur le document Réponse.

Complétez le tableau en donnant le numéro de la couche du modèle OSI concernée par les différentes entités ou protocoles présents sur le réseau utilisé.

Entité/protocole	Câble UTP	Routeur ADSL	Ethernet	Connecteur RJ45	TCP	IP	FTP
Couche(s)	1						

Le réseau possède une adresse IP de classe C : 195.212.5.0

Question F.2.2 – Répondez directement sur le document Réponse.

Proposez un plan d'adressage pour les différentes machines connectées.

	Routeur	Serveur	PC de contrôle	Rack VME
Adresse IP				
Masque				

F.3 Dialogue FTP

La configuration réalisée à l'aide de la liaison RS232 est mémorisée en RAM non volatile. Dans cette configuration les paramètres réseau du rack ont été donnés :

- adresse IP du rack
- adresse IP du serveur
- chemin et nom du fichier à télécharger
- nom et mot de passe pour la connexion FTP

La deuxième phase d'initialisation consiste à télécharger à partir du serveur le fichier correspondant au module du noyau VxWorks. Le client FTP est donc le rack qui demande automatiquement, après mise sous tension, une connexion FTP anonyme avec le serveur et le chargement du fichier VxWorks.

Les deux trames reproduites ci-dessous sont extraites d'un relevé réalisé par un analyseur de protocole lors de la séquence de démarrage à la mise sous tension du rack.

L'analyse présente, pour les trames 17 et 18, la description des différents protocoles utilisés : Ethernet, IP, TCP et FTP.

Pour chaque protocole, la première ligne donne les caractéristiques principales, les lignes suivantes les valeurs et la signification des différents champs.

La dernière partie donne pour chaque trame le contenu brut en hexadécimal suivi d'une représentation ASCII.

Network Monitor trace Mon 06/05/98 16:02:32 BootVxWoks.TXT

```
*****
Frame   Time      Src MAC Addr  Dst MAC Addr  Protocol  Description
17      10.944    00005B001BDB 0050046A8DCA  FTP       Req. from Port 1024
```

+ FRAME: Base frame properties

```
ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
+ ETHERNET: Destination address : 0050046A8DCA
+ ETHERNET: Source address : 00005B001BDB
ETHERNET: Frame Length : 96 (0x0060)
ETHERNET: Ethernet Type : 0x0800 (IP: DOD Internet Protocol)
ETHERNET: Ethernet Data: Number of data bytes remaining = 82 (0x0052)
```

IP: ID = 0x3EE; Proto = TCP; Len: 82

```
IP: Version = 4 (0x4)
IP: Header Length = 20 (0x14)
```

```
+ IP: Service Type = 0 (0x0)
IP: Total Length = 82 (0x52)
IP: Identification = 1006 (0x3EE)
```

```
- IP: Flags Summary = 0 (0x0)
  IP: .....0 = Last fragment in datagram
  IP: .....0. = May fragment datagram if necessary
```

```
IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 30 (0x1E)
IP: Protocol = TCP - Transmission Control
IP: Checksum = 0x174F
```

```
IP: Source Address = 195.212.5.24
IP: Destination Address = 195.212.5.1
IP: Data: Number of data bytes remaining = 62 (0x003E)
```

TCP: .AP..., len: 42, seq: 320061-320102, ack: 207440213, win: 4096, src: 1024, dst: 21 (FTP)

```
TCP: Source Port = 0x0400
TCP: Destination Port = FTP [control]
TCP: Sequence Number = 320061 (0x4E23D)
TCP: Acknowledgement Number = 207440213 (0xC5D4955)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
```

```
+ TCP: Flags = 0x18 : .AP...
TCP: Window = 4096 (0x1000)
TCP: Checksum = 0x5528
TCP: Urgent Pointer = 0 (0x0)
TCP: Data: Number of data bytes remaining = 42 (0x002A)
```

```
FTP: Req. from Port 1024, 'RETR tornado\target\config\bab40\vxworks'
FTP: FTP Command = RETR
FTP: FTP Data: Number of data bytes remaining = 38 (0x0026)
```

```
00000: 00 50 04 6A 8D CA 00 00 5B 00 1B DB 08 00 45 00 .P.j....[.....E.
00010: 00 52 03 EE 00 00 1E 06 17 4F C3 D4 05 18 C3 D4 .R.....O.....
00020: 05 01 04 00 00 15 00 04 E2 3D 0C 5D 49 55 50 18 .....=.]IUP.
00030: 10 00 55 28 00 00 52 45 54 52 20 74 6F 72 6E 61 ..U(..RETR torna
00040: 64 6F 5C 74 61 72 67 65 74 5C 63 6F 6E 66 69 67 do\target\config
00050: 5C 62 61 62 34 30 5C 76 78 77 6F 72 6B 73 0D 0A \bab40\vxworks..
```

```
*****
Frame   Time      Src MAC Addr  Dst MAC Addr  Protocol  Description
18      10.945    0050046A8DCA 00005B001BDB  FTP       Resp. to Port 1024
```

+ FRAME: Base frame properties

```
ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
```

```

+ ETHERNET: Destination address : 00005B001BDB
+ ETHERNET: Source address : 0050046A8DCA
  ETHERNET: Frame Length : 150 (0x0096)
  ETHERNET: Ethernet Type : 0x0800 (IP: DOD Internet Protocol)
  ETHERNET: Ethernet Data: Number of data bytes remaining = 136 (0x0088)
IP: ID = 0x1D7E; Proto = TCP; Len: 136
  IP: Version = 4 (0x4)
  IP: Header Length = 20 (0x14) + IP: Service Type = 0 (0x0)
  IP: Total Length = 136 (0x88)
  IP: Identification = 7550 (0x1D7E)
- IP: Flags Summary = 0 (0x0)
  IP: .....0 = Last fragment in datagram
  IP: .....1. = Cannot fragment datagram
  IP: Fragment Offset = 0 (0x0) bytes
  IP: Time to Live = 128 (0x80)
  IP: Protocol = TCP - Transmission Control
  IP: Checksum = 0x5B88
  IP: Source Address = 195.212.5.1
  IP: Destination Address = 195.212.5.24
  IP: Data: Number of data bytes remaining = 116 (0x0074)
TCP: .AP..., len: 96, seq: 207440213-207440308, ack: 320103, win: 8659, src: 21 (FTP),
dst: 1024
  TCP: Source Port = FTP [control]
  TCP: Destination Port = 0x0400
  TCP: Sequence Number = 207440213 (0xC5D4955)
  TCP: Acknowledgement Number = 320103 (0x4E267)
  TCP: Data Offset = 20 (0x14)
  TCP: Reserved = 0 (0x0000)
+ TCP: Flags = 0x18 : .AP...
  TCP: Window = 8659 (0x21D3)
  TCP: Checksum = 0xEECE
  TCP: Urgent Pointer = 0 (0x0)
  TCP: Data: Number of data bytes remaining = 96 (0x0060)
FTP: Resp. to Port 1024, '150 Opening BINARY mode data connection...'
  FTP: FTP Error Return Code = 150
  FTP: FTP Command Arg1 = Opening
  FTP: FTP Data: Number of data bytes remaining = 85 (0x0055)

00000: 00 00 5B 00 1B DB 00 50 04 6A 8D CA 08 00 45 00  ..[....P.j....E.
00010: 00 88 1D 7E 40 00 80 06 5B 88 C3 D4 05 01 C3 D4  ...~@...[.....
00020: 05 18 00 15 04 00 0C 5D 49 55 00 04 E2 67 50 18  ....]IU...gP.
00030: 21 D3 EE CE 00 00 31 35 30 20 4F 70 65 6E 69 6E  !.....150 Openin
00040: 67 20 42 49 4E 41 52 59 20 6D 6F 64 65 20 64 61  g BINARY mode da
00050: 74 61 20 63 6F 6E 6E 65 63 74 69 6F 6E 20 66 6F  ta connection fo
00060: 72 20 74 6F 72 6E 61 64 6F 5C 74 61 72 67 65 74  r tornado\target
00070: 5C 63 6F 6E 66 69 67 5C 62 61 62 34 30 5C 76 78  \config\bab40\vx
00080: 77 6F 72 6B 73 28 34 37 32 33 36 39 20 62 79 74  works(472369 byt
00090: 65 73 29 2E 0D 0A                                     es)...
```

Question F.3.1

Indiquez les adresses Ethernet et les adresses IP du serveur et du rack.

Question F.3.2

Donner pour la trame 17 le nombre d'octets total et le nombre d'octets correspondant à chaque protocole.

Question F.3.3

Les paquets IP sont-ils fragmentés ? Justifiez votre réponse.

Question F.3.4

Quelles sont les commandes et les réponses FTP utilisées dans les trames 17 et 18? Quelle partie du dialogue est représentée par ces deux trames ?

F.4 Client-Serveur

La communication des résultats des mesures entre le rack de communication et le PC de contrôle utilise le réseau Ethernet 10baseT au travers du protocole IP .

On décide d'établir une communication en mode **non connecté**.

Question F.4.1

En vous référant à l'annexe D7 , précisez les valeurs des 3 paramètres d'ouverture de la socket de communication. Ces paramètres adoptent-ils les mêmes valeurs sur le PC de contrôle et sur le rack de communication ? Expliquez.

La socket de communication étant ouverte avec succès tant au niveau du PC de contrôle que du rack de communication, l'application attache maintenant cette socket à un point de communication. Cet attachement (binding) nécessite la fourniture d'une structure de type **sockaddr_in**.

Extrait de l'annexe D7:

```
struct sockaddr_in
{
    sa_family_t    sin_family; /* address family: AF_INET */
    u_int16_t      sin_port;   /* port in network byte order */
    struct in_addr sin_addr;   /* internet address */
};

/* Internet address. */
struct in_addr
{
    u_int32_t      s_addr;     /* IPv4 address in network byte order */
};
```

Dans notre application, c'est le PC de contrôle qui sera le premier en attente de réception de datagrammes sur le port **1057**. Les adresses IP des différentes machines sont récupérées dans les trames du paragraphe F.3...

Les questions qui suivent permettent de définir les valeurs des différents membres de la structure de type **sockaddr_in** à fournir à l'appel **bind()**.

Question F.4.2

Le type **u_int16_t** correspond à un entier non signé sur 16 bits.

Quelle valeur doit être fournie pour le champ **sin_port** de la structure sur le PC de contrôle?

Question F.4.3

Quelle valeur doit être fournie pour le champ **sin_port** de la structure sur le rack de communication?

Question F.4.4

Comment le PC de contrôle connaît-il le port utilisé par le rack de communication ?

Question F.4.5

Comment le rack de communication connaît-il le port utilisé par le PC de contrôle?

G. Annexes du sujet

- Annexe S1** *Carte CPU BAB40 (extraits)*
- Annexe S2** *Carte d'extension VMOD-IO (extraits)*
- Annexe S3** *Module VMOD TTL/O*
- Annexe S4** *Module VMOD-GPIB(extraits)*
- Annexe S5** *Coupleur parallèle CIO Z8536*
- Annexe S6** *Schéma structurel du module VMOD-RL*
- Annexe S7** *Circuit ULN2003*
- Annexe S8** *Driver GPIB*

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve Etude d'un Système Informatisé

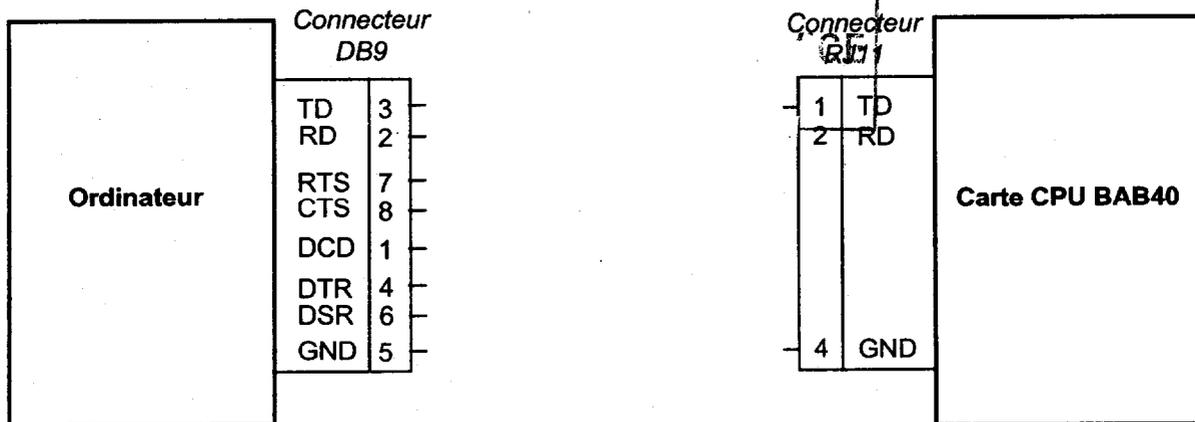
Système d'acquisition de température de bobinages

Sujet – deuxième partie : Document Réponse

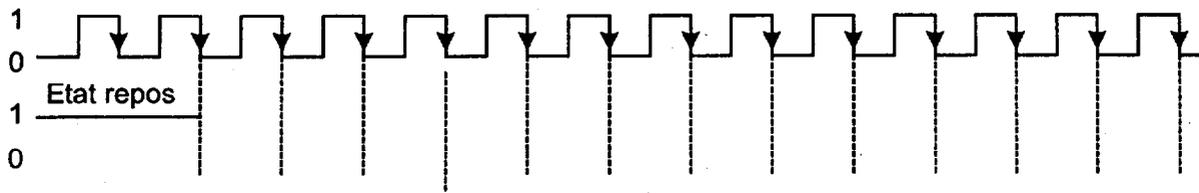
Réponse D.2.1

	Rack de commutation	Multimètre 1	Multimètre 2	Multimètre 3	Multimètre 4
Fonction	Ecouteur ou parleur				
Adresse	7				
Code Ecouteur	\$27				
Code Parleur	\$47				

Réponse F.1.2



Réponse F.1.3



Réponse F.2.1

Entité/protocole	Câble UTP	Routeur ADSL	Ethernet	Connecteur RJ45	TCP	IP	FTP
Couche(s)	1						

Réponse F.2.2

	Routeur	Serveur	PC de contrôle	Rack VME
Adresse IP				
Masque				

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve Etude d'un Système Informatisé

Système d'acquisition de température de bobinages

ANNEXES DU SUJET

Ces annexes sont communes à la première et à la deuxième partie du sujet.

Annexe S1	Carte CPU BAB40 (extraits)	3 pages
Annexe S2	Carte d'extension VMOD-IO (extraits)	2 pages
Annexe S3	Documentation VMOD TTL/O	3 pages
Annexe S4	Carte d'extension VMOD-GPIB (extraits)	2 pages
Annexe S5	Documentation du CIO Z8536	5 pages
Annexe S6	Schéma VMOD-RL	1 page
Annexe S7	Documentation du ULN2003	1 page
Annexe S8	Documentation du Driver GPIB	4 pages

ANNEXE S1

Carte CPU BAB40 (extraits)

BAB-40/60

1 Specification

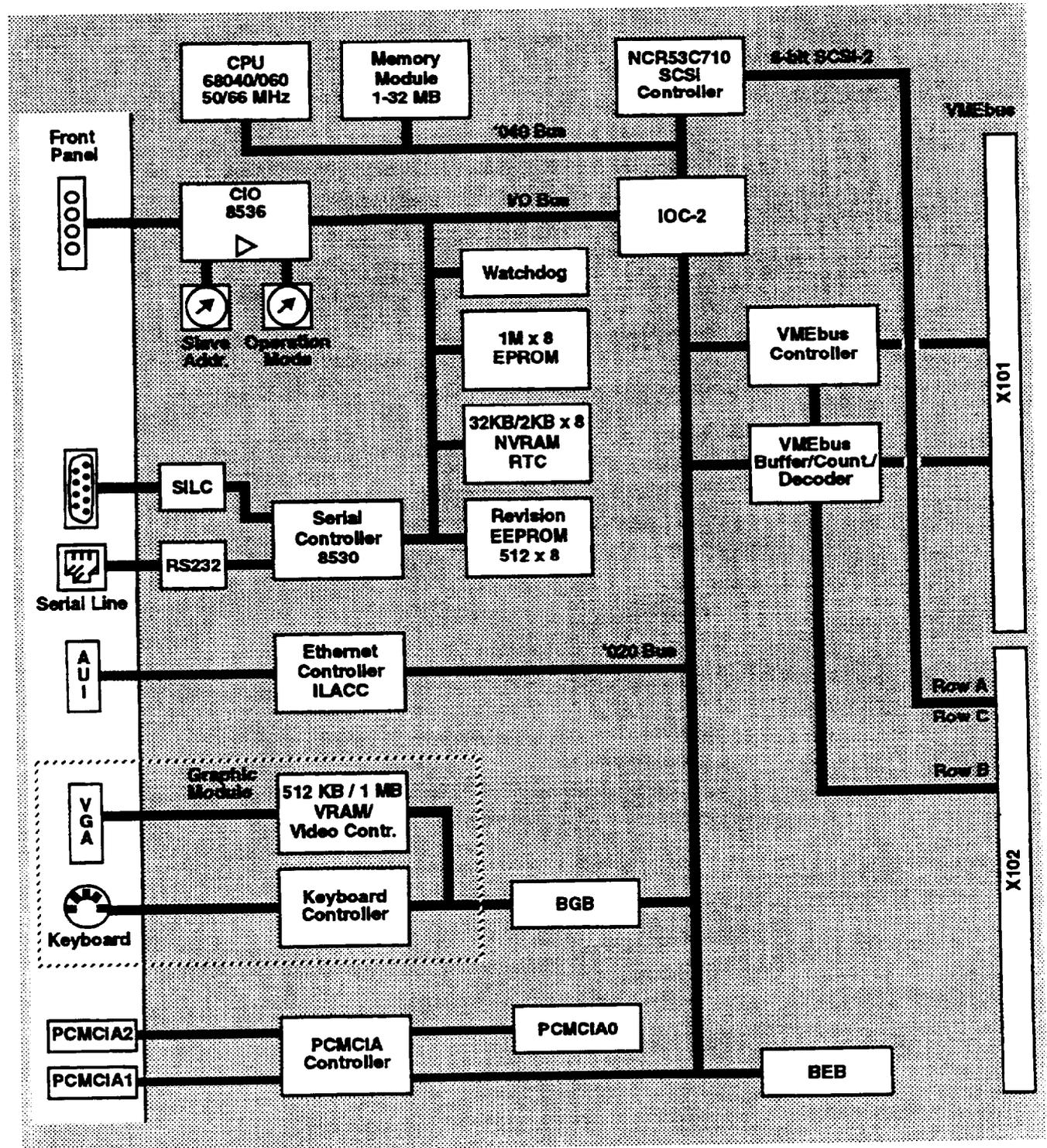
1 Specification

1.1 Main Features

- One 68(EC)040/060 CPU at 50 MHz or 66 MHz
- Memory
 - PS/2 SIMM memory module (1, 2, 8, 16, 32 MB) for data/program storage (44 MB/s at 33 MHz bus speed)
 - 2 KB SRAM and RTC for storage of variable system parameters MK48T12 (MK48T18/8 KB, DS1644/32 KB)
 - Up to 1 MB x 8 K EPROM
- Two PCMCIA sockets on front panel for two type I or II PC Cards or one type III PC Card (Flash, SRAM and ATA-HD only)
- One internal PCMCIA socket (type I, type II, or with some restrictions type III (Flash, SRAM and ATA-HD only))
- Ethernet interface (32-bit ILACC)
- VMEbus Interface Controller:
 - System controller and arbiter
 - VMEbus interrupter and interrupt handler
 - 32-bit slave BLT 20 MB/s
 - Master / slave write posting
- IOC-2 gate array:
 - 68040 to 68020 bus converter
 - Dynamic bus sizing for VMEbus and BEB
 - Translation of BLT into bursts on '040 bus to allow snooping of BLT cycles
 - Separate arbitration on '040 and '020 bus
 - I/O bus interface
 - Support for VMEbus UATs to allow snooping
 - Interface for a single byte-wide EPROM
- Three 16-bit timer / counter
- Two serial ports (RS 232, RS 422, RS 485)
- Smart SCSI-2 (NCR 53C710) interface with burst capability (max. transfer capacity 10 MB/s) and single-ended 8-bit SCSI data bus
- Two rotary switches for selection of operation modes and base address
- Status display on front panel
- Watchdog timer 130 ms ... 17 min
- BEB for interfacing to various mezzanine busses
- BGB for graphic and keyboard module

1.2 General Description

Figure 1: Block Diagram



3 Programmers Reference

3.1 Address Map

The BAB-40/60 is designed to utilize the entire 4 GB address range of the 68040/060 chip. Using the address modifier of the VMEbus, the address range may be enlarged by subdivision into data and program areas and / or user and supervisor areas. The BAB-40/60 recognizes two address areas: the local address space and the global VMEbus address space.

Table 23: Address Assignment of BAB-40/60

Address Range	Device	VMEbus Address Modifier	Cache ¹⁾	Burst ²⁾	Access Width [b]
\$0000.0000 - \$01FF.FFFF	Local RAM	local	Y	Y	32
\$0200.0000 - \$03FF.FFFF	Local RAM (mirrored)	local	N	Y	32
\$0400.0000 - \$05FF.FFFF	Video RAM	local	Y	N	32
\$0600.0000 - \$07FF.FFFF	Video RAM (mirrored)	local	N	Y	32
\$0800.0000 - \$FBFF.FFFF	VMEbus Extended	A32	N ³⁾	N	32/16/8
\$FC00.0000 - \$FC3F.FFFF	PCMCIA1	local	N	N	16/8
\$FC40.0000 - \$FC7F.FFFF	PCMCIA2	local	N	N	16/8
\$FC80.0000 - \$FCBF.FFFF	PCMCIA0	local	N	N	16/8
\$FCC0.0000 - \$FCFF.FFFF	Reserved	-	-	-	-
\$FD00.0000 - \$FDFF.FFFF	BEB0	local	N	N	32/16/8
\$FE00.0000 - \$FE7F.FFFF	BEB1	local	N	N	32/16/8
\$FE80.0000 - \$FEBF.FFFF	EPROM	local	Y	N	8
\$FEC0.0000 - \$FECF.FFFF	Local I/O	local	N	N	32/16/8
\$FED0.0000 - \$FEFF.FFFF	Reserved	-	-	-	-
\$FF00.0000 - \$FFFE.FFFF	VMEbus Standard I/O	A24	N	N	32/16/8
\$FFFF.0000 - \$FFFF.FFFF	VMEbus Short I/O	A16	N	N	16/8

1. Y = /TCI driven high, N = /TCI driven low.
2. Y = /TBI driven high, N = /TBI driven low.
3. Caching may be enabled via system control register.

ANNEXE S2

Carte d'extension VMOD-IO (extraits)

1. General

The VMOD-IO is a VMEbus-compatible carrier board for up to four plug-in modules with MODULbus-Interface. Even if the mother board is equipped with four modules, only one slot in the VME-system is Needed.

With use of the VMOD-IO the system integrator is able to build up VME-systems with flexible configurations for needs in industrial environment.

1.1 Specifications

- VMEbus nonintelligent carrier board for MODULbus
- double euro-card with A24/D16 VMEbus slave interface
- 4 plug-in sockets for MODULbus I/O
- different vector from each MODULbus socket
- jumper selectable interrupt level
- 2KByte short-I/O or standard-address range
- needs only one VME-Slot
- front panel and P2 connection of I/O lines
- optional on-board DC/DC converter to supply analog modules

1.2 Functional overview

The VMOD-IO carrier board can be equipped with a maximum of 4 MODULbus I/O-modules.

Every MODULbus socket has a 512byte address space and can be selected as byte-device or as word-device.

For integration in complex VMEbus-systems the VMOD-IO can be used within the Short-I/O address space or within the Standard memory address space.

The plug-in modules are able to cause a modified vectored interrupt to the VMEbus via the carrier board with selectable level and configurable vector. The vector holds information about the causing module in its 4 LSB's.

2. Addressing

The VMOD-IO uses 2048 bytes within the 64Kbyte wide Short-I/O or in the 16Mbyte wide Standard address range of the VMEbus.

2.1 Board-address

The basic address of the board can be set by using 2 code-switches in the SHORT address range and with 4 code-switches in the STANDARD address range.

Switch S1 corresponds to the address line A11, switch S2 to address lines A12 - A15, switch S3 to address lines A16 - A19 and switch S4 accordingly to address lines A20 - A23.

The selection of the address range can be ensured by jumper J4. If the SHORT address range is chosen, the address on the code-switches S3, S4 and on the corresponding address lines is irrelevant.

2.1.1 Example of setting

Example 1:

To select address 68xxH in the SHORT address range the code-switch S2 has to be set to "6" and code-switch S1 has to be set to "8".

Example 2 :

To select address 2568xxH in the STANDARD address range switch S4 has to be set to "2", switch S3 to "5", switch S2 to "6" and switch S1 to "8".

2.2 Address-modifiers

The VMOD-IO decodes the address modifiers which are specified by the VMEbus. This means that during an access on the AM-lines there must be a code which describes what sort of an access it is, i.e. USERDATA, SYSTEM I/O etc.

The VMOD-IO accepts User- and system- Short accesses with AM-codes 29H and 2DH and User/System Standard accesses with AM-codes 39H and 3DH. The choice is made by jumper J8, J9 and J10. Jumper J8 corresponds with AM3, J9 with AM4 and J10 with AM5. AM0 is set "high" and AM1 is set "low".

For Short-accesses the jumpers have to be set as follows:

J4 set to position 2-3

J10	J9	J8
1	0	1

0 = jumper set

For Standard-accesses the jumpers have to be set as follows :

J4 set to position 1-2

J10	J9	J8
1	1	1

0 = jumper set

2.3 Address-ranges

The addresses of the MODULbus-sockets are :

MODULbus socket	address
socket 0	0H - 1FFH
socket 1	200H - 3FFH
socket 2	400H - 5FFH
socket 3	600H - 7FFH

MODULbus addresses

ANNEXE S3

DOCUMENTATION VMOD TTL/O

VMOD TTL/O
MODULbus digital
Input / Output Module

JANZ Computer AG

General

The VMOD-TTL/O is an opto-isolated digital input/output module for MODULbus carrier-boards and is ideally suited for industrial control purpose where parallel inputs are required.

Only proven and reliable components are used in order to ensure perfect operation in industrial environments. The high standard in quality and manufacturing ensure a highly reliable product.

Specifications

- 20 bit TTL-level, opto-coupled inputs/outputs
- Single 4bit and double 8bit I/O channels
- In / output configurable per channel
- Max. 48mA open collector outputs
- 3x16bit counter/timer internally or externally controlled
- Interrupt on change of input state or input pattern
- Physically dimensions 55mm x 110mm

Functional Overview

The VMOD-TTL/O is a single module with MODULbus interface able to use on carrier boards for specific host computer bus system (VMEbus, PC/AT...)

The module is fitted with the Counter/Timer-I/O device (CIO) Z8536 from ZILOG. Thereby the module provides 20 parallel in- or outputs. These are divided in two channels of 8bit each and one channel of 4bit. Each channel is opto-isolated from system potential and is optionally configurable as input or output.

All of the inputs/outputs are connected to a 25pin D-Sub connector on the front of the module. Further on the inputs/outputs of channel B and channel C are led to a 20-pin I/O connector on the module for back panel connection of the carrier-board.

On the VMOD-TTL/O a pattern recognition logic is able to generate interrupts on a bit pattern or a rising or falling edge at the inputs of channel A or B. The 8 inputs of each channel can be logically combined by "AND" or "OR" functions. Further on interrupts can be generated by "zero count" of the counter/timer.

Addressing

The VMOD-TTL/O transmits or receives data on the low-byte data bus of the MODULbus carrier-board. It can be plugged on every MODULbus socket.

The module address can be determined as follows:
Board address + MODULbus socket + Module register

The module registers have the following addresses:

Relative Address	Function	Status
0	Port C's data register CIO	rd/wr
1	Port B's data register CIO	rd/wr
2	Port A's data register CIO	rd/wr
3	Control register CIO	rd/wr

Input- Output Configuration

All of the 20 inputs/outputs of the VMOD-TTL/O have TTL-level. They are realized on the hardware side via three drivers and opto-couplers, so that in- or outputs are available in two 8bit channels and in

one 4bit channel. Each channel is configurable as input or output by the user.

A corresponding commitment for the input/output of data at a later date has been supported by software in the right way.

The inputs/outputs are fitted with drivers of the type 74LS645-1 and opto-couplers of the type TLP-521. the open collector outputs are fitted with 4K7 pull-up resistors on the module, so that the user must only connect the external supply Vext for the outputs (Vext = 5V) for TTL-level, Vext must not exceed 35V).

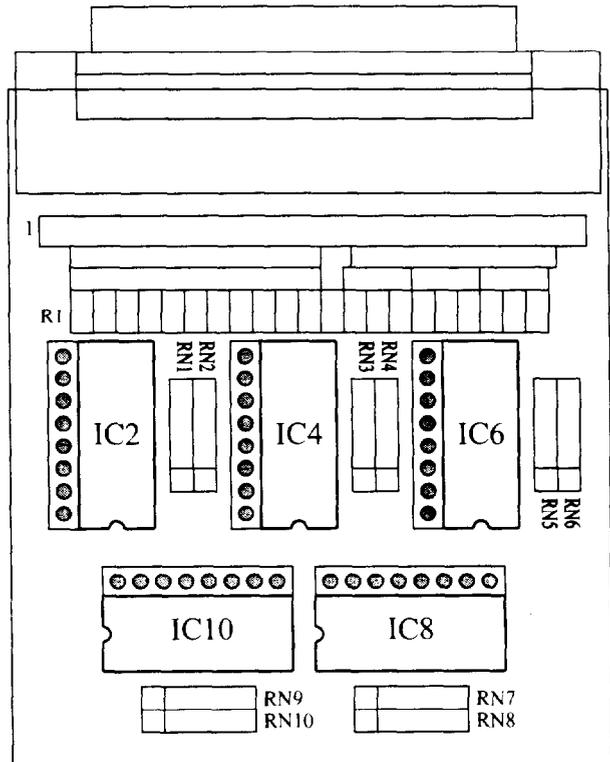
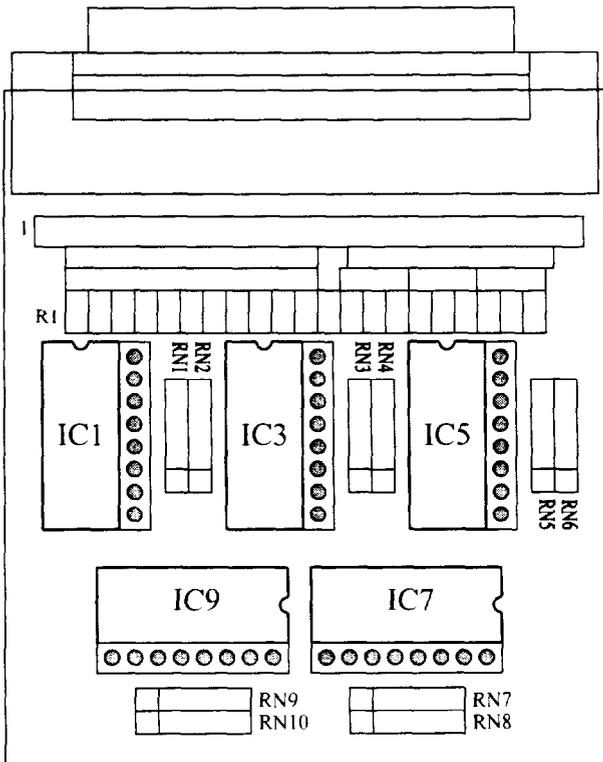
All inputs are related to a common external return potential GNDext. The input current is typically

8mA. In input mode the external supply Vext is not used.

The direction of the drivers i.e whether in- or output is decided by the jumpers J1 (port A), J2 (port B) and J3 (port C). The direction of the opto-couplers is decided by rotating the opto-coupler IC1 – IC4 (port A), IC5 – IC8 (port B) and IC9 – IC10 (port C).

For “receive”, which means that the I/O-channels have been programmed as input, the jumper J1 – J3 have to be open and the opto-couplers IC1, IC3, IC5, IC7 and IC9 have to be set as follows:

For “transmit”, which means that the I/O-channels have been programmed as output, the jumper J1 – J3 have to be set and the opto-couplers IC2, IC4, IC6, IC8 and IC10 have to be set as follows:



Each I/O channel is configurable as input or output independent of the other channels. The following table shows the jumper settings and opto-coupler placement for different I/O configurations:

Port A	Port B	Port C	J1	J2	J3	Opto-coupler placement
input	input	input	open	open	open	IC1, IC3, IC5, IC7, IC9
input	input	output	open	open	set	IC1, IC3, IC5, IC7, IC10
input	output	input	open	set	open	IC1, IC3, IC6, IC8, IC9
input	output	output	open	set	set	IC1, IC3, IC6, IC8, IC10
output	input	input	set	open	open	IC2, IC4, IC5, IC7, IC9
output	input	output	set	open	set	IC2, IC4, IC5, IC7, IC10
output	output	input	set	set	open	IC2, IC4, IC6, IC8, IC9
output	output	output	set	set	set	IC2, IC4, IC6, IC8, IC10

Pin-assignment

The 20 inputs/outputs are conducted to a 25 pins D-Sub connector on the module front. The pin assignment of the connector is show in the following table:

pin #	signal	pin #	signal
1	D0 (port A)	14	D3 (port C)
2	D2 (port A)	15	D1 (port C)
3	D4 (port A)	16	Vext (port B)
4	D6 (port A)	17	D7 (port B)
5	GNDext	18	D5 (port B)
6	D0 (port B)	19	D3 (port B)
7	D2 (port B)	20	D1 (port B)
8	D4 (port B)	21	Vext (port A)
9	D6 (port B)	22	D7 (port A)
10	GNDext	23	D5 (port A)
11	D0 (port C)	24	D3 (port A)
12	D2 (port C)	25	D1 (port A)
13	Vext (port C)		

More over the inputs are conducted via a 20 pins I/O connector to the back panel connector on the

MODULbus carrier-board. The pin assignment of the 20 pins I/O connector is as follows:

pin #	signal
1	Vext (port B)
2	Vext (port C)
3	D0 (port B)
4	D1 (port B)
5	D2 (port B)
6	D3 (port B)
7	D4 (port B)
8	D5 (port B)
9	D6 (port B)
10	D7 (port B)
11	D0 (port C)
12	D1 (port C)
13	D2 (port C)
14	D3 (port C)
15	GNDext
16	GNDext

Interrupt

The VMOD-TTL/O is fitted with a Z8536 (Counter Timer I/O) from Zilog. The CIO contains two independent 8bit wide bi-directional I/O ports and a 4bit wide special function I/O port (port C).

The inputs/outputs of the VMOD-TLL/O are realized by the 8bit port's A and B and by the 4bit port C of the Z8536.

All inputs of the port's A and B are able to generate interrupts to the MODULbus carrier-board via the interrupt line INT. With jumper INT the interrupts of the VMOD-TLL/O can be disabled (jumper open). The input bit's of port A and B can

be logically combined for the interrupt condition by a "AND" or "OR" function.

- The conditions for generating interrupts are:
- Input pattern
- Rising edge
- Falling edge
- Rising or falling edge
- "Zero count" of counter/timer

The interrupt vector must be configured on the MODULbus carrier-board, so that the interrupt vector register within the CIO device can not be used.

Technical Data

MODULbus connection
Digital inputs/outputs
Power supply
Power consumption
Operating temperature
Permissible humidity
Dimensions
Weight

Two 20pin connectors
Max. 20, opto-isolated
Vcc = 4.75V ... 5.25V
Typ. 0.25A
0 - 70°C
0% - 80%, non condens.
55mm x 112mm
ca. 100g

ANNEXE S4

Carte d'extension VMOD-GPIB (extraits)

1. Functional Overview

The JANZ General Purpose Interface Bus module VMOD-GPIB connects a VME- or IBM PC/XT/AT computer system to a general-purpose interface bus. The module is designed according to the industry standard ANSI/IEEE488.2.

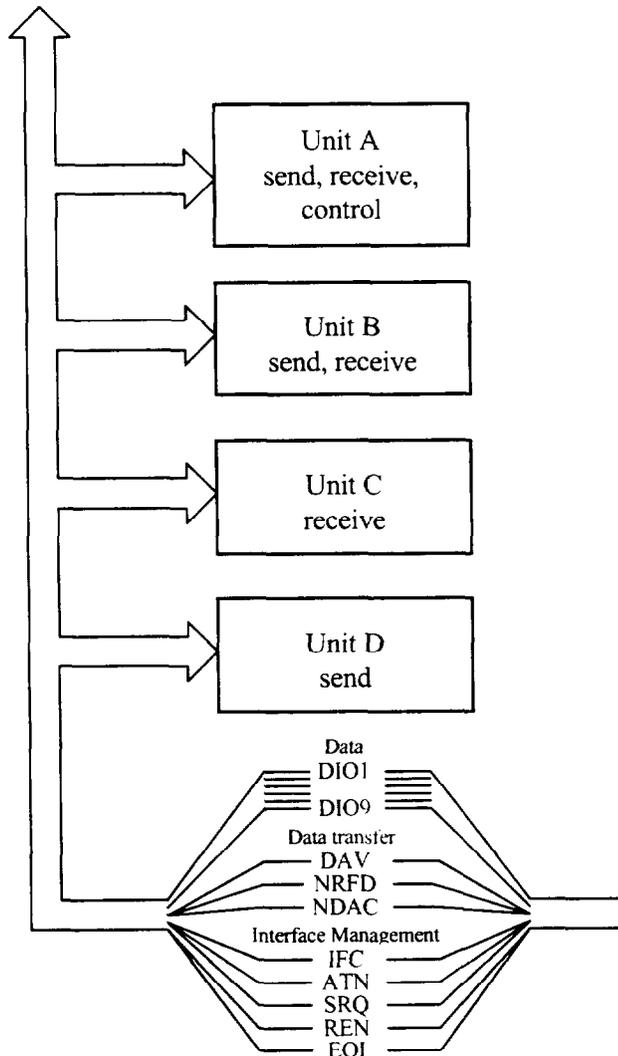


Figure 1-1: GPIB Bus Structure

The VMOD-GPIB converts every VME- or PC/AT computer system into an intelligent data acquisition or control unit. The parallel bus structure (8 data lines) guarantees high data throughput rates. The 3 wire handshake (DAV, NRFD, NDAC) eliminates any transfer errors. Another 5 wires (IFC, ATN, SRQ, REN, EOI) are for GPIB management purpose (see Figure 1-1).

Up to 15 GPIB-units are connectable on the same physical bus. A unit can be a 'Listener', a 'Talker' or a 'Controller'. The VMOD-GPIB has all 3 capabilities choosable by software.

- 1) receiving messages in the 'Listener'-mode
- 2) sending messages in the 'Talker'-mode
- 3) managing bus activities in the 'System Controller'-mode

2. Addressing

The VMOD-GPIB can be plugged in every MODULbus socket of the carrier board (MOD-AT, VMOD-IO, VMOD-IG, VMOD-40). The 16 internal registers (8 read / 8 write) of the GPIB-controller μ PD7210 are mapped in the module offset address range 00H-0EH. A write with any data to an offset address between 30H-3FH puts the VMOD-GPIB into the controller mode. In this mode, the module is able to drive the GPIB signals IFC (interface clear) and REN (remote enable). The controller mode is left by writing to an address between 20H-2FH. Each VMOD-GPIB register is byte or word addressable (see Table 2-1).

Register	Mode	Offset Address
Data In	read	00H
Data Out	write	00H
Interrupt Status 1	read	02H
Interrupt Mask 1	write	02H
Interrupt Status 2	read	04H
Interrupt Mask 2	write	04H
Serial Poll Status	read	06H
Serial Poll Mode	write	06H
Address Status	read	08H
Address Mode	write	08H
Command Pass Through	read	0AH
Auxiliary Mode	write	0AH
Address 0	read	0CH
Address 0/1	write	0CH
Address 1	read	0EH
End of String	write	0EH
...
Controller On	write	20H-2FH
Controller Off	write	30H-3FH

Table 2-1: VMOD-GPIB Register Addressing

3. Jumper Settings

The VMOD-GPIB module has only one jumper (J1), which enables the DMA feature when set. In this case, the MODULbus signals "DMRQ" (DMA Request) and "DMAK" (DMA AcKnowledge) take control over the data flow to or from the GPIB-Controller (refer to the NEC μ PD7210 description). Only the carrier board VMOD-IG supports DMA-transfer.

JUMPER J1	VMOD-GPIB Operating Mode
open	DMA disabled
closed	DMA enabled

Table 3-1: Jumper J1 Configuration

ANNEXE S5

DOCUMENTATION du CIO Z8536

Z8536 CIO
Counter/Timer and
Parallel I/O Unit

ZILOG

Features

- Two independent 8-bit, double-buffered bidirectional I/O ports plus a 4-bit special purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/WAIT signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or non-retriggerable.
- Easy to use since all registers are read/write.

General Description

The Z8536 CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers

(command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique internal address, so that any register can be accessed in two operations. All data registers can be directly accessed in a single operation. The CIO is easily interfaced to all popular microprocessors.

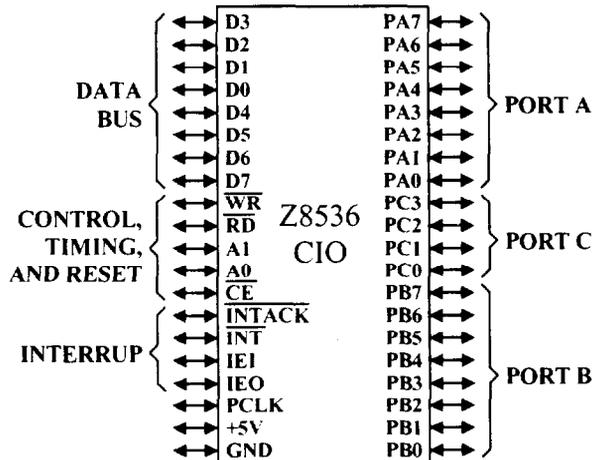


Figure 1. Pin Functions

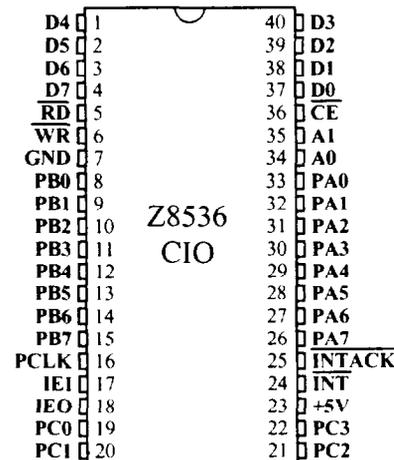


Figure 2. Pin Assignments

Pin Description

A0-A1. *Address Lines* (input). These two lines are used to select the register involved in the CPU transaction: Port A's Data register, Port B's Data register, Port C's Data register, or a control register.

CE. *Chip Enable* (input, active Low). A Low level on this input enables the CIO to be read from or written to.

D0-D7. Data Bus (Bidirectional 3-state). These eight data lines are used for transfers between the CPU and the CIO.

IEI. Interrupt Enable In (input, active high). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. Interrupt Enable Out (output, active high). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

INT. Interrupt Request (output, open-drain, active low). This signal is pulled Low when the CIO requests an interrupt.

INTACK. Interrupt Acknowledge (input, active Low). This input indicates to the CIO that an interrupt Acknowledge cycle is in progress.

INTACK must be synchronized to PCLK, and it must be stable throughout the Interrupt Acknowledge cycle.

PA0-PA7. Port A I/O lines (bidirectional, 3-state, or open-drain). These eight I/O lines transfer

information between the CIO's Port A and external device.

PB0-PB7. Port B I/O lines (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC0-PC3. Port C I/O lines (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{\text{WAIT}}$, and $\overline{\text{REQUEST}}$ lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the CIO's Port C.

PCLK. Peripheral Clock (input, TTL-compatible). This is the clock used by the internal control logic and the counter/timers in timer mode. It does not have to be the CPU clock.

$\overline{\text{RD}}^*$. Read (input, active Low). This signal indicates that a CPU is reading from the CIO. During an Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the data bus if the CIO is the highest priority device requesting an interrupt.

$\overline{\text{WR}}^*$. Write (input, active Low). This signal indicates a CPU Write to the CIO.

* When $\overline{\text{RD}}$ and $\overline{\text{WR}}$ are detected Low at the same time (normally all illegal condition), the CIO is reset.

Architecture

The CIO Counter/Timer and Parallel I/O element (figure 3) consists of a CPU interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose 4-bit port), three 16-bit counter/timers, an interrupt-control logic block, and

the internal-control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

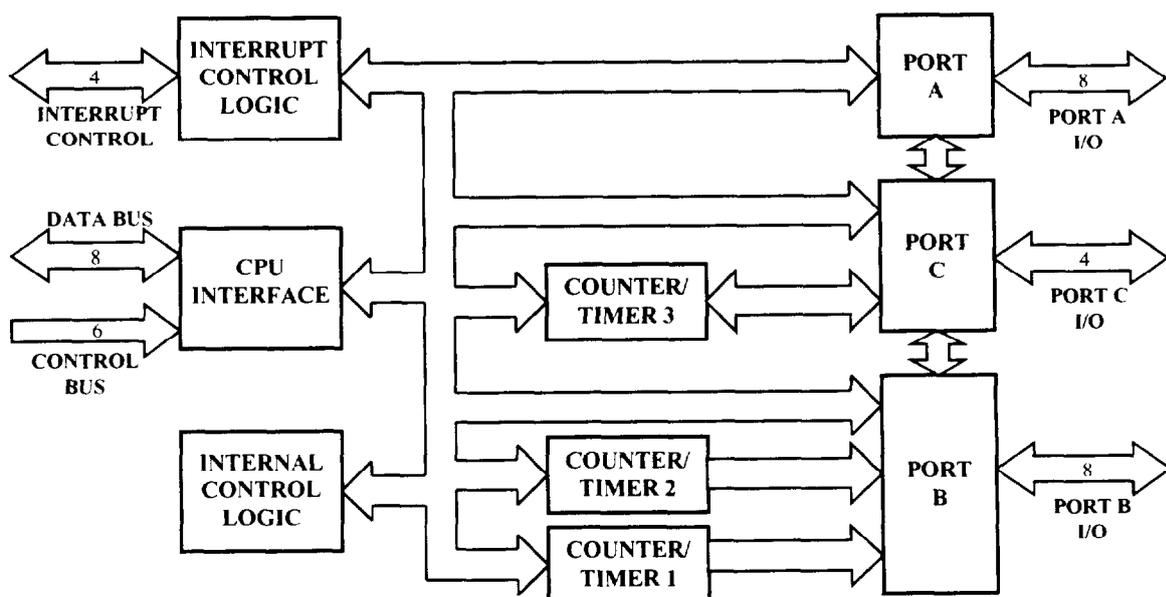


Figure 3. CIO Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Port A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity Register provides the capability of inverting the data path. A 1 specifies

inverting, and a 0 specifies non-inverting. All discussions of the port operations assume that the path is non-inverting.

The value returned when reading an input bit reflects the state of the input just prior the input data to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its outputs is set to 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Programming

The data registers within the CIO are directly accessed by address lines A₀ and A₁ (Table 3). All other internal registers are accessed by the following two-step sequence, with the address lines specifying a control operation. First, write the address of the target register to an internal 6-bit Pointer Register; then read from or write to the target register. The Data registers can also be accessed by this method.

An internal state machine determines if accesses with A₀ and A₁ equaling 1 are to the Pointer Register or to an internal control register (Figure 11). Following any control read operation, the state machine is in State 0 (the next control access is to the Pointer Register). This can be used to force the state machine into a known state. Control reads in State 0 return the contents of the last register pointed to. Therefore, a register can be read

continuously without writing to the Pointer. While the CIO is in State 1 (next control access is to the register pointed to), many internal operations are suspended- no I/Os are set and internal status is frozen. Therefore, to minimize interrupt latency and to allow continuous status updates, the CIO should not be left in State 1.

The CIO is reset by forcing \overline{RD} and \overline{WR} Low simultaneously (normally an illegal condition) or by writing a 1 to the Reset bit. Reset disables all functions except a read from or write to the Reset bit; writes to all others bits are ignored, and all reads return 01_H. In this state, all control bits are forced to 0 and may be programmed only after clearing the Reset bit (by writing a 0 to it).

A ₀	A ₁	Register
0	0	Port C's Data Register
0	1	Port B's Data Register
1	0	Port A's Data Register
1	1	Control Register

Table 3. Register Selection

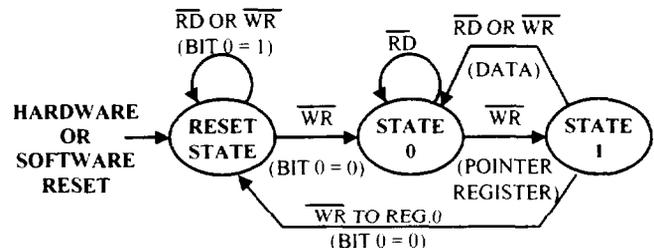
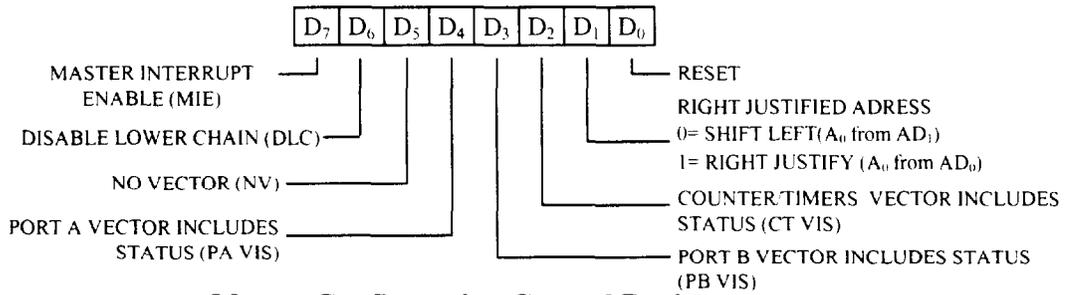


Figure 11. State Machine Operation

Registers

Master Interrupt Control Register

Addresses: 000000
(Read/Write)



Master Configuration Control Register

Addresses: 000001
(Read/Write)

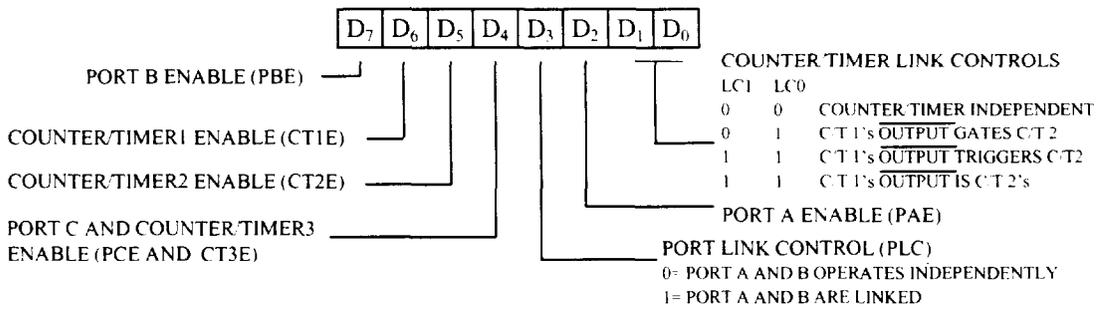


Figure 12. Master Control Register

Port Mode Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)

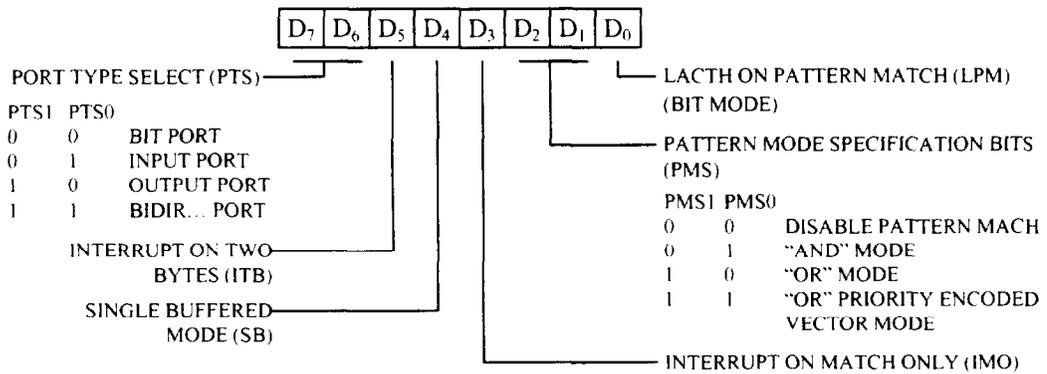
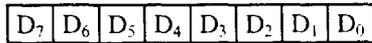


Figure 13. Port Specification Register

Data Path Polarity Registers

Addresses: 100010 Port A
 101010 Port B
 000101 Port C (4 LSBs only)
 (Read/Write)



DATA PATH POLARITY (DPP)
 0 = NON-INVERTING
 1 = INVERTING

Data Direction Registers

Addresses: 100011 Port A
 101011 Port B
 000110 Port C (4 LSBs only)
 (Read/Write)



DATA DIRECTION (DD)
 0 = OUTPUT BIT
 1 = INPUT BIT

Special I/O Control Registers

Addresses: 100100 Port A
 101100 Port B
 000111 Port C (4 LSBs only)
 (Read/Write)

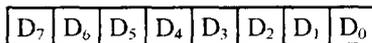


SPECIAL INPUT/OUTPUT (SIO)
 0 = NORMAL INPUT OR OUTPUT
 1 = OUTPUT WITH OPEN-DRAIN
 INPUT WITH 1's CATCHER

Figure 14. Bit Path Definition Registers

Port Data Registers

Addresses: 001101 Port A*
 001110 Port B*
 (Read/Write)



*These registers can be addressed directly.

Port C Data Register

Addresses: 001111 *
 (Read/Write)



4 MSBs
 0 = WRITING OF CORRESPONDING LSB ENABLED
 1 = WRITING OF CORRESPONDING LSB INHIBITED
 (READ RETURNS 1)

Figure 15. Port Data Registers

Main Control Register

Address	Register name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Port A specification Register

Address	Register name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Most Often Accessed Registers

Address	Register name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data (can be accessed directly)
001110	Port B's Data (can be accessed directly)
001111	Port C's Data (can be accessed directly)

Port B specification Register

Address	Register name
101000	Port A's Mode Specification
101001	Port A's Handshake Specification
101010	Port A's Data Path Polarity
101011	Port A's Data Direction
101100	Port A's Special I/O Control
101101	Port A's Pattern Polarity
101110	Port A's Pattern Transition
101111	Port A's Pattern Mask

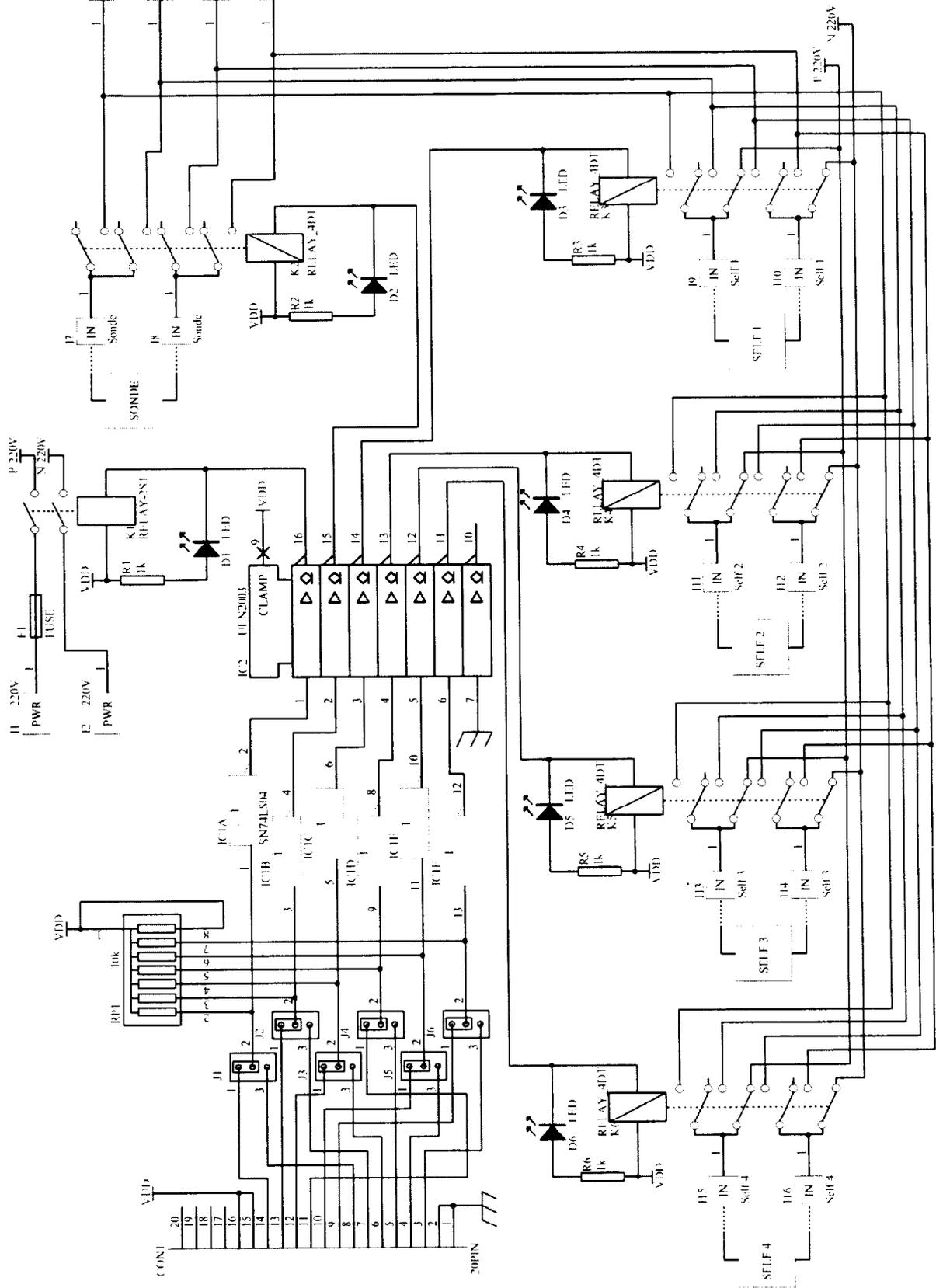
Figure 16. Register Address Summary

Title:

VMOD-RL

Rev: 1.1

Size: A4



4

3

2

1

4

3

2

1

ANNEXE S7

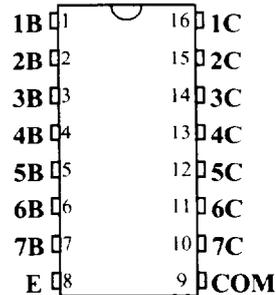
DOCUMENTATION du ULN2003

ULN2001A, ULN2002A, ULN2003A, ULN2004A DARLINGTON TRANSISTOR ARRAYS

HIGH-VOLTAGE HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS

- 500-mA Rated Collector Current (Single Output)
- High-Voltage Outputs ... 50 V
- Output Clamp Diodes
- Inputs Compatible With Various Types of Logic
- Relay Driver Applications
- Designed to Be Interchangeable With Sprague ULN2001A Series

D or N PACKAGE
(TOP VIEW)



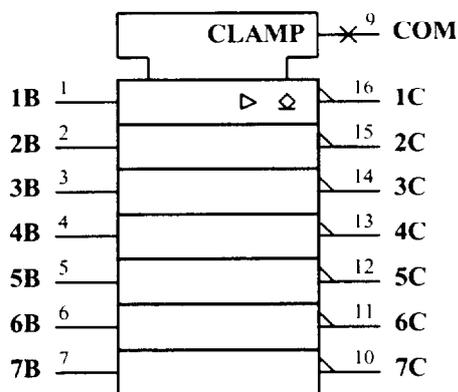
Description

The ULN2001A, ULN2002A, ULN2003A, and ULN2004A are monolithic high-voltage, high-current Darlington transistor arrays. Each consists of seven NPN Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of a single Darlington pair is 500 mA. The Darlington pairs may be paralleled for higher current capability. Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers.

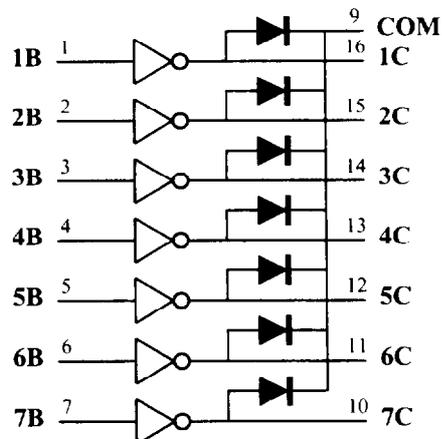
For 100-V (otherwise interchangeable) versions, see the SN75465 through SN75469.

The ULN2001A is a general-purpose array and can be used with TTL and CMOS technologies. The ULN2002A is specifically designed for use with 14- to 25-V PMOS devices. Each input of this device has a zener diode and resistor in series to control the input current to a safe limit. The ULN2003A has a 2.7-k Ω series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices. The ULN2004A has a 10.5-k Ω series base resistor to allow its operation directly from CMOS devices that use supply voltages of 6 to 15 V. The required input current of the ULN2004A is below that of the ULN2003A, and the required voltage is less than that required by the ULN2002A.

logic symbol†



logic diagram



† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.

ANNEXE S8

DOCUMENTATION DU DRIVER GPIB

mgpib(1)

TORNADO REFERENCE: VXWORKS

1

NAME mgpib - VMOD-GPIB driver

SYNOPSIS

mgpibDrv() - GPIB driver initialization routine
mgpibDevCreate() - create a device for the VMOD-GPIB

STATUS mgpibDrv(void)
STATUS mgpibDevCreate(char * name, int brdflag, int modno, UINT baddr, int vector, int level)

DESCRIPTION

This is the driver for the VMOD-GPIB IEEE-488 interface module. The driver support only master functionality.

USER CALLABLE ROUTINES

Most of the routines in this driver are accessible only through the I/O system. Two routines, however, must be called directly:

mgpibDrv() to initialize the driver, and
mgpibDevCreate() to create devices.

Before the driver can be used, it must be initialized by calling gpibDrv(). This routine should be called exactly once, before any reads, writes, or calls to mgpibDevCreate().

Before the VMOD-GPIB module can be used, it must be created by mgpibDevCreate().

MgpibDevCreate()

NAME

mgpibDevCreate() - create a device for the VMOD-GPIB

SYNOPSIS

```
STATUS mgpibDevCreate
(
    char * name,      /* name to use for this device */
    int brdflag,     /* board flag */
    int modno,       /* module number */
    UINT baddr,      /* MGPIB base address */
    int vector,      /* MGPIB vector */
    int level        /* MGPIB level */
)
```

DESCRIPTION

This routine creates a device for the GPIB module port. Each port to be used should have exactly one device associated with it by calling this routine. The maximum number of GPIB modules that is supported is <MAX_MGPIB_SLOTS> as defined in <mgpib.h>.

<name> will be the device name, this can then be used to open the device after it is created.

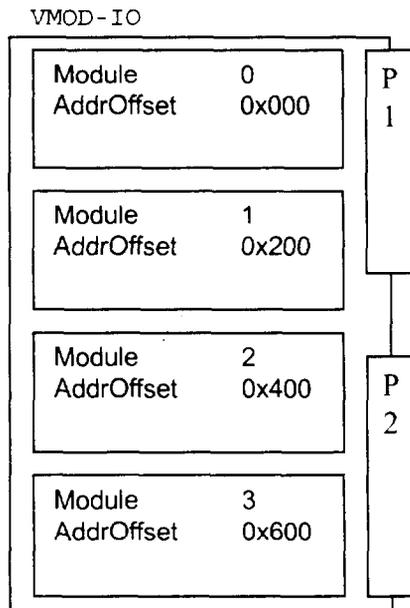
The integer number <brdflag> specifies the carrier board type, on which your module is installed. Currently the following carrier board types are supported:

- 0 VMOD-IO, the nonintelligent 6U VMEbus MODULbus carrier board.
- 1 VMOD-40/60, the 68k VMEbus processor board with MODULbus.
- 2 MOD-AT/MOD-ATS, the nonintelligent carrier boards for ISAbus.
- 3 MOD-PCI, the nonintelligent carrier board for PCibus.
- 4 LKMOD, a portable library for usage of CPU local MODULbus slots. currently supported by VMOD-P5 local slots.

Some of the board types might not be available, depending on the compile time configuration of the driver.

VMOD-IO

The VMOD-IO can take up to four modules, so the <modno> parameter will range from <0..3> (where <0> is the topmost module). For the <baddr> parameter, the access address and address space of the VMOD-IO on the VMEbus as well as the host CPU address map must be taken into consideration.



For further need to specify the base interrupt vector that you have jumpered on your carrier board. Note that <vector> will be equal for all devices on the same VMOD-IO.

By <level> you specify the hardware interrupt level that you have jumpered on the VMOD-IO. This information is used to enable VMEbus interrupts on the CPU board (by using sysIntEnable()).

The VMOD-IO has a strange interrupt vector generation: It might produce 15 different interrupt vectors if a board is equipped with 4 interrupt generating modules.

All the possible interrupt vectors need to be connected to proper interrupt service routines when the devices are created, and therefore some interrupt routines need to be connected to several interrupt vectors to cover all combinations. The driver does so automatically when a device

is created (This is the reason why you only need to specify the base interrupt vector when you call `mgpibDevCreate()`).

The assignment of interrupt vectors to interrupt routines is done as follows:

vector offset (binary)	interrupt routine
0000	module 3
...	
0111	module 3
1000	module 2
...	
1011	module 2
1100	module 1
1101	module 1
1110	module 0
1111	not assigned

With this assignment, module 3 has the highest priority (in case of simultaneously pending interrupts from several modules). Note that each module has its unique interrupt service routine, and that no negotiation between these interrupt routines is necessary. It does not matter if slots were not connected to any device (and no interrupt routine is inserted into the vector-table at a slots positions), or if all the devices are of a different type.

All VxWorks drivers 'made by JANZ' follow this assignment system. If you want to mix your drivers with ours, then you need to implement the same scheme.

VMOD-IO EXAMPLE

Suppose that you have a CPU that accesses the VMEbus short IO range at the local address `<0xffff0000>` (which is a common assumption), and that you have configured your VMOD-IO to reside on address `<0xa800>` within short IO range, then a device for a VMOD-ICAN in the second-topmost slot would be created as follows:

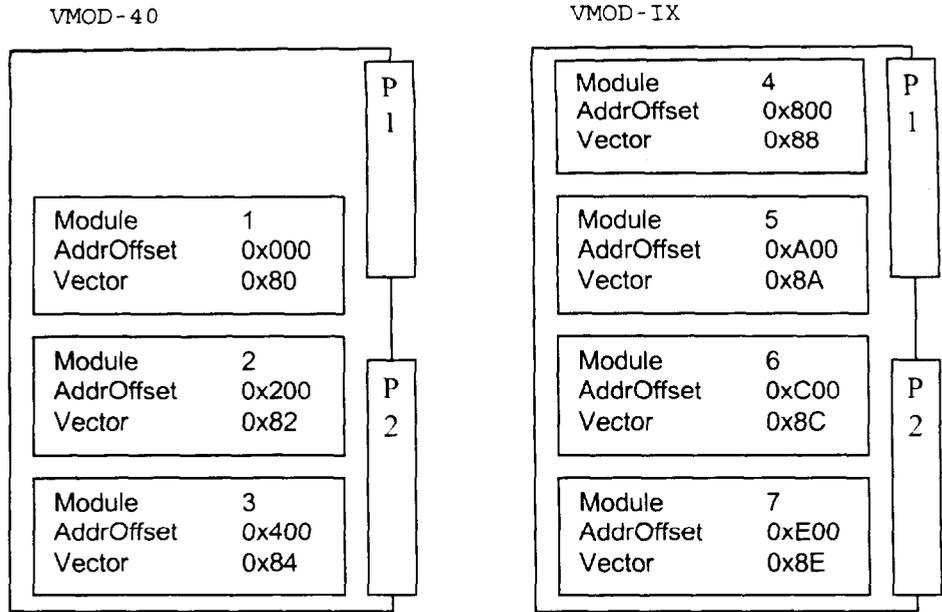
```
mgpibDevCreate("/mgpib_io_1", 0, 1, 0xffffaa00, 0xa0, 3);
```

This example further assumes that the VMOD-IO uses IRQ-level 3 and the interrupt vector is configured to be `<0xax>`.

VMOD-40

The VMOD-IO can take up to seven modules, the `<modno>` parameter will range from `<1..7>`. Module `<1>` is the topmost module on the base board, and `<4>` is the topmost module on the ix extension board. The `<baddr>` parameter will always be `<0xfe400000 + AddrOffset>`. This would be `<0xfe400400>` for the third module.

The MODULbus sockets on the VMOD-40 are always assigned to interrupt level 5 (you still you must specify this). When you have not modified the BSP, interrupt vector numbers starting at `0x80` are used for MODULbus interrupts.



VMOD-40 EXAMPLE

To create a device that operates on the topmost MODULbus slot of a VMOD-40, you need to say:

```
mgpibDevCreate("/mgpib_1", 1, 1, 0xfe400000, 0x80, 5);
```

mgpibDrv()

NAME

mgpibDrv() - GPIB driver initialization routine

SYNOPSIS

STATUS mgpibDrv (void)

DESCRIPTION

This routine initializes the driver, and performs hardware initialization of the GPIB module.

This routine should be called exactly once, before any reads, writes, or calls to mgpibDevCreate(). Normally, it is called by usrRoot() in usrConfig.c.

RETURNS

OK, or ERROR if the driver cannot be installed.

SEE ALSO

mgpib, mgpibDevCreate()

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve
Etude d'un Système Informatisé

Dossier technique

Système d'acquisition de température de bobinages

AVERTISSEMENT

Le candidat devra se munir de ce dossier technique le jour de l'épreuve 'Etude d'un Système Informatisé'

Ce dossier technique comprend deux parties :

1. Une présentation générale de 14 pages
2. Une documentation technique composée de 7 annexes numérotées D1 à D7

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve
Etude d'un Système Informatisé

Dossier technique

Système d'acquisition de température de bobinages

PRESENTATION DU SYSTEME

Important

Pour l'examen, le candidat devra être en possession de ce dossier et de ses supports de cours sur :

- L'architecture des systèmes programmés (décodage d'adresse, programmation d'un coupleur, interruptions) ;
- Le langage UML (diagramme de collaboration, séquences et classes) ;
- Le langage C++ ;
- Les liaisons parallèles et séries (bus IEEE 488, RS232) ;
- Les protocoles et la programmation réseaux (Ethernet, IP, TCP/UDP, sockets) ;

Plan

1. Présentation du système

- 1.1 *Expression du besoin*
- 1.2 *Synoptique général*
- 1.3 *Principe de la mesure*
- 1.4 *Spécifications*
- 1.5 *Architecture matérielle*
- 1.6 *Analyse UML*
- 1.7 *Ressources documentaires*

2. Annexes techniques

- Annexe D1 Norme de sécurité EN 60 335-1 (extraits)*
- Annexe D2 Exemple de fiche de mesure*
- Annexe D3 Récapitulatif partiel des notations UML 1.0*
- Annexe D4 Le Bus d'instrumentation IEEE 488 (GPIB)*
- Annexe D5 Documentation des fonctions GPIB*
- Annexe D6 Multimètre HP 34401A- Guide de l'interface IEEE 488 (Extraits)*
- Annexe D7 Programmation Réseau- Syntaxe de quelques appels systèmes*

Nota : Le sujet s'intéressera essentiellement à la partie embarquée du système présenté.

1 Présentation du système

1.1 Expression du besoin

Pour être commercialisé, un four à micro-ondes doit respecter des normes de sécurité. Ces normes sont fixées par des organismes officiels et sont décrites dans des documents de référence. Ces standards de sécurité assurent un niveau de qualité minimum du produit. Un produit ne respectant pas ces limites peut s'avérer dangereux pour le consommateur.

Une des normes s'appliquant aux fours à micro-ondes est la EN 60 335-1 (voir Annexe D1). Dans cette norme, le paragraphe 11.8 (échauffement) définit des cycles « normaux » de fonctionnement permettant d'échauffer les composants internes du four. A la fin des cycles, la température maximum de chaque composant est mesurée. Elle devra être inférieure à une limite prédéfinie pour que le four soit approuvé.

Un four à micro-ondes est constitué, entre autres, d'éléments bobinés : transformateur H.T. pour l'alimentation du magnétron, ventilateur, moteur du plateau tournant, moteur de commandes de soupape, transformateur d'alimentation de la carte électronique, moteur de minuterie...

Chaque bobinage a une classe donnant une valeur maximale d'échauffement. Exemple : un bobinage de classe H doit avoir un échauffement normal inférieur à 140 °C.

Le système d'acquisition de températures de bobinage doit permettre d'acquérir et d'afficher la température de tous les bobinages d'un four à micro-ondes à la fin d'un cycle de chauffe normalisé en respectant la norme EN 60 335-1.

Ce système est en exploitation dans plusieurs sociétés de fabrication de fours à micro-onde. Elles disposent de laboratoires d'essais et de mesures permettant la mise au point et la certification des fours.

Un exemple de fiche de mesure d'échauffement est donné en Annexe D2.

1.2 Synoptique général

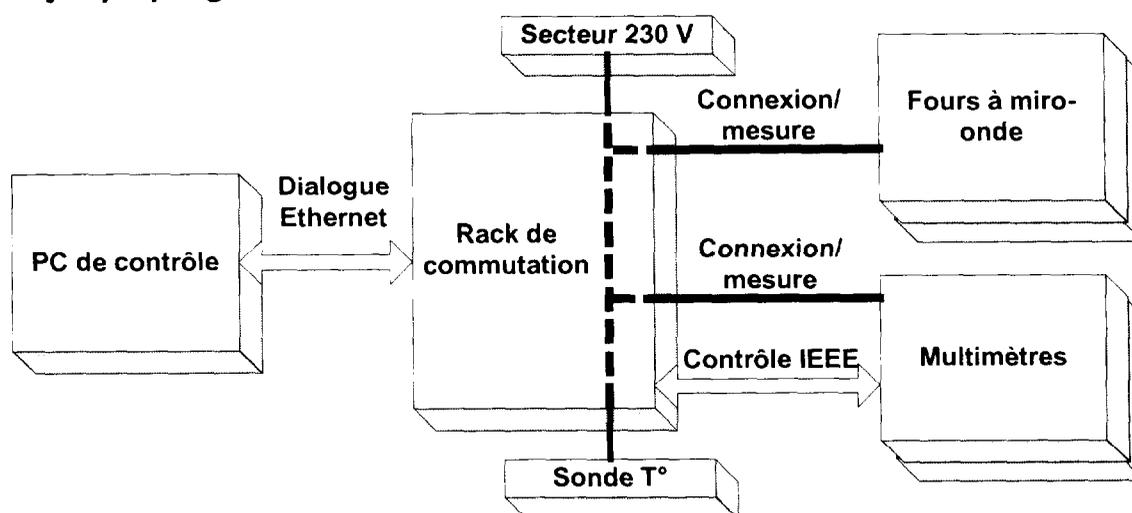


Figure 1 Synoptique général

Pour chaque bobinage de four, des mesures de résistance réalisées par des multimètres permettent de déterminer l'échauffement (voir principe de la mesure au §1.3 page 3).

Le rack de commutation permet de relier de 1 à 4 multimètres à un maximum de 4 fours à micro-onde. Il gère également la commande des multimètres via un bus d'instrumentation IEEE 488.

Son rôle est donc :

- de commuter le mode alimentation normale des bobinages (230 V) ;
- de connecter indépendamment chaque bobinage au multimètre choisi ;
- de couper l'alimentation des bobinages pour les mesures à chaud ;
- de connecter une sonde au multimètre pour mesurer la température ambiante ;
- de configurer le(s) multimètre(s) par le bus d'instrumentation IEEE 488 ;
- de commander et lire une mesure par le bus IEEE 488.

La connexion entre le rack de commutation et le four à micro-ondes permet soit d'alimenter normalement le bobinage, soit de mesurer sa température.

Le PC contrôle par une liaison Ethernet le rack de commutation et permet au technicien :

- de sélectionner le(s) four(s) et le(s) multimètres intervenants dans la mesure ;
- de choisir le mode de fonctionnement (alimentation normale du four ou mesure) ;
- de sélectionner les bobines et la sonde de température ;
- de contrôler et d'interpréter les mesures (départ, erreur, visualisation, sauvegarde...).

Une exploitation locale en réseau est prévue à l'aide d'un serveur chargé de stocker les résultats des mesures de différentes gammes de four. L'exploitation distante est réalisée par l'intermédiaire d'un routeur connecté au réseau Internet (voir architecture matérielle au §1.5).

1.3 Principe de la mesure

1.3.1 Calcul de l'échauffement

Le principe de la mesure est imposé par la norme EN 60 335-1. On doit effectuer une mesure de la résistance du bobinage lorsqu'il est à température ambiante et une mesure de résistance lorsqu'il est chaud. En connaissant sa matière (cuivre ou aluminium), ainsi que la température ambiante, on calcule alors son échauffement grâce à la formule suivante (spécifiée dans la norme EN 60 335-1).

$$\Delta T = \frac{(R_f - R_i) * (k + T_i)}{R_i} - (T_f - T_i)$$

ΔT : échauffement du bobinage

R_f : résistance du bobinage à chaud (finale)

R_i : résistance du bobinage à froid (initiale)

T_i : température ambiante à froid

T_f : température ambiante à chaud

k : coefficient (234.5 pour le cuivre ; 225 pour l'aluminium)

1.3.2 Mesure à froid

On doit mesurer la résistance à froid (R_i) de chaque bobinage et la température ambiante (T_i).

La résistance du bobinage étant constante, il n'y a pas de contrainte de temps particulière. Il suffit de connecter un à un les bobinages sur le multimètre et d'effectuer une série de mesures.

On pourra prendre 10 mesures, vérifier que l'écart entre la plus petite et la plus grande est inférieur à 0.1% puis effectuer la moyenne.

1.3.3 Mesure à chaud

Pour les mesures à chaud, le problème est plus délicat.

On ne peut pas prendre les mesures de résistance de bobinage tant que celui-ci est alimenté sous 230V.

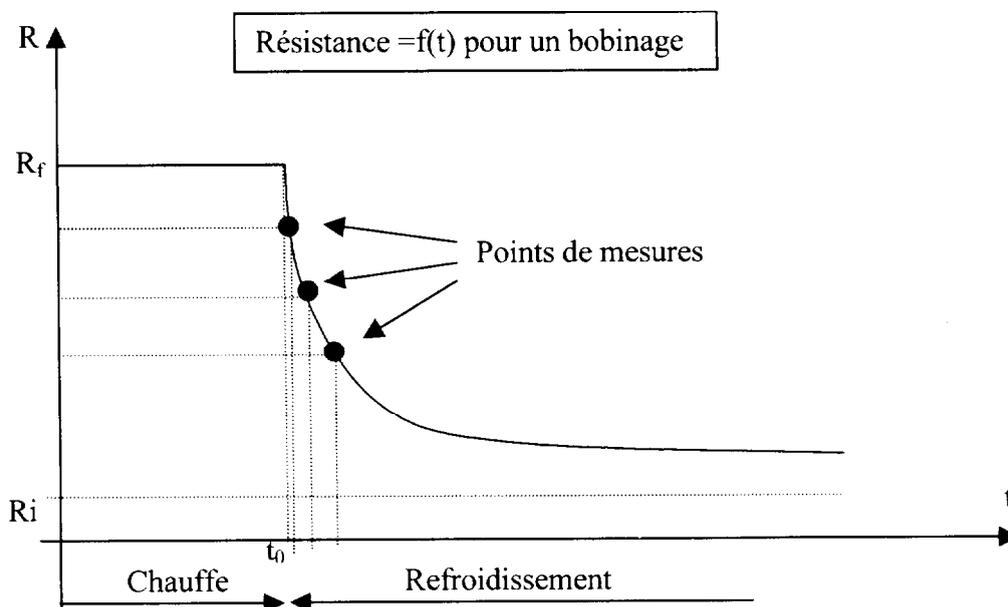
On doit en premier lieu couper l'alimentation du four à micro-ondes puis connecter au multimètre, un à un, les bobinages à mesurer. On doit attendre ensuite que le courant de mesure se stabilise (effet selfique) et enfin effectuer une série de mesures.

Conséquence : on n'est pas capable de mesurer directement la résistance à chaud au moment de l'arrêt pour tous les bobinages.

Solution : on effectue plusieurs points de mesures pendant le refroidissement pour chaque bobinage. Ensuite on détermine les équations $R=f(t)$ pendant le refroidissement et enfin on calcule $R_f = f(t_0)$.

Tous les bobinages doivent être mesurés en moins de 2 minutes.

Exemple de courbes



t_0 : début de refroidissement.

Un point de mesures peut être constitué d'un nuage de points (5 points).

R_f : valeur de résistance à déterminer

L'équation de courbe de refroidissement est du type $R=A \exp^{(-t/\tau)} + R_i$.

A : constante à déterminer. (A est compris entre 1 et 100 k Ω)

τ : constante à déterminer (ordre de grandeur : on peut considérer que le bobinage est refroidi à 95% après 3h donc $3\tau = 3h$ et $\tau = 3600s$).

Figure 2 Courbe de Refroidissement

1.3.4 Connexion des éléments

La mesure de la résistance du bobinage est effectuée selon la méthode 4 fils.

Envoi d'un courant constant par deux fils dans le bobinage, mesure par deux autres fils de la tension alors présente. On calcule la résistance en appliquant la loi d'ohm $R=U/I$.

La sonde de température est connectée directement sur le multimètre qui possède un mode spécifique pour ce type de composant.

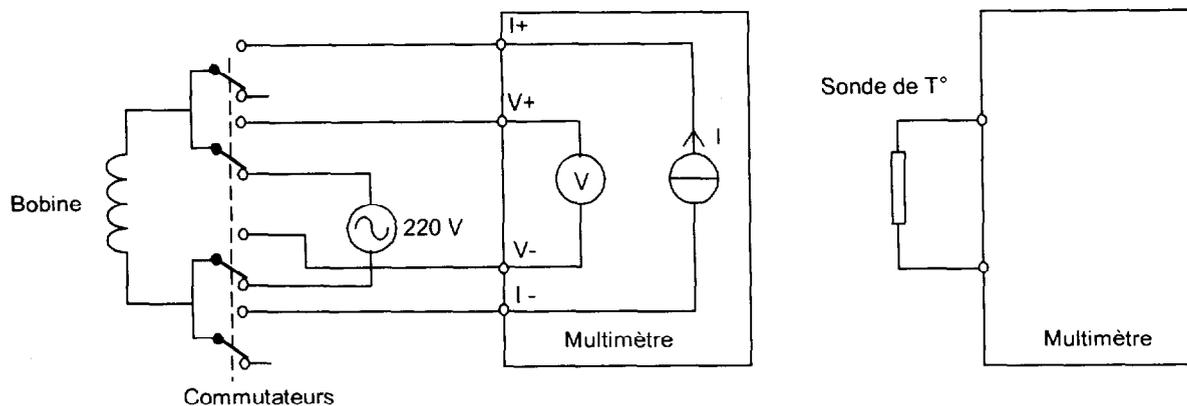


Figure 3 Connexion des éléments

Après coupure de l'alimentation des relais, la loi de variation en fonction du temps de la valeur de la résistance mesurée est, comme exposé précédemment, une exponentielle décroissante d'équation $R = A \cdot e^{-t/\tau} + R_i \dots$

On rappelle que la dérivée de $A \cdot e^{-t/\tau} + R_i$ est une fonction d'équation $-(A/\tau) e^{-t/\tau}$. La valeur de cette dérivée à l'instant t est la pente de l'exponentielle $A \cdot e^{-t/\tau} + R_i$ à cet instant.

Cette exponentielle décroissante peut être assimilée sur les 120 premières secondes à une droite d'équation :

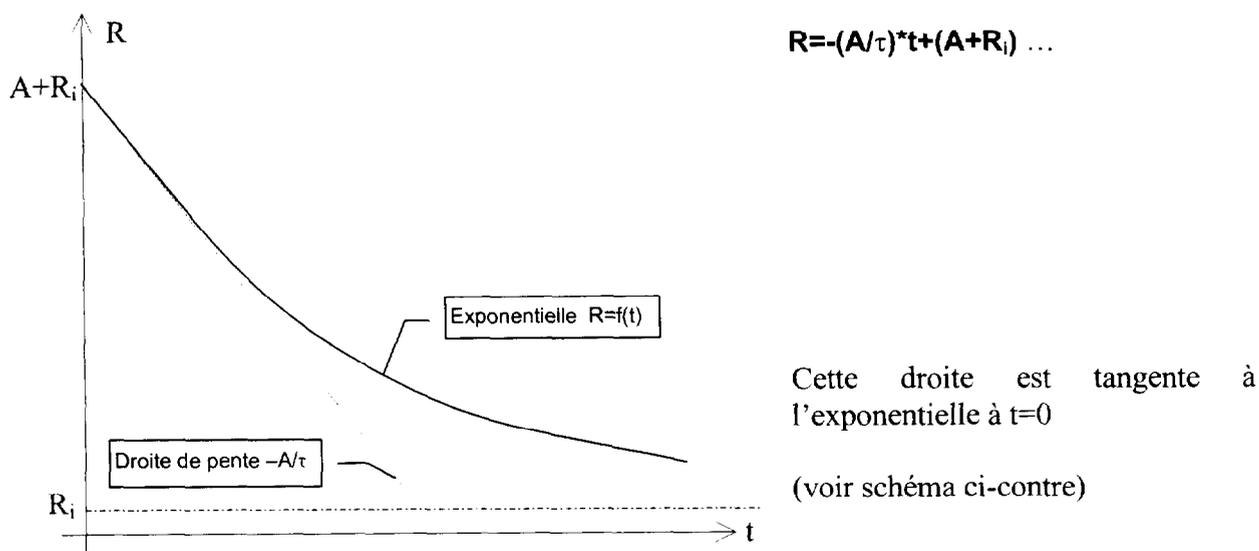
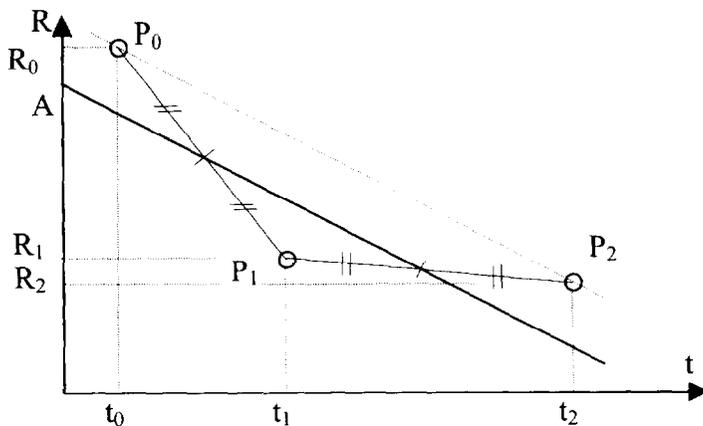


Figure 4 Droite tangente

Au cours de ces 120 secondes, chacun de ces bobinages fait l'objet de trois séries de mesures. De ces trois points, on essaie de déterminer l'équation de la droite $R=f(t)$.

Habituellement, on utilise pour trouver l'équation d'une droite à partir de points de mesure, les formules de régression linéaire. Cependant, dans le cas où le nombre de points permettant de déterminer la droite est de trois, la droite obtenue est parallèle aux deux points extrêmes et

équidistante de chacun des trois points (entre autres elle passe par le centre du segment qui relie chacun des points extrêmes au point central).



La droite obtenue a pour équation :

$$R = \frac{R_2 - R_0}{t_2 - t_0} * t + \frac{(R_1 + R_2) * (t_2 - t_0) - (t_1 + t_2) * (R_2 - R_0)}{2 * (t_2 - t_0)}$$

Figure 5 Interpolation

1.4 Spécifications

1.4.1. Fonctionnalités

□ Mesurer à froid

On veut mesurer la résistance de chaque bobinage à température ambiante.

La mesure à froid est lancée par le technicien. Il doit pouvoir suivre à l'écran le déroulement de l'acquisition.

Le nombre de bobinages connectés au système de mesure étant variable, il faut tout d'abord détecter automatiquement leur présence. Une fois que l'on sait sur quelles voies de mesures sont les enroulements, on procède à leurs mesures.

On doit aussi relever la température ambiante.

Une fois les mesures acquises, elles doivent être approuvées visuellement par le technicien.

Celui ci saisira manuellement le type de matériau de chaque bobinage (cuivre par défaut ou aluminium).

Si les mesures sont satisfaisantes, elles seront sauvegardées dans un fichier. Les informations à sauvegarder sont les valeurs et les voies connectées. On se propose d'avoir au maximum 10 fichiers de sauvegarde des mesures à froid. Le nom du fichier doit être facilement retrouvable à partir de la référence, la date et l'heure de la mesure et le nom du technicien.

□ Visualiser les mesures à froid

On veut pouvoir visualiser les dernières mesures à froid d'un four. Pour cela on sélectionne un des fichiers de mesures et on affiche les valeurs mesurées.

□ Mesurer a chaud

Avant d'effectuer la mesure à chaud, on sélectionne le fichier des mesures à froid correspondant au four pour récupérer le numéro des voies connectées.

Le lancement de la mesure est manuel ou déclenché automatiquement après un certain temps à saisir.

On doit pouvoir suivre à l'écran la prise de mesures.

On effectue 3 cycles identiques de mesures sur chaque bobinage avec 5 valeurs par cycle.

On relève la température ambiante.
La température du bobinage doit être déterminée à $\pm 1^{\circ}\text{C}$.

On sauvegarde les valeurs dans un fichier. On doit pouvoir identifier facilement les mesures à froid associées.

Un bouton doit permettre de lancer directement la visualisation des mesures et le calcul des températures.

□ Consulter les mesures à chaud

On veut pouvoir visualiser et modifier les mesures relevées à chaud.

Une modification peut être effectuée quand une mesure est erronée. Deux méthodes sont proposées : saisie manuelle d'une nouvelle valeur ou calcul linéaire à partir des 2 valeurs les plus proches.

Si une modification est effectuée, on sauvegardera les données dans le fichier.

□ Calculer et éditer

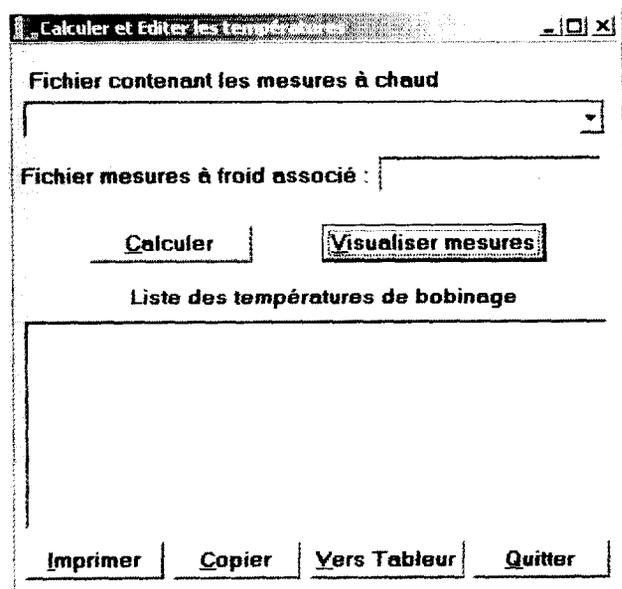
Une fois que les mesures sont effectuées, on calcule et on affiche les températures de bobinage.

Si cette fenêtre a été lancée à partir des mesures à chaud, on affiche la référence des mesures à froid et à chaud.

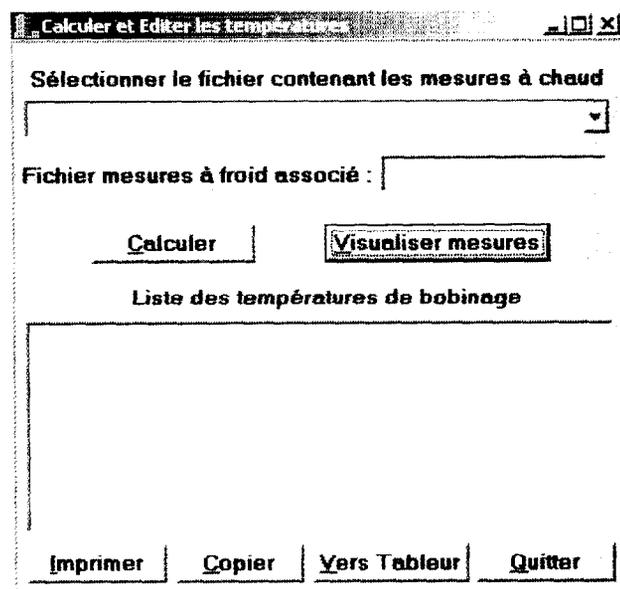
Si cette fenêtre est lancée à partir de la page d'accueil, on sélectionne d'abord le fichier contenant les mesures à chaud.

Une fois que les calculs ont été effectués, on a la possibilité de les imprimer dans une fiche standard. On doit trouver la référence de l'essai, la date ainsi que la résistance à froid, la température à chaud pour chaque bobinage ainsi que sa désignation qui sera saisie manuellement.

On veut aussi envoyer directement les informations vers un tableur pour qu'elles soient éditées.



Fenêtre lancée à partir de la boîte de dialogue : Mesure à chaud.



Fenêtre lancée à partir de la page d'accueil.

1.4.2 Eléments en relation avec le système

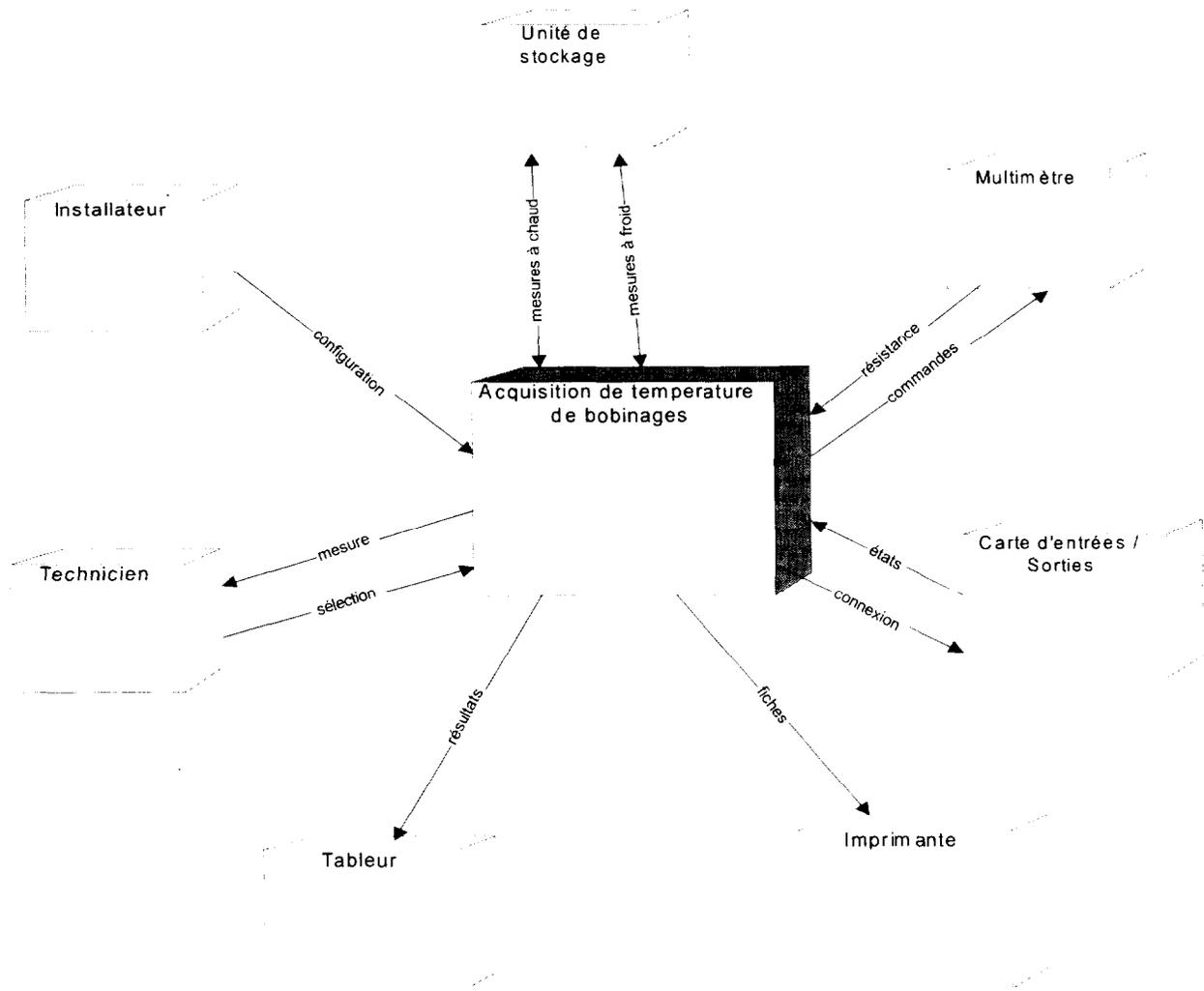


Figure 6 Diagramme de déploiement

Le technicien pilote le système pendant les mesures.
L'installateur configure le système notamment lors du 1^{er} lancement.
Les mesures à froid et à chaud sont sauvegardées sur l'unité de stockage (disque dur).
Les valeurs de résistance sont acquises à l'aide d'un multimètre IEEE.
Le rack de commutation permet de connecter un bobinage sur le multimètre ou sur le 230V.
Les données peuvent être transférées vers un tableur pour être exploitées.
Une fiche de mesures standard peut être imprimée.

□ **Rack de commutation**

Il permet de connecter les bobinages au multimètre et de commander les mesures sur celui-ci.

□ **Multimètre**

Les multimètres utilisés sont des HP34401A. Il sont commandés à partir du rack de commutation via le bus IEEE488.

□ **Tableur**

Le PC de contrôle fonctionne sous Windows 98, le tableur Excel est utilisé. Le transfert des résultats est réalisé via OLE (*object linking and embedding*).

□ **Unité de stockage**

L'unité de stockage est le fichier. Chaque série de mesures sera mémorisée dans un fichier qui sera stocké sur le disque dur du PC de contrôle.

1.4.3 Spécifications technologiques

Résistance d'un bobinage à froid :	entre 1 Ω et 50k Ω
Nombre de bobinages mesurés :	1 à 16 (8 par four au maximum)
Température de bobinage à déterminer :	entre 0 et 300 °C à $\pm 1^\circ\text{C}$
Température ambiante :	entre 20 et 30°C
Coefficient de température de la résistivité du cuivre:	234.5
Coefficient de température de la résistivité de l'aluminium:	225
Durée d'acquisition totale :	< 120s
Temps d'attente avant la 1 ^{ère} mesure :	5 à 10s
Précision de la mesure du temps :	1ms
Nombre de cycles de mesure :	3
Nombre de mesures par cycle :	5
Temps d'acquisition d'une mesure :	<=100ms
Nombre de fichiers de mesure directement sélectionnable:	5
Eléments paramétrables :	- répertoire de stockage des fichiers de mesures ; - adresse du multimètre ; - nom de l'interface IEEE.

1.5 Architecture matérielle

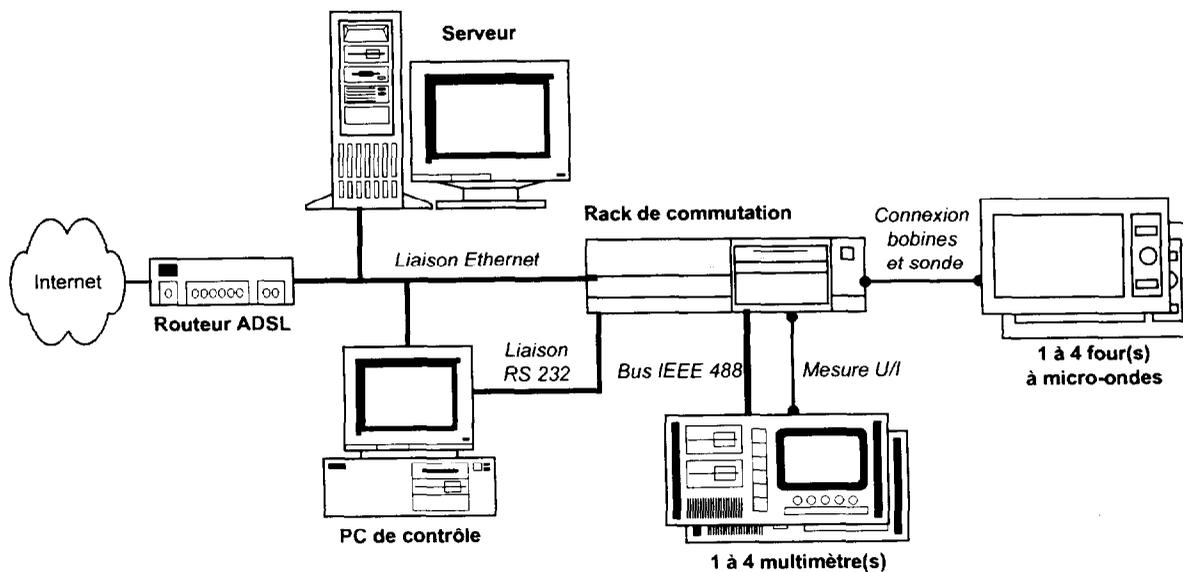


Figure 7 Eléments et liaisons

Le dialogue entre le PC de contrôle et le rack de commutation est réalisé essentiellement via la liaison Ethernet et s'appuie sur le protocole TCP/IP.

La liaison RS232 est seulement utilisée en phase d'initialisation pour configurer les paramètres réseau du rack (adresses IP du rack et du serveur, localisation du noyau temps réel à télécharger...). Elle peut également être utilisée pour transmettre des alertes ou des messages administratifs depuis le rack de communication vers le PC de contrôle.

Le serveur permet de stocker les résultats des mesures de différentes gammes de fours pour une exploitation locale. Il contient également l'environnement de développement utilisé, entre autre, par le PC de contrôle ainsi que les ressources du noyau temps réel VxWorks utilisé par le rack de commutation.

En phase de démarrage, à la mise sous tension du rack, celui-ci télécharge à partir du serveur un fichier correspondant au module du noyau VxWorks nécessaire au fonctionnement de l'application envisagée.

Le programme d'application, ici l'acquisition de température de bobinages, est ensuite chargé dans le rack à partir du PC contrôleur.

Le routeur ADSL permet une exploitation distante via Internet des résultats des mesures.

Le rack de commutation au format VME intègre une carte CPU BAB40 à base de processeur 68040, les ports RS232 et Ethernet sont intégrés à cette carte.

La carte CPU est associée à une première carte d'extension (VMOD-BAB) supportant 3 modules d'entrée/sortie :

- un module équipé d'un coupleur parallèle Z8536 (VMOD-TTL/O) ;
- deux modules dotés de relais (VMOD-RL) pour la connexion des bobinages du four.

Une deuxième carte d'extension au format VME est intégrée au rack (VMOD-IO), elle supporte 4 modules :

- un module composé d'une interface IEEE 488 pour le dialogue avec les multimètres (VMOD-GPIB) ;
- un module VMOD-TTL/O ;
- deux modules VMOD-RL.

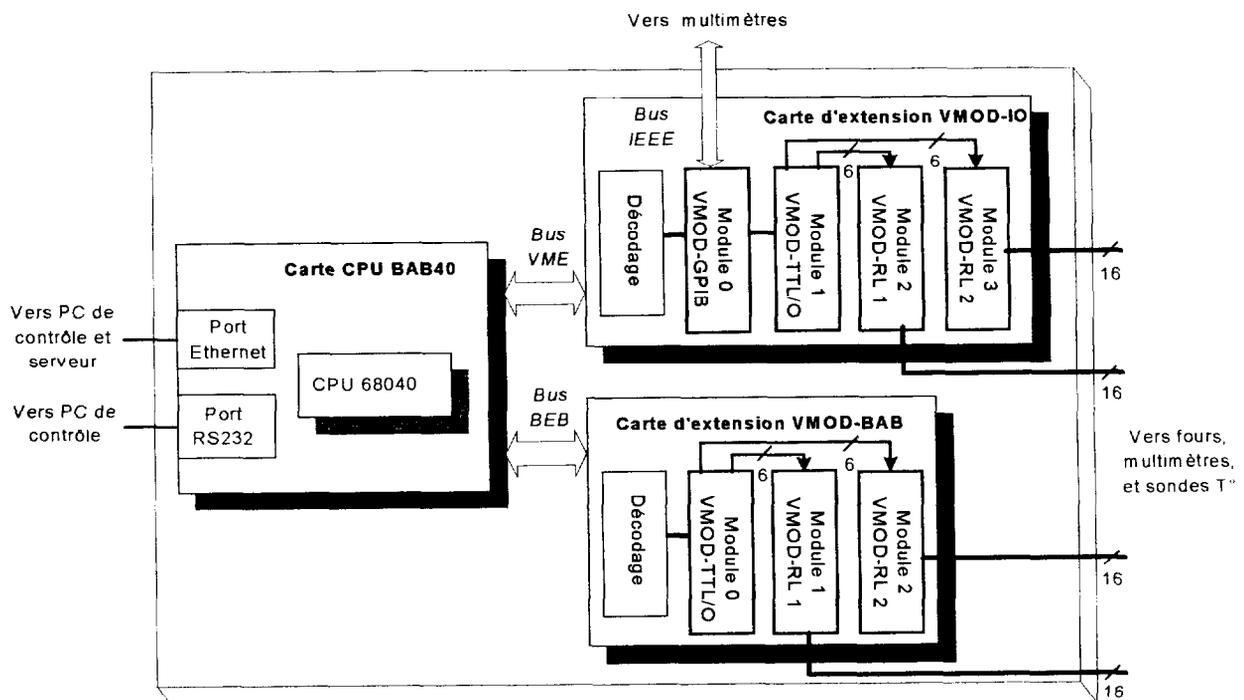


Figure 8 Rack de commutation

Pour chaque module relais, les six lignes d'entrée issues des modules TTL commandent six relais pour la commutation de 4 bobines, une sonde de température et l'alimentation 230 V.

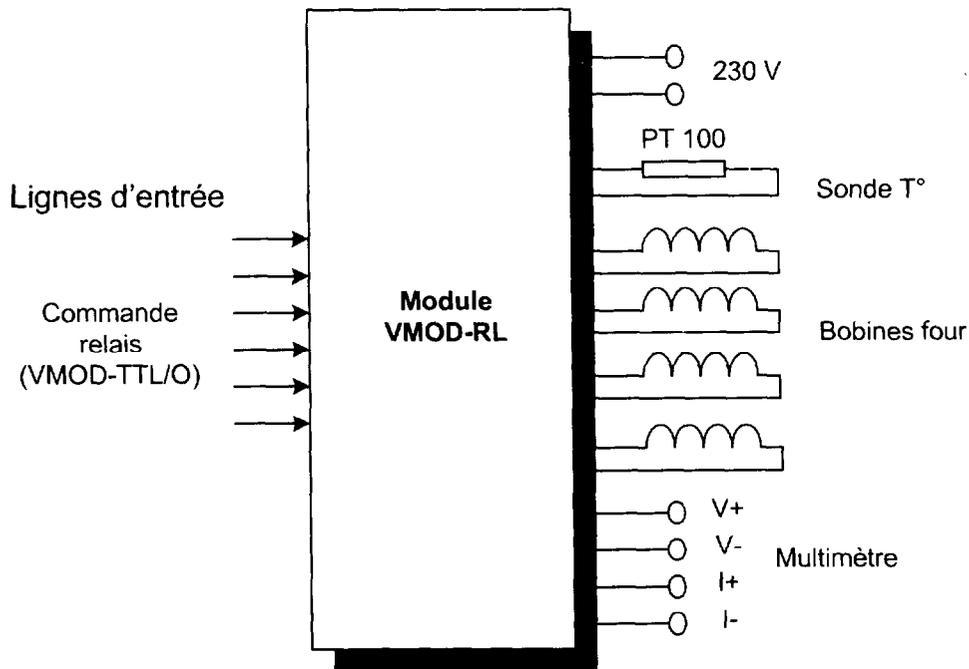


Figure 9 Module relais

1.6 Architecture logicielle

Les différents modules logiciels sont répartis entre le rack de commutation, le PC de contrôle et le serveur.

- Le rack de commutation supporte le noyau temps réel VxWorks. Comme vu au paragraphe précédent, l'application et les modules nécessaires sont téléchargés dans le rack de commutation lors de la mise en marche du système. La partie embarquée de l'application reçoit des commandes depuis le PC de contrôle et gère l'ensemble de ses ressources matérielles pour satisfaire à ces demandes.
- Le PC de contrôle supporte également une application développée en langage C++ qui gère l'IHM et le traitement des données.
- Le serveur stocke et archive les données transmises par le PC de contrôle. L'exploitation de ces données est faite quand le technicien le juge nécessaire, par l'intermédiaire d'applications développées sous Excel.

1.7 Analyse UML

1.7.1 Modélisation de la partie embarquée.

1.7.1.1 Diagramme de classes.

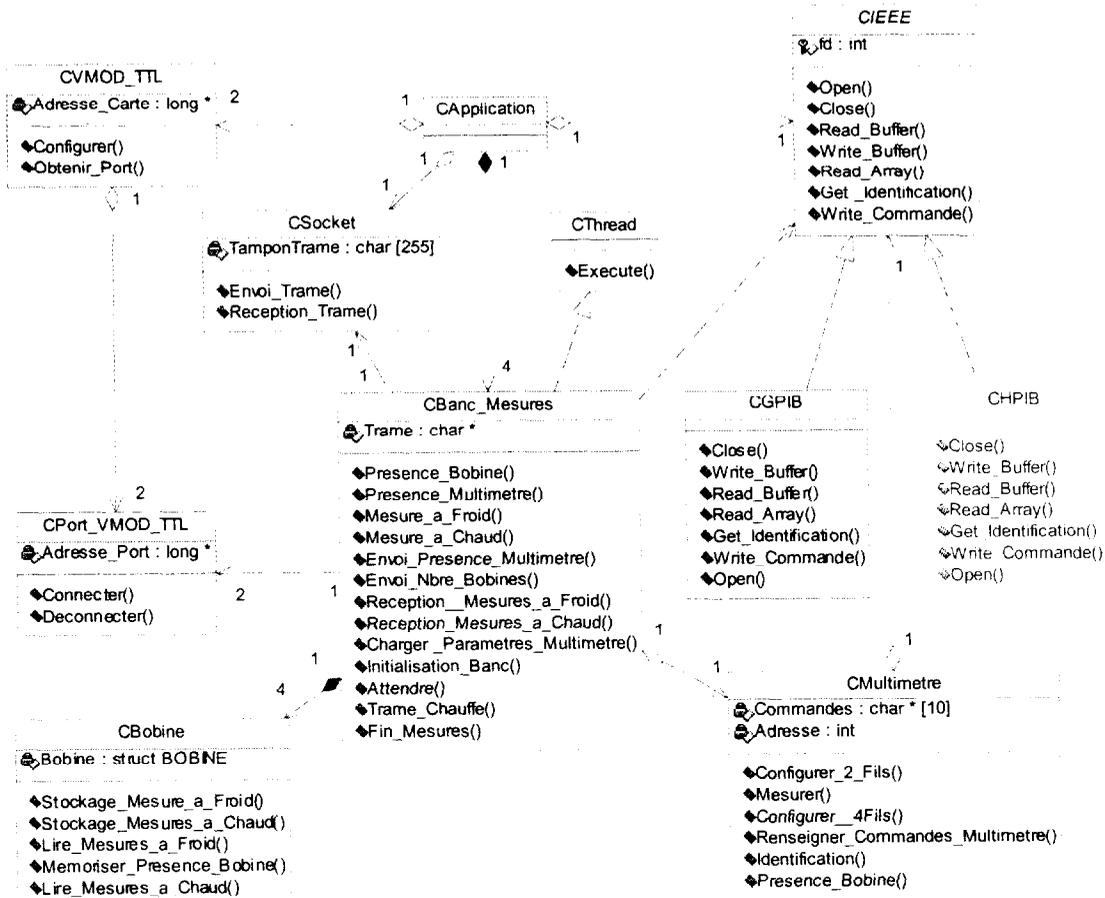


Figure 10 Diagramme de classes

Remarques :

- Seuls les attributs et les opérations essentiels à la compréhension du diagramme de classes sont représentés..
- La partie PC de contrôle (IHM) ne sera pas étudiée.
- Des questions seront posées tant sur la forme (questions de cours) que sur le fond (questions de compréhension) sur la représentation UML.
- Par convention tout nom précédé d'un « C » majuscule représentera le nom d'une classe : Ex : Csocket, Cbobine, Cthread
- La classe CHPIB représente la classe de gestion d'une carte IEEE différente, non utilisée dans le cadre actuel. Elle n'est présentée ici que pour montrer le rôle de la classe abstraite CIEEE.

1.7.1.2 Fonctionnement

L'application est téléchargée dans la carte CPU du rack de commutation. Elle est lancée immédiatement.

Lors de son instanciation l'objet **:CApplication** crée :

- un objet : **CSocket** en allocation dynamique ,
- deux objets : **CVMOD_TTL** en allocation dynamique ,
- un objet : **CGPIB** en allocation dynamique,
- quatre objets : **CBanc_Mesures** qui utiliseront la ressource partagée : CGPIB.

Ensuite elle lance les threads relatifs aux quatre banc de mesures.

Chaque objet : **CVMOD_TTL** crée

- deux objets : **CPort_VMOD_TTL** en allocation dynamique. Leurs adresses sont calculées par l'objet : **CVMOD_TTL** à partir de sa propre adresse.

Chaque objet : **Cbanc_Mesures** crée

- quatre objets : **CBobine**
- un objet : **CMultimetre** en allocation dynamique.

1.8 Ressources documentaires

1.8.1 UML

- Intégrer UML dans vos projets : Nathalie Lopez, Jorge Migueis, Emmanuel Pichon : Edition Eyrolles
- Modélisation objet avec UML: Auteur Pierre-Alain Muller :Edition Eyrolles
- Le guide de l'utilisateur UML: Auteurs Grady Booch, James Rumbaugh, Ivar Jacobson : Edition Eyrolles
- Visual Modeling with Rational Rose and UML :Auteur Terry Quatrani :Edition ADDISON-WESLEY.
- Rational Rose 98 using Rational Rose.

1.8.2 Réseaux

- La communication sous UNIX : Jean-Marie Rifflet : Edisciences
- TCP/IP : Mac Millan – S&SM
- Transmissions et Réseaux : S. Lohier D. Présent : Dunod

BTS INFORMATIQUE INDUSTRIELLE

Session 2001

Epreuve Etude d'un Système Informatisé

Dossier technique

Système d'acquisition de température de bobinages

DOCUMENTATION TECHNIQUE

Annexe D1	Norme de sécurité EN 60 335-1	5 pages
Annexe D2	Exemple de fiche de mesure	1 page
Annexe D3	Récapitulatif partiel des notations UML	4 pages
Annexe D4	Le bus d'instrumentation IEEE 488	6 pages
Annexe D5	Documentation des fonctions GPIB	2 pages
Annexe D6	Multimètre HP 34401A – Guide de l'interface	14 pages
Annexe D7	Programmation Réseau	8 pages

ANNEXE D1

Norme de sécurité EN 60 335-1

(extraits)

11. Echauffements

11.1 Les appareils et leur entourage ne doivent pas atteindre en usage normal des températures excessives.

La vérification consiste à déterminer les échauffements des différentes parties dans les conditions spécifiées dans les paragraphes 11.2 à 11.7, mais si l'échauffement d'un enroulement de moteur dépasse la valeur spécifiée dans le tableau du paragraphe 11.8 ou si il y a doute en ce qui concerne la classification du système d'isolation employé dans un moteur, par les essais du paragraphe 11.10.

Pour les appareils munis d'un enrouleur de câble automatique, la vérification est effectuée par l'essai complémentaire du paragraphe 11.9.

Dans le cadre des essais du paragraphe 11.10, il peut être nécessaire de mesurer, dans les conditions spécifiées dans les paragraphes 11.2 à 11.7, l'échauffement maximal atteint par l'enroulement du rotor et l'enroulement du stator des moteurs.

11.2 Les appareils portatifs sont suspendus dans leur position normale, en air calme.

Les appareils à encastrer sont encastrés dans des parois en contre-plaqué peint en noir mat, de 20mm d'épaisseur environ.

Les autres appareils de chauffage sont placés dans un coin d'essai. Le coin d'essai est constitué de deux parois à angle droit, d'un plancher et, si nécessaire, d'un plafond, ces parties étant en contre-plaqué peint en noir mat de 20mm d'épaisseur. L'appareil est placé dans le coin d'essai comme suit:

- *Les appareils utilisés normalement sur le sol ou sur une table sont placés sur le plancher aussi près que possible des parois.*
- *Les appareils qui sont normalement fixés à un mur sont installés sur l'une des parois, aussi près de l'autre paroi et du plancher, ou d'un plafond, qu'ils peuvent l'être en usage normal, à moins d'indications différentes données par le constructeur en ce qui concerne leur installation.*
- *Les appareils qui sont normalement fixés à un plafond sont installés au plafond, aussi près des parois qu'ils peuvent l'être en usage normal, à moins d'indications différentes données par le constructeur en ce qui concerne leur installation.*

Les autres appareils à moteur sont placés ou fixés sur un support de contre-plaqué peint en noir mat de 20 mm d'épaisseur environ, comme suit :

- *Les appareils normalement utilisés sur le sol ou sur une table sont placés sur un support horizontal.*
- *Les appareils normalement fixés à un mur sont montés sur un support vertical.*
- *Les appareils normalement fixés à un plafond sont fixés au-dessous d'un support horizontal.*

11.3 Les échauffements, autres que ceux des enroulements, sont déterminés au moyen de couples thermoélectriques à fil fin, choisis et disposés de façon à réduire au minimum leur influence sur la température de la partie à essayer.

Les couples thermoélectriques employés pour déterminer l'échauffement de la surface des parois, des planchers et des plafonds sont fixés sur la face intérieure de plaquettes en cuivre ou laiton noirci, de 15 mm de diamètre et de 1 mm d'épaisseur, encastrées de niveau avec la surface.

Autant qu'il est possible, la position de l'appareil est telle que les parties susceptibles d'atteindre les températures les plus élevées soient en contact avec les plaquettes.

Pour la détermination des échauffements des poignées, des boutons, des manettes et des organes analogues, sont prises en considération toutes les parties qui sont saisies en usage normal et, pour les organes en matière isolante, les parties en contact avec du métal chaud.

L'échauffement de l'isolation électrique, autre que celui des enroulements, est déterminé à la surface de l'isolation, aux endroits où un défaut pourrait provoquer un court-circuit, établir un contact entre les parties actives et les parties métalliques accessibles, provoquer un contournement de l'isolation ou réduire les lignes de fuite ou les distances dans l'air au-dessous des valeurs spécifiées au paragraphe 29.1.

Les échauffements des enroulements sont déterminés par la méthode de variation de résistances sauf si les enroulements ne sont pas uniformes ou si de sévères complications sont à attendre en faisant les connexions nécessaires, auquel cas les échauffements sont déterminés au moyen de thermocouples.

S'il est nécessaire de démonter l'appareil pour placer les thermocouples, la puissance est mesurée à nouveau pour vérifier que l'appareil a été remonté correctement.

Le point de ramification des conducteurs d'un câble ainsi que l'endroit où les conducteurs entrent dans les douilles sont des exemples d'endroits où les thermocouples sont disposés.

11.4 Les appareils de chauffage sont mis en fonctionnement les conditions de dégagement utile de chaleur, tous les éléments chauffants étant alimentés sous une tension telle que la puissance absorbée soit 1,15 fois la puissance nominale maximale.

11.5 Les appareils à moteur sont mis en fonctionnement sous la charge normale et sous la tension la plus défavorable comprise entre 0,94 fois la tension nominale minimale et 1,06 fois la tension nominale maximale.

11.6 Pour les appareils mixtes, lorsque les moteurs sont mis en fonctionnement sous une tension égale à 1,06 fois la tension nominale maximale, la puissance absorbée par les éléments chauffants est telle que spécifiée au paragraphe 11.4. Lorsque les moteurs sont mis en fonctionnement sous une tension égale à 0,94 fois la tension nominale minimale, la puissance absorbée par les éléments chauffants est réduite à 0,90 fois la puissance nominale.

S'il est nécessaire d'effectuer l'essai à une tension intermédiaire, la puissance absorbée par les éléments chauffants est réglée en proportion.

11.7 L'appareil est mis en fonctionnement :

- pendant la durée nominale de fonctionnement dans le cas des appareils pour service temporaire ;
- suivant des cycles consécutifs de fonctionnement, jusqu'à obtention de l'état de régime dans le cas des appareils pour service intermittent, les périodes de fonctionnement et de repos étant les périodes nominales de fonctionnement ;
- jusqu'à obtention de l'état de régime dans le cas des appareils pour service continu.

11.8 Pendant l'essai, les coupe-circuit thermiques ne doivent pas fonctionner, les échauffements doivent être surveillés en permanence et ne doivent pas dépasser les valeurs indiquées dans le tableau suivant et la matière de remplissage éventuelle ne doit pas couler.

Pour les appareils qui ne sont pas soumis à l'essai de l'article 12, les mesures spécifiées au paragraphe 13.1 sont effectuées à la fin du présent essai.

Parties	Echauffements deg C (K)
Enroulements ¹⁾ si l'isolation est :	
- en matière de la classe A ²⁾	75 (65)
- en matière de la classe E ²⁾	90 (80)
- en matière de la classe B ²⁾	95 (85)
- en matière de la classe F ²⁾	115
- en matière de la classe H ²⁾	140
Broches des socles de connecteurs :	
- pour conditions très chaudes	130
- pour conditions chaudes	95
- pour conditions froides	40
Bornes, y compris les bornes de terre, pour conducteurs externes des appareils fixes à moins qu'ils ne soient munis de câbles d'alimentation	60
Ambiance des interrupteurs et thermostats ³⁾ :	
- non marqués T	30
- marqués T	T-25
Enveloppe isolante en caoutchouc ou en polychlorure de vinyle des conducteurs internes et externes y compris les câbles d'alimentation :	
- non marqués T	50 ⁴⁾
- marqués T	T-25

Gaine de câble utilisée comme isolation supplémentaire	35
Caoutchouc autre que synthétique employé pour des bagues d'étanchéité ou autres parties dont la détérioration pourrait affecter la sécurité :	
- lorsqu'il est utilisé comme isolation supplémentaire ou comme isolation renforcée	40
- dans les autres cas	50
Douilles E 26 et E 27:	
- du type métallique ou céramique	160
- du type en matière isolante autre que céramique	120
- marquées T	T-25
Douilles E 14, B 15 et B 22 :	
- du type métallique ou céramique	130
- du type en matière isolante autre que céramique	90
- marquées T	T-25
Matières utilisées pour l'isolation autres que celles spécifiées pour les conducteurs et les enroulements ⁶⁾ :	
- textiles, papier ou carton imprégnés ou vernis	70
- stratifiés agglomérés avec :	
• des résines mélamine-formaldéhyde, phénol-formaldéhyde ou phénol-furfural	85 (175)
• résine à base d'urée formaldéhyde	65 (150)
- matières moulées :	
• phénol-formaldéhyde à charge cellulosique	85 (175)
• phénol-formaldéhyde à charge minérale	100 (200)
• mélamine-formaldéhyde	75 (150)
• urée formaldéhyde	65 (150)
- polyester renforcé de fibre de verre	110
- caoutchouc ou silicone	145
- polytétrafluoréthylène	265
- mica pur et les matériaux en céramique fortement frittés lorsque ces matériaux sont utilisés comme isolation supplémentaire ou comme isolation renforcée	400
- matières thermoplastiques ⁷⁾	-
Bois en général ⁸⁾	65
- supports, parois, plafond, plancher en bois du coin d'essai et parois légères en bois :	
• appareils fixes spécifiquement mentionnés dans une deuxième partie comme capables de fonctionner en permanence pendant de longues périodes	60
• autres appareils	65
Surfaces extérieures des condensateurs :	
- avec indication de la température maximale de fonctionnement (T)	T-35
- sans indication de la température maximale de fonctionnement	
• petits condensateurs céramiques pour la réduction des perturbations de la radiodiffusion et de la télévision	50
• autres condensateurs	20
Enveloppe extérieure des appareils sans éléments chauffants, sauf les poignées qui sont tenues en usage normal	60
Poignées, boutons, manettes et organes analogues qui, en usage normal, sont tenus de façon continue (par exemple, dans les fers à souder) :	
- en métal	30
- en porcelaine ou matière vitrifiée	40
- en matière moulée, caoutchouc ou bois	50
Poignées, boulons, manettes et organes analogues qui, en usage normal, ne sont tenus que pendant de courtes périodes (par exemple des interrupteurs) :	
- en métal	35
- en porcelaine ou matière vitrifiée	45

- en matière moulée, caoutchouc ou bois	60
Parties en contact avec de l'huile ayant un point d'éclair de t °C	t-50
Tout point où l'isolation d'un conducteur peut entrer en contact avec une boîte à bornes ou un compartiment utilisé pour la connexion à une canalisation fixe d'un appareil fixe qui n'est pas muni de câbles d'alimentation :	
- lorsque la notice d'instructions prescrit l'utilisation de conducteurs d'alimentation marqués T	T-25 ⁵⁾
- dans les autres cas	50 ⁴⁾

¹⁾ Pour tenir compte du fait que la température des enroulements des moteurs universels, des relais, des solénoïdes, etc., est généralement supérieure à la moyenne aux points où sont placés les couples thermoélectriques, les valeurs qui ne sont pas entre parenthèses sont applicables quand la méthode de la résistance est employée, et les valeurs entre parenthèses s'appliquent lorsque des thermocouples sont utilisés. Pour les enroulements de vibreurs et des moteurs à courant alternatif les valeurs qui ne sont pas entre parenthèses s'appliquent dans les deux cas.

²⁾ La classification est conforme à la Publication 85 de la CEI : Recommandations relatives à la classification des matières destinées à l'isolement des machines et appareils électriques en fonction de leur stabilité thermique en service.

Comme exemples de matières de la classe A, on peut citer :

- le coton, la soie naturelle, la soie artificielle et le papier imprégnés;
- les émaux oléorésineux ou à base de résines polyamides.

Comme exemples de matières de la classe B, on peut citer :

- l'amiante, la fibre de verre, les résines mélamine-formaldéhyde, phénol-formaldéhyde.

Comme exemples de matières de la classe E, on peut citer :

- des résines moulées à charge cellulosique, les stratifiés coton et les stratifiés papier, agglomérés avec des résines mélamine-formaldéhyde, phénol-formaldéhyde ou phénol-furfural ;
- les résines polyester à chaînes transversales, les films de triacétate de cellulose, les films de téréphtalate de polyéthylène ;
- les toiles vernies à base de téréphtalate de polyéthylène agglomérées avec des vernis à base de résines alkydes modifiés à l'huile;
- les émaux à base de résines formal-polyvinile. Polyuréthane ou époxyde.

Pour les moteurs entièrement fermés, des limites d'échauffement pour les matières de la classe A, de la classe E, de la classe B, de la classe F et de la classe H, peuvent être augmentées de 5 deg C (5 K).

Un moteur fermé est un moteur construit de façon à empêcher la circulation de l'air entre l'intérieur et l'extérieur de l'enveloppe mais non suffisamment enfermé pour être considéré comme hermétique (airlight).

³⁾ T signifie la température maximale de fonctionnement.

L'ambiance des interrupteurs et thermostats est la température de l'air au point le plus chaud à une distance de 5 mm de la surface de l'élément constituant considéré.

Dans le cadre de cet essai, les interrupteurs et les thermostats s'ils portent l'indication des caractéristiques nominales individuelles, peuvent être considérés comme ne portant pas l'indication de la température maximale de fonctionnement, si le constructeur de l'appareil complet le demande.

⁴⁾ Cette limite est applicable aux câbles, cordons et fils conformes aux HD 21 ou HD 22. Pour les autres, elle peut être différente.

⁵⁾ Cette limite deviendra applicable aussitôt qu'il existera des normes de la CEI relatives aux enroulements et aux câbles souples à température élevée.

⁶⁾ Les valeurs entre parenthèses s'appliquent, si la matière est utilisée pour des poignées, des boutons, des manettes et des organes analogues et est en contact avec du métal chaud.

⁷⁾ Il n'est pas fixé de limite particulière pour les matières thermoplastiques, qui doivent satisfaire aux essais du paragraphe 30.1 ou 30.2, en vue desquels les échauffements doivent être déterminés.

⁸⁾ La limite spécifiée concerne la détérioration du bois : elle ne tient pas compte de la détérioration des finis de surface.

S'il est fait usage de ces matières ou d'autres, elles ne doivent pas être soumises à des températures supérieures à leurs possibilités telles qu'elles ont été déterminées par des essais de vieillissement sur ces matières.

Les valeurs du tableau sont basées sur une température ambiante ne dépassant pas habituellement 25 °C, mais pouvant atteindre, occasionnellement 35 °C. Toutefois, les échauffements spécifiés sont basés sur une température ambiante de 25 °C. La valeur de l'échauffement d'un enroulement en cuivre est calculée à partir de la formule :

$$\Delta t = \frac{(R_2 - R_1)(234,5 + t_1)}{R_1} - (t_2 - t_1)$$

où:

Δt est l'échauffement

R_1 est la résistance au début de l'essai

R_2 est la résistance à la fin de l'essai

t_1 est la température ambiante au début de l'essai

t_2 est la température ambiante à la fin de l'essai

Au début de l'essai, les enroulements doivent se trouver à la température ambiante.

Il est recommandé de déterminer la résistance des enroulements à la fin de l'essai en effectuant des mesures de résistance aussitôt que possible après ouverture du circuit, puis à des intervalles rapprochés de façon à pouvoir tracer une courbe de variation de la résistance en fonction du temps pour déterminer la résistance au moment de l'ouverture du circuit.

ANNEXE D2

Exemple de fiche de mesure

CEFEMO	STANDARD LABO	R N	E	8	
TEST : ECHAUFFEMENT		§11	TYPE : AK2CS50V		
Matériel utilisé :		DATE : 11/01/00			
Banc de mesure N° CEFEMO : 130002		ISSUE : M. BETHESS			

Normes :	60-335-1	60-335-2-6	60-335-2-25	Autres
----------	----------	------------	-------------	--------

Configuration :	
N° du four :	98-62

Nombre de cycle :	3	Durée :	1x 9min + 6min MW + 30min Grill	Fichier :	Mwgril
Puissance :	1,15 Pn W	Position du four :	Black corner		
Tension :	254 V	Capa :	1,077 µF		
Type d'essai :	MICROWAVE + Grill				

N°	Désignation	Limit K	Ech. K
1	Anode	*	171
2	HV Capacitor	60	26
3	Support plastic HV	*	33
4	Heater connection temperature	*	41
5	Internal wiring HV	125	28
6	Supply cord separation point	50	31
7	Cord bushing	*	30
8	R.T.motor amb.	80	41
9	Fan Motor amb.	155	50

N°	Désignation	Limit K	Ech. K
10	Magnetron limiter	150	49
11	Noise filter	95	50
12	Primary switch	100	49
13	Control/Secondary switch	100	47
14	VPC / Grill relay amb.	60	35
15	Grill thermal cut-out	150	32
16	Internal wiring silicon	155	36
17			
18	Température ambiante	25°C	23

Voie	Enroulement	R° ohms	Limit K	Cycle 1	Cycle 2	Cycle 3			
1	Primary	1,3774	160	80	94	28			
2	Fan Motor	105,05	115			86			
3	RT Motor	12385	115			72			
4	LV Transformer	713,97	95			38			
5									
6									
7									
8									
9	Secondary	99,178	160	64	86	29			

Puissance absorbée à 10 secondes en W	1847	1649	1784			
Courant de fuites maxi en mA						

Commentaires sur l'essai :

ANNEXE D3

RECAPITULATIF PARTIEL DES NOTATIONS UML 1.0

DIAGRAMME DE CAS D'UTILISATION

Acteur, cas d'utilisation et association

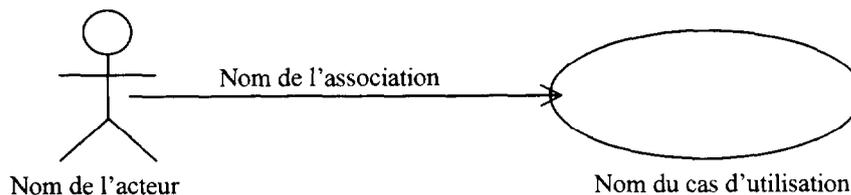
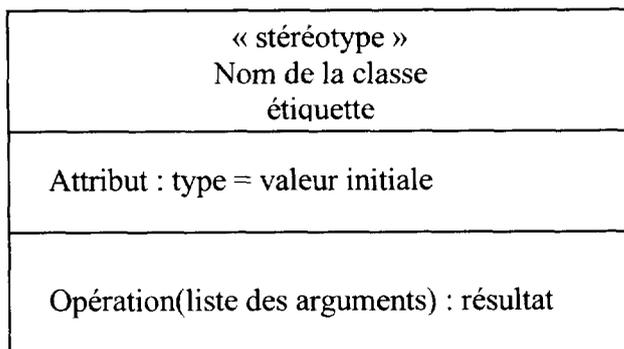
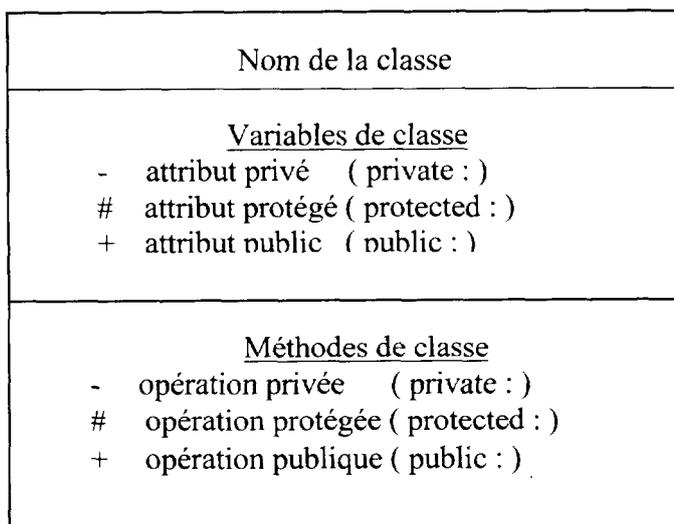


DIAGRAMME DE CLASSES

Classe, attribut et opérations



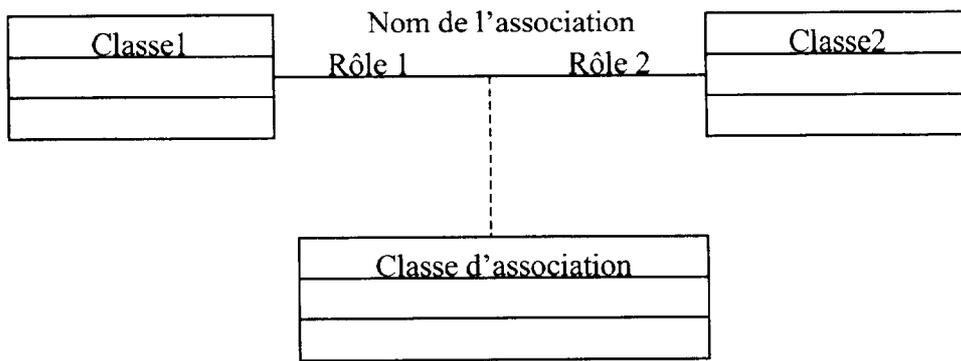
Accès aux propriétés



Visibilité : Symboles graphiques Rational Rose

- (+) Attribut ou opération public
- (#) Attribut ou opération protégé
- (-) Attribut ou opération privé

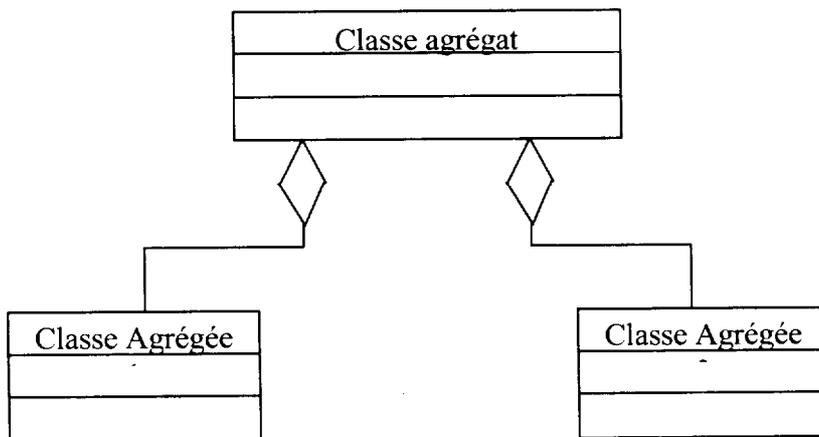
Association, classe d'association et nom de rôle



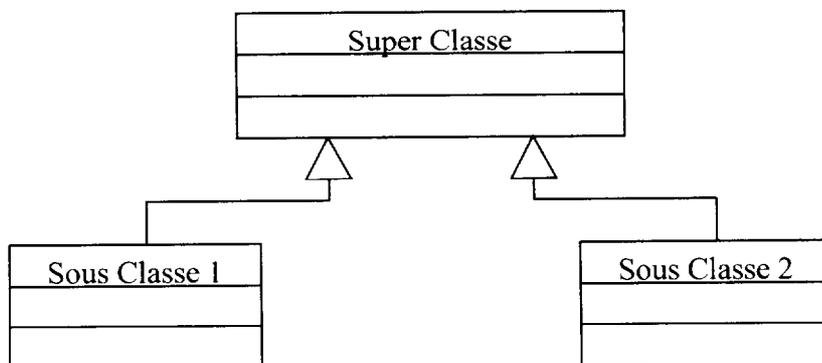
Cardinalité

1	Un exactement
0..1	Zéro ou un
0..*	Zéro ou plusieurs
1..*	Au moins un
*	plusieurs

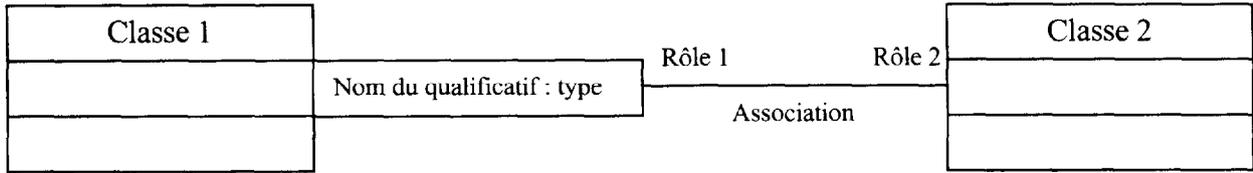
Agrégation



Héritage



Qualificatif



Contrainte

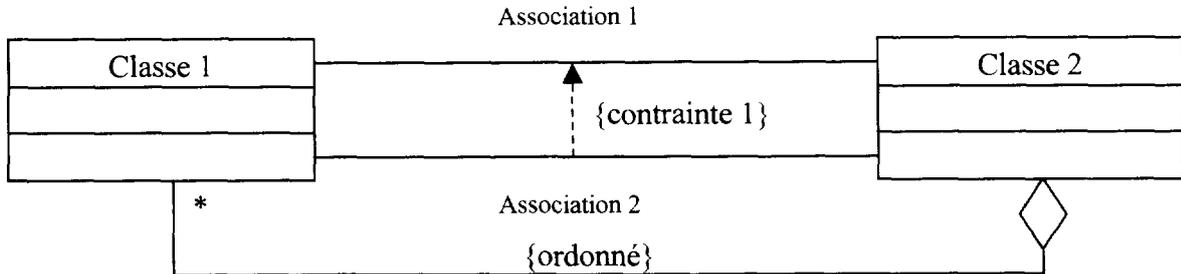
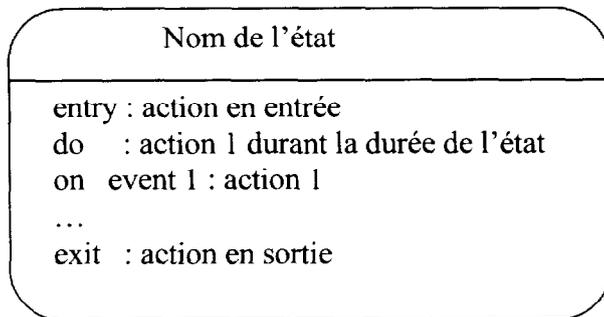
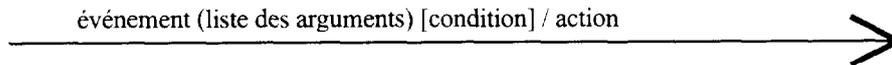


DIAGRAMME D'ETATS

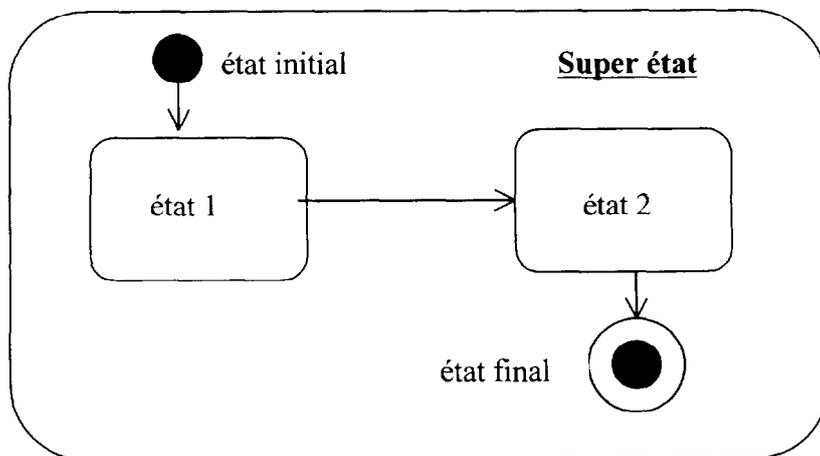
Etat



Transition



Imbrication d'états



SCENARIOS

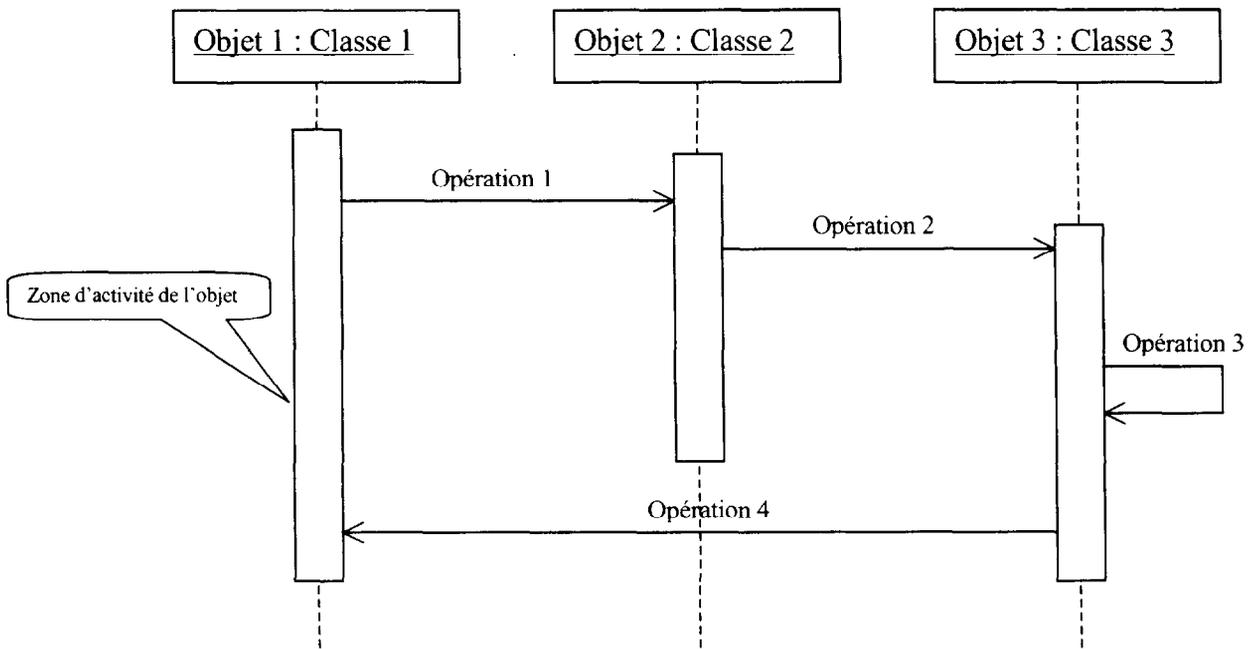
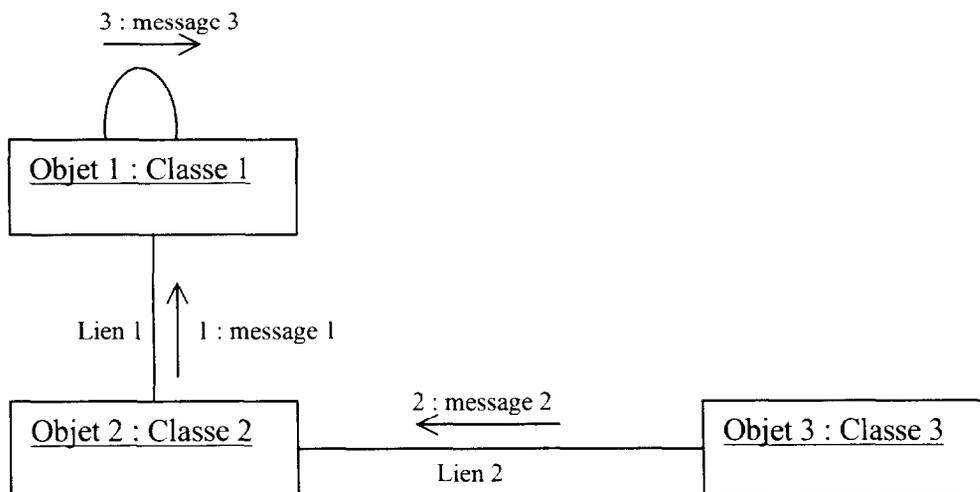


DIAGRAMME DE COLLABORATION



ANNEXE D4

Le Bus d'instrumentation IEEE 488 (GPIB)

1. Caractéristiques générales

Le bus IEEE488 ou GPIB est un bus parallèle permettant de relier des appareils de mesure à un ordinateur et possédant les caractéristiques principales suivantes :

- 15 appareils au maximum connectés ;
- Câblage en étoile ou en bus sur une longueur maximale de 20 m ;
- vitesse maximale de 1 Mo/s.

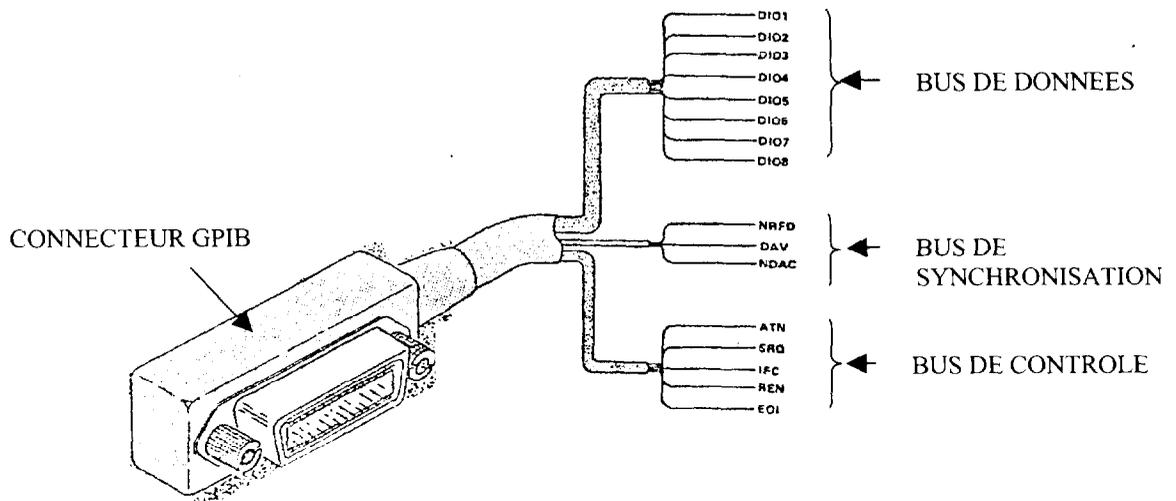
Trois fonctions sont définies sur le bus :

- La fonction **contrôleur**, en général assurée par un ordinateur ou un micro-ordinateur, se charge de la gestion des transferts d'information sur le bus.
- La fonction **parleur** (*talker*) est affectée à un appareil à la fois par le contrôleur à un instant donné. Celui-ci envoie ses informations aux écouteurs.
- La fonction **écouteur** (*listener*) est attribuée à un ou plusieurs appareils par le contrôleur.

Les échanges peuvent être effectués en mode **commande** (le contrôleur commande les appareils et désigne le parleur et les écouteurs) ou en mode **transfert d'informations** (le parleur envoie ses informations vers les écouteurs)

2. Description physique

Le bus est physiquement constitué par l'ensemble des câbles de liaison qui transportent les informations d'un appareil à l'autre dans n'importe quel sens. Il s'agit d'un câblage passif de seize lignes, une masse logique, cinq blindages partiels et un blindage général.



Cet ensemble de lignes est composé de trois groupes fonctionnels :

a) ***Le bus de données*** est constitué des lignes DIO1 à DIO8 (DIO = **D**ata **I**n **O**ut) qui servent à transporter les informations proprement dites. Celles-ci peuvent, suivant les circonstances être :

- Des données numériques, alphanumériques ou binaires.
- Des adresses de périphériques.
- Des commandes normalisées (multi-lignes).
- Des mots d'état (*status byte*).

b) ***Le bus de synchronisation (handshake)*** est composé de trois lignes qui transportent les signaux de contrôle. Ils garantissent la fiabilité du transfert des données circulant sur le bus suivant un protocole de type « poignée de main » (*handshake*).

Ce protocole possède les particularités suivantes :

- Le transfert des données s'effectue de manière asynchrone. Le taux d'échange s'ajuste automatiquement à la vitesse de l'émetteur et du ou des récepteurs, plus exactement à la vitesse de l'équipement le plus lent.
- Tous les octets sont transmis sur le bus de données suivant ce protocole.
- Un ou plusieurs appareils peuvent accepter les données.
- Lorsque des commandes transitent sur le bus, l'appareil le plus lent déterminent le taux de transfert de ces ordres. Tous les équipements participent à ce protocole.
- La vitesse de transfert peut diminuer, le temps qu'un appareil prenne une lecture et la retourne ensuite pour qu'elle soit entièrement assimilée par le contrôleur.

Les trois lignes du bus sont :

NOM	DESCRIPTION
DAV	<u>D</u>ata <u>V</u>alid (donnée correcte) : cette ligne à l'état bas confirme que les informations présentes sur le bus de données sont valides et peuvent être acceptées en toute sécurité par les équipements concernés. Le contrôleur, comme tout émetteur connecté sur le bus, pilote cette ligne lorsqu'il envoie des octets.
NRFD	<u>N</u>ot <u>R</u>eady <u>F</u>or <u>D</u>ata (pas prêt pour les données) : active à l'état bas, cette ligne permet de signaler qu'un appareil peut ou ne peut pas accepter des informations. Tous les équipements qui reçoivent des commandes, pilotent cette ligne comme les récepteurs auxquels on adresse des données.
NDAC	<u>N</u>ot <u>D</u>ata <u>A</u>Ccepted (données non acceptées) : l'appareil signale, grâce à cette ligne, s'il accepte ou refuse les informations reçues. Une fois encore, tous les équipements sollicités par des commandes contrôlent ce signal, comme les récepteurs destinataires de données. La ligne NDAC ne remonte pas au niveau haut, tant que le dernier et le plus lent des équipements récepteurs n'a pas approuvé les données.

c) ***Le bus de contrôle et commande (contrôle)*** : composé de 5 lignes, il permet à un calculateur spécialisé de gérer les appareils interconnectés. Ces lignes veillent à la

circulation ordonnée des informations sur le bus. Il s'agit de commandes dites *uniligne*, puisqu'elles n'utilisent qu'un seul fil.

Ces cinq lignes sont :

NOM	DESCRIPTION
ATN	ATtention : lors de la validation, tous les instruments deviennent récepteur et participent à la communication. Ils doivent répondre dans un délai de 200 μ s. Ce signal signifie aux équipements la présence d'un message de commande ou de donnée sur le bus. Au niveau bas, ATN prévient tous les appareils qu'une commande IEEE 488 se trouve sur le bus (un ordre ou bien une adresse). Seul le contrôleur active cette ligne.
IFC	InterFace Clear (Remise à zéro de l'interface) : cette ligne, uniquement pilotée par le contrôleur en charge, permet à ce dernier d'arrêter à tout instant (de manière asynchrone) l'opération en cours sur le bus. Tous les équipements doivent en permanence tester la ligne IFC et répondre, en cas de sollicitation, en 100 μ s. Ce signal correspond au <i>reset</i> général du bus IEEE 488.
REN	Remote ENable (pilotage par le bus validé) : son niveau, exclusivement géré par le contrôleur en charge, force chaque appareil concerné à être piloté par le bus
SRQ	Service ReQuest (demande de service) : un appareil utilise cette ligne pour demander la parole et, éventuellement, interrompre l'activité du bus (mise en œuvre d'interruption possible). L'utilisation typique de ce signal, consiste à signaler la disponibilité d'une donnée ou bien avertir le contrôleur d'un problème quelconque sur un instrument. Pour déterminer l'équipement qui réclame son attention, le contrôleur effectue ce que l'on appelle un <i>polling</i> (sondage) : il interroge individuellement chaque appareil (<i>serial poll</i>) ou bien tous les récepteurs à la fois (<i>parallel poll</i>).
EOI	End Or Identify : lorsque la ligne ATN se trouve à l'état haut (FALSE), le contrôleur ou un instrument du bus peut activer la ligne EOI et la passer à l'état haut afin d'indiquer une fin de transmission. <ul style="list-style-type: none"> • EOI signal la fin des données. • EOI témoigne de la conduction d'un <i>polling</i> parallèle. • EOI est activée par l'instrument qui émet.

1. Adressage des appareils

Chaque appareil possède au minimum une adresse d'écouteur (*Listener*) ou de parleur (*Talker*), celle-ci est configurée matériellement (commutateurs) ou logiciellement.

L'adressage des appareils peut être assuré sur deux niveaux, primaire et secondaire. A l'adresse primaire correspond une « fonction simple » mais si celle-ci est « étendue », elle bénéficiera d'un second niveau d'adressage (secondaire). Le système comporte ainsi 31 adresses primaires et autant de secondaires. C'est au contrôleur actif qu'il revient d'adresser les appareils.

Pour cela, il émet en mode commande un octet comportant :

- un bit de parité P (poids fort) ;
- un code sur 2 bits définissant le mode ;
- 5 bits d'adresse.

Le code sur deux bits est le suivant :

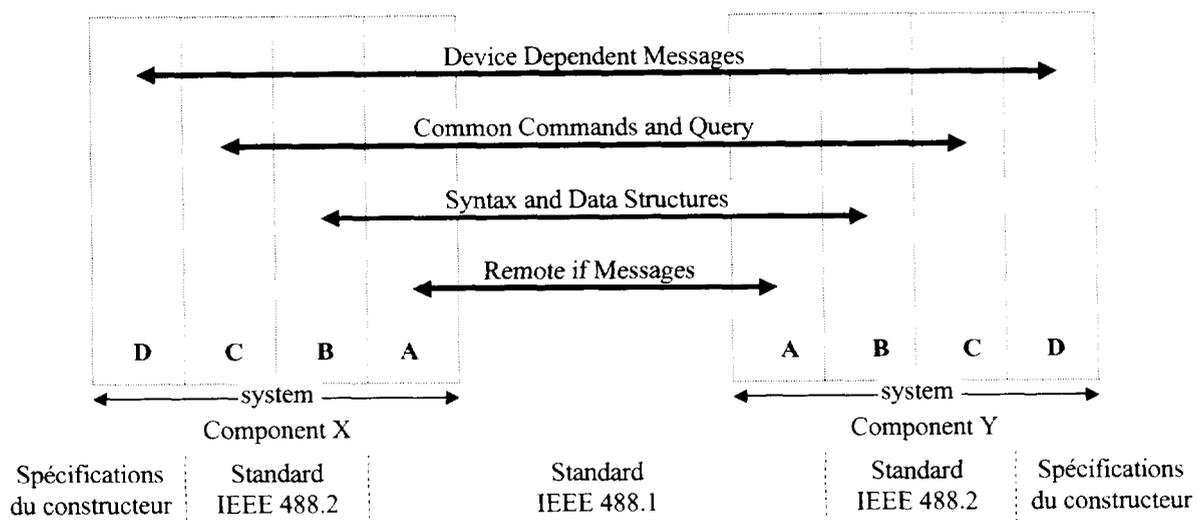
- 01 indique qu'il s'agit d'une adresse primaire d'écouteur ;
- 10 indique que l'adresse primaire est celle d'un parleur ;
- 11 est formé pour une commande secondaire ;
- 00 pour une commande auxiliaire.

Par exemple, pour un écouteur configuré à l'adresse 9, le code 41 (29 en hexadécimal et 00101001 en binaire) correspondant au caractère ASCII «) » est envoyé par le contrôleur.

Il est évident que si des appareils se voient attribuer une adresse commune, ils seront activés en même temps.

Dans le cas de commandes auxiliaires, un format sur un octet tel que P001XXXX (P étant le bit de parité) est une commande auxiliaire adressée.

4. Normalisation en couches de l'interface IEE 488



- **La couche A (Remote if Messages)** est la couche la plus basse de l'interface GPIB, elle décrit la partie physique du contrôleur (connecteurs, câblages, signaux électriques...) ainsi qu'un ensemble de commandes de base.
- **Les couche B et C (Syntax and Data Structures et Common Commands and Query)** représentent les fonctions de communication bas-niveau définies par l'IEEE 488.2.
- **La couche D (Device Dependent Messages)** définit la syntaxe mise en œuvre par le constructeur pour piloter son équipement, le langage SCPI (*Standard Commands for Programmable Instruments*) en est un exemple.

Les constructeurs fournissent généralement des bibliothèques de fonctions logicielles permettant de transmettre les commandes IEE 488.2 ainsi que les commandes SCPI. La librairie SICL (*Standard Instrument Control Library*) fournie par HP en est un exemple.

5. Les commandes IEEE 488.1 :

Quatre types de commande peuvent être transmises lorsque le bus fonctionne en mode commande (ATN = 1). A chacune de ces commandes correspond un code sur 8 bits.

- Les adresses d'écouteur (LAG : *Listen Address Group*) et de parleur (TAG : *Talken Address Group*) sont décrites précédemment.
- Les commandes multilignes universelles (UCG : *Universel Command Group*) sont reçues par tous les appareils, elles permettent de déclencher une action particulière remise à zéro d'un appareil par exemple).
- Les commandes adressées (ACG : *Addressed Command Group*) concernent les appareils préalablement initialisés comme écouteurs. Elles permettent au contrôleur d'envoyer une commande simultanée, de synchronisation par exemple.
- Les deux commandes non adressées qui peuvent être considérées comme des extensions des adresses d'écouteur et de parleur.

6. Les commandes IEEE 488.2 :

Au niveau supérieur, la norme IEEE 488.2 met en place un certain nombre de commandes acceptées par tous les appareils respectant le standard. Ces commandes sont envoyées sur le bus en mode données (ATN = 0) et correspondent à des chaînes de caractères ASCII (Mnemonic). Certaines commandes sont obligatoires, d'autres optionnelles.

***IDN ?**

IDeNtification query : demande à l'instrument de s'identifier par l'envoi d'un message standard, constitué de quatre champs séparés par des virgules :

- Champ 1 : fabricant Obligatoire,
- Champ 2 : modèle Obligatoire,
- Champ 3 : numéro de série ASCII "0" si non disponible (48 décimal),
- Champ 4 : révision du programme ASCII "0" si non disponible (48 décimal),

Par exemple, HEWLETT-PACKARD, 347A, 2221A01113, A1. La longueur maximale ne peut excéder 72 caractères.

***RST**

ReSeT : stoppe toutes les opérations en cours et réinitialise l'appareil dans un état prédéterminé.

***TST ?**

Self Test Query : cette commande ordonne à l'instrument de lancer une procédure d'autotest. Le résultat peut varier de -32 767 à +12 767. Un zéro indique la bonne marche des opérations. La documentation renseignera l'utilisateur sur les causes d'une réponse non nulle.

***OPC**

Operation complete : demande à l'instrument d'armer le bit 0 dans le registre SESR à la fin du travail en cours. Ainsi, par une programmation adéquate du masque *Service Request Enable Register*, l'équipement déclenchera une demande de service lorsque la commande en cours d'exécution sera achevée.

***OPC ?**

Operation Complete query : cette instruction impose à l'appareil de placer la valeur ASCII "1" (49 en décimal) dans son tampon de sortie, lorsqu'il a terminé tout travail en cours.

***WAI**

WAI tu continue : cet ordre force l'appareil à attendre la fin des commandes qu'il a entreprises. Par exemple, lancer une opération de calibration, patienter jusqu'à sa fin par *WAI, puis lancer une mesure.

***CLS**

CLear Status : cet ordre remet à zéro le mot d'état ainsi que toutes les structures de données qui lui sont associées, comme l'Event status Register par exemple. Il initialise également tous les tampons, à l'exception de celui de sortie(Output Queue).

***ESE Standard**

Event status Enable : la commande *ESE permet de sélectionner la participation des événements au bit final ESB. Par exemple, *ESE 36 valide les bits 5 (*command Error*) et 2 (*Query Error*).

***ESE ?**

Event status Enable query : cet ordre lit et retourne le contenu du registre SESR. La valeur évolue entre 0 et 255.

***ESR ?**

Event status Register query : cette commande expédie à l'utilisateur la valeur associée au registre d'événement. Son décodage ultérieur renseignera sur les actions entreprises au sein de l'instrument. Sa lecture initialise à zéro ce registre.

***SRE**

Service Request Enable : cet ordre, suivi d'un masque, déterminera le ou les bits du mot d'état qui déclencheront une demande de service. Par exemple, pour permettre au bit 4 (*Message Available MAV*) de valider un SRQ, on expédiera *SRE 16.

***SRE?**

Service Request Enable query : retourne la valeur du masque de programmation expédié par la commande *SRE. La gamme s'étend de 0 à 63 ou de 128 à 191. En effet, le bit 6 (RQS) ne peut être programmé.

***STB ?** SStatus Byte query : cette commande lit le mot d'état, avec le bit 6 *Master Summary Status* en lieu et place de RQS. La réponse est un entier compris entre 0 et 255.

ANNEXE D5

DOCUMENTATION DES FONCTIONS GPIB

NOM gpib - fonctions de la bibliothèque GPIB

PROTOTYPES

```
int gpib_set_timeout(int fd, int val);
int gpib_print(int fd, int addr, char *data);
int gpib_input(int fd, int addr, char *buffer);
int gpib_getbuffer(int fd, int addr, char *buffer, int count);
int gpib_sendbuffer(int fd, int addr, char *data, int count);
int gpib_defeoi(int fd, int c);
int gpib_sendcmd(int fd, int cmd);
```

DESCRIPTION

Ces fonctions permettent de piloter le module GPIB et d'accéder aux données des périphériques IEEE connectés.

Avant de pouvoir être utilisées, ces fonctions doivent disposer d'un descripteur de fichier (fd) qui ne pourra être obtenu qu'après initialisation du driver (voir `mgpibDrv()`) et création du périphérique correspondant à la carte ou au module GPIB (voir `mgpibDevCreate()`)...

int gpib_set_timeout(int fd, int val)

FONCTION: gpib_set_timeout - set the GPIB timeout counter

PARAMETER: (int)fd - file descriptor obtained from an open
(int)val - timeout value in ms

RETURN: 0 - without any error
-1 - function failed, errno is set

int gpib_print(int fd, int addr, char *data)

FONCTION: gpib_print - send string to the addressed device

PARAMETER: (int)fd - file descriptor obtained from an open
(int)addr - device address
(char *)data - buffer address

RETURN: 0 - without any error
-1 - function failed, errno is set

int gpib_input(int fd, int addr, char *buffer)

FONCTION: gpib_input - read string from the addressed device

PARAMETER: (int)fd - file descriptor obtained from an open
(int)addr - device address
(char *)buffer - buffer address

RETURN: >= 0 - # of bytes
-1 - function failed, errno is set

int gpib_getbuffer(int fd, int addr, char *buffer, int count)

FONCTION: gpib_getbuffer - read data from the addressed device

PARAMETER: (int)fd - file descriptor obtained from an open
(int)addr - device address
(char *)buffer - buffer address
(int)count - # of bytes to read

RETURN: 0 - without any error
-1 - function failed, errno is set

int gpib_sendbuffer(int fd, int addr, char *data, int count)

FONCTION: gpib_sendbuffer - send buffer to the addressed device

PARAMETER: (int)fd - file descriptor obtained from an open
(int)addr - device address
(char *)data - buffer to send to the device
(int)count - # of bytes to write

RETURN: 0 - without any error
-1 - function failed, errno is set

int gpib_defeoi(int fd, int c)

FONCTION: gpib_defeoi - define new EOI character

PARAMETER: (int)fd - file descriptor obtained from an open
(int)c - EOI character

RETURN: 0 - without any error
-1 - function failed, errno is set

int gpib_sendcmd(int fd, int cmd)

FONCTION: send a command, user has to include device address if necessary.

PARAMETER: (int)fd - file descriptor obtained from an open
(int)cmd - command (IEC_UNLISTEN, IEC_LISTEN, IEC_SDC ...)

RETURN: 0 - without any error
-1 - function failed, errno is set

ANNEXE D6

Multimètre HP 34401A

Guide de l'interface IEEE 488 (Extraits)

Résumé des commandes :

Cette section résume les commandes SCPI (Standard Commands for Programmable Instruments: Commandes standards pour les appareils programmables) permettant la programmation du multimètre. Pour plus de détails sur les diverses commandes, reportez-vous aux sections suivantes de ce chapitre.

Tout au long de ce manuel, les commandes SCPI sont présentées avec la syntaxe suivante : les crochets ([]) indiquent un mot-clé ou un paramètre optionnel, les accolades ({}) délimitent les paramètres d'une chaîne de commande et les symboles de comparaison (<>) indiquent un paramètre qui doit être remplacé par une valeur.

Commandes de configuration de mesure

MEASure

```
:VOLTage:DC? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:VOLTage:DC:RATio? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:VOLTage:AC? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:CURRent:DC? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:CURRent:AC? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:RESistance? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:FRESistance? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:FREQuency? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:PERiod? {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:CONTinuity?
:DIODE?
```

CONFigure

```
:VOLTage:DC: {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:VOLTage:DC:RATio {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:VOLTage:AC {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:CURRent:DC {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:CURRent:AC {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:RESistance {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:FRESistance {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:FREQuency {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:PERiod {<gamme> |MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}
:CONTinuity
:DIODE
```

CONFigure?

```
[SENSe:]
FUNCTION "VOLTage:DC"
FUNCTION "VOLTage:DC:RATio"
FUNCTION "VOLTage:AC"
FUNCTION "CURRent:DC"
FUNCTION "CURRent:AC"
FUNCTION "RESistance"(ohms 2 fils)
FUNCTION "FRESistance"(ohms . 4 fils)
FUNCTION "FREQuency"
FUNCTION "PERiod"
FUNCTION "CONTinuity"
FUNCTION "DIODE"
FUNCTION?
```

[SENSe:]

```
VOLTage:DC:RANGE {<gamme> |MIN|MAX}
VOLTage:DC:RANGE? {MIN|MAX}
VOLTage:AC:RANGE {<gamme> |MIN|MAX}
VOLTage:AC:RANGE? {MIN|MAX}
CURRent:DC:RANGE {<gamme> |MIN|MAX}
CURRent:DC:RANGE? {MIN|MAX}
CURRent:AC:RANGE {<gamme> |MIN|MAX}
CURRent:AC:RANGE? {MIN|MAX}
RESistance:RANGE {<gamme> |MIN|MAX}
RESistance:RANGE? {MIN|MAX}
```

```
FRESistance:RANGE {<gamme> |MIN|MAX}
FRESistance:RANGE? {MIN|MAX}
FREQuency:VOLTage:RANGE {<gamme> |MIN|MAX}
FREQuency.VDLTage:RANGE? {MIN|MAX}
PERiod:VOLTage:RANGE {<gamme> |MIN|MAX}
PERiod:VOLTage:RANGE? {MIN|MAX}
```

[SENSe:]

```
VOLTage:DC:RANGE:AUTO {OFF|ON}
VOLTage:DC:RANGE:AUTO?
VOLTage:AC:RANGE:AUTO {OFF|ON}
VOLTage:AC:RANGE:AUTO?
CURRent:DC:RANGE:AUTO {OFF|ON}
CURRent:DC:RANGE:AUTO?
CURRent:AC:RANGE:AUTO {OFF|ON}
CURRent:AC:RANGE:AUTO?
RESistance:RANGE:AUTO {OFF|ON}
RESistance:RANGE:AUTO?
FRESistance:RANGE:AUTO {OFF|ON}
FRESistance:RANGE:AUTO?
FREQuency:VOLTage:RANGE:AUTO {OFF|ON}
FREQuency:VOLTage:RANGE:AUTO?
PERiod:VOLTage:RANGE:AUTO {OFF|ON}
PERiod:VOLTage:RANGE:AUTO?
```

[SENSe:]

```
VOLTage:DC:RESolution {<résolution>|MIN|MAX}
```

VOLTage:DC:RESolution? {MIN|MAX}
 VOLTage:AC:RESolution {<résolution>|MIN|MAX}
 VOLTage:AC:RESolution? {MIN|MAX}
 CURRent:DC:RESolution {<résolution>|MIN|MAX}
 CURRent:DC:RESolution? {MIN|MAX}
 CURRent:AC:RESolution {<résolution>|MIN|MAX}
 CURRent:AC:RESolution? {MIN|MAX}
 RESistance:RESolution {<résolution>|MIN|MAX}
 RESistance:RESolution? {MIN|MAX}
 FRESistance:RESolution {<résolution>|MIN|MAX}
 FRESistance:RESolution? {MIN|MAX}

[SENSe:]

VOLTage:DC:NPLCycles {0.02|0.2|1|10|100|MIN|MAX}
 VOLTage:DC:NPLCycles? {MIN|MAX}
 CURRent:DC:NPLCycles {0.02|0.2|1|10|100|MIN|MAX}
 CURRent:DC:NPLCycles? {MIN|MAX}
 RESistance:NPLCycles {0.02|0.2|1|10|100|MIN|MAX}
 RESistance:NPLCycles? {MIN|MAX}
 FRESistance:NPLCycles {0.02|0.2|1|10|100|MIN|MAX}
 FRESistance:NPLCycles? {MIN|MAX}

[SENSe:]

FREQuency:APERTure {0.01|0.1|1|MIN|MAX}
 FREQuency:APERTure? {MIN|MAX}
 PERiod:APERTure {0.01|0.1|1|MIN|MAX}
 PERiod:APERTure? {MIN|MAX}

[SENSe:]

DETEctor:BANDwidth {3|20|200|MIN|MAX}
 DETECTOR:BANDwidth? {MIN|MAX}

[SENSe:]

ZERO:AUTO {OFF|ONCE|ON}
 ZERO:AUTO?

INPUT

:IMPedance:AUTO {OFF|ON}
 :IEPedance:AUTO?

ROUTE :TERMinals?

Commandes d'opérations mathématiques

CALCulate

:FUNction {NULL|DB|DBM|AVERAge|LIMit}
 :Fonction?
 :STATe {OFF|ON}
 :STATe?

CALCulate

:AVERAge:MINimum?
 :AVERAge:MAXimum?
 :AVERAge:AVERAge?
 :AVERAge:COUNT?

CALCulate

:NULL:OFFSet {<valeur>|MIN|MAX}
 :NULL:OFFSet? {MIN|MAX}

CALCulate

:DB:REFerence {<valeur>|MIN|MAX}
 :DB:REFerence? {MIN|MAX}

CALCulate

:DBM:REFerence {<valeur>|MIN|MAX}
 :DBM:REFerence? {MIN|MAX}

CALCulate

:LIMit:LOWer {<valeur>|MIN|MAX}
 :LIMit:LOWer? {MIN|MAX}

:LIMit:UPPer {<valeur>|MIN|MAX}
 :LIMit:UPPer? {MIN|MAX}

Commandes de déclenchement

INITiate

READ?

TRIGger

:SOURce {BUS|IMMediate|EXTErnal}
 :SOURce?

TRIGger

:DELay {<secondes>|MIN|MAX}
 :DELay? {MIN|MAX}

TRIGger

:DELay:AUTO {OFF|ON}
 :DELay:AUTO?

SAMple

:COUNT {<valeur>|MIN|MAX}
 :COUNT? {MIN|MAX}

TRIGger

:COUNT {<valeur>|MIN|MAX|INFinite}
 :COUNT? {MIN|MAX}

Commandes système

FETCh?

READ?

DISPlay {OFF|ON}

DISPlay?

DISPlay

:TEXT <chaîne délimitée>
 :TEXT?
 :TEXT:CLEAr

SYSTem

:BEEPer
 :BEEPer:STATe {OFF|ON}
 :BEEPer:STATe?

SYSTem:ERRor?

SYSTem:VERSion?

DATA:POINts?

*RST

*TST ?

*IDN ?

Commandes de rapport d'état

SYSTem:ERRor?

STATus

:QUEStionable:ENABle <valeur d'activation>
 :QUEStionable:ENABle?
 :QUEStionable:EVENT?

STATus

:PRESet

*CLS

*ESE <valeur d'activation>

*ESE?

*ESR?

*OPC

*OPC?

*PSC {0|1}

*PSC?
 *SRE <valeur d'activation>
 *SRE?

Commandes d'étalonnage

CALibration?
 CALibration:COUNT?
 CALibration
 :SECure:CODE <nouveau code>
 :SECure:STATE {OFF|ON}, <code>
 :SECure:STATE?
 CALibration
 :STRing <chaîne délimitée>
 :STRing?
 CALibration
 :VALue <valeur>
 :VALue?

Commandes de l'interface RS-232

SYSTem
 :LOCAl
 :REMote
 :RWLock

Commandes communes IEEE-488.2

*CLS
 *ESE <valeur d'activation>
 *ESE?
 *ESR?
 *IDN?
 *OPC
 *OPC?
 *PSC {0|1}
 *PSC?
 *RST
 *SRE <valeur d'activation>
 *SRE?
 *STB?
 *TRG
 *TST?

Séquence de programmation simplifiée

Les sept étapes de la séquence simplifiée ci-dessous vous permettent de programmer des mesures sur le multimètre depuis l'interface distante.

1. Mettez le multimètre dans un état connu (par exemple, l'état de ré-initialisation).
2. Modifiez les paramètres du multimètre en fonction de la configuration souhaitée.
3. Définissez les conditions de déclenchement.
4. Initialisez ou armez le multimètre pour la mesure.
5. Déclenchez le multimètre pour qu'il effectue une mesure.
6. Récupérez les valeurs placées dans la mémoire tampon de sortie ou dans la mémoire interne.
7. Lisez les résultats avec votre contrôleur de bus.

La méthode la plus simple pour programmer des mesures sur le multimètre consiste à utiliser les commandes **MEASure?** et **CONFIgure**. Vous pouvez sélectionner, en une seule commande, la fonction de mesure, la gamme et la résolution. Le multimètre définit automatiquement les autres paramètres de mesure (filtre ca, réglage automatique de zéro, nombre de déclenchements, etc.) à partir des valeurs par défaut présentées ci-dessous.

Valeurs par défaut des commandes MEASure? et CONFIgure

Commande	Valeur pour MEASure? et CONFIgure
Filtre CA (DET:BAND)	20 Hz (filtre moyen)
Réglage automatique de zéro (ZERO :AUTO)	OFF si la résolution donne NPLC < 1; ON si la résolution donne NPLC ≥ 1
Résistance d'entrée (INP:IMP:AUTO)	OFF (fixée à 10 MW pour toutes les gammes de tensions cc)
Nombre de mesures par déclenchement (SAMP:COUN)	1 mesure
Nombre de déclenchements (TRIG:COUN)	1 déclenchement
Retard de déclenchement (TRIG:DEL)	Automatique
Source de déclenchement (TRIG:SOUR)	Immédiate
Fonction mathématique(sous-système CALCulate)	OFF

Utilisation de la commande MEASure?

La méthode la plus facile pour programmer des mesures sur le multimètre consiste à utiliser la commande **MEASure?**. Cependant, cette commande n'offre pas beaucoup de souplesse. Lorsque vous l'exécutez, le multimètre choisit automatiquement les meilleurs réglages possibles en fonction de la configuration demandée puis lance immédiatement la mesure. Vous ne pouvez modifier aucun paramètre (mis à part la fonction, la gamme et la résolution) avant la mesure. Les résultats sont envoyés dans la mémoire tampon de sortie.

Le fait d'envoyer une commande **MEASure?** revient au même que d'envoyer une commande **CONFigure** suivie immédiatement d'une commande **READ?**.

Utilisation de la commande **CONFigure**

La commande **CONFigure** offre un peu plus de souplesse de programmation. Lorsque vous l'exécutez, le multimètre choisit automatiquement les meilleurs réglages possibles en fonction de la configuration demandée (comme pour la commande **MEASure?**), mais il ne lance pas la mesure, ce qui vous laisse la possibilité de modifier des paramètres. Vous pouvez ainsi "affiner" la configuration à partir des conditions prédéfinies. Avec les sous-systèmes **INPut**, **SENSe**, **CALCulate** et **TRIGger**, vous disposez de nombreuses commandes de bas niveau pour programmer le multimètre (la commande **SENSe: FUnction** vous permet de modifier la fonction de mesure sans avoir recours à **MEASure?** ni à **CONFigure**).

Lancez la mesure à l'aide de la commande **INITiate** ou **READ?**.

Utilisation des paramètres gamme et résolution

Avec les commandes **MEASure?** et **CONFigure**, vous pouvez sélectionner en une seule commande la fonction de mesure, la gamme et la résolution. Le paramètre gamme sert à indiquer la valeur prévue pour le signal d'entrée. Le multimètre s'en sert pour sélectionner la gamme de mesure adéquate.

Pour les mesures de fréquence et de période, le multimètre utilise la même "gamme" pour toutes les entrées comprises entre 3 Hz et 300 kHz. Le paramètre gamme ne sert qu'à indiquer la résolution. Pour cette raison, il n'est pas nécessaire d'envoyer une nouvelle commande pour chaque nouvelle fréquence à mesurer.

Le paramètre résolution sert à indiquer la résolution voulue pour la mesure. Il doit être spécifié dans la même unité que la fonction de mesure, et non en nombre de chiffres. Par exemple, pour une mesure de tension cc, indiquez la résolution en volts ; pour une mesure de fréquence, indiquez la résolution en hertz.

Pour pouvoir utiliser le paramètre résolution, vous devez spécifier une gamme.

Utilisation de la commande **READ?**

La commande **READ?** fait passer le système de déclenchement de l'état de "repos" à l'état d'attente de déclenchement". Après la réception d'une commande **READ?**, la mesure ne commence qu'une fois que les conditions de déclenchement spécifiées sont satisfaites. Les résultats sont envoyés immédiatement dans la mémoire tampon de sortie. Vous devez récupérer les valeurs avec votre contrôleur de bus, sinon le multimètre arrête les mesures dès que la mémoire tampon de sortie est pleine. Lorsque vous utilisez la commande **READ?**, les résultats ne sont pas stockés dans la mémoire interne du multimètre.

La commande **READ?** est équivalente à une commande **INITiate** immédiatement suivie d'une commande **FETCh?**, hormis le fait qu'elle ne met pas les mesures en mémoire interne.

Attention

Si vous envoyez deux commandes d'interrogation à la suite sans lire la réponse de la première, puis que vous essayez de lire la réponse de la seconde, vous risquez d'obtenir des données de la première réponse, suivies par l'intégralité de la seconde réponse. Pour éviter cela, lisez toujours la réponse à vos commandes d'interrogation. Si, dans une situation particulière, il ne vous est pas possible de lire la réponse, envoyez une commande de remise à zéro d'appareil avant d'envoyer la seconde commande d'interrogation.

Utilisation des commandes **INITiate** et **FETCh?**

Les commandes **INITiate** et **FETCh?** fournissent le niveau de contrôle le plus bas (et le maximum de souplesse) pour le déclenchement et la récupération des mesures. La commande **INITiate** doit être utilisée après avoir configuré le multimètre pour la mesure. Elle fait passer le système de déclenchement de l'état de "repos" à l'état d' "attente de déclenchement". Après la réception d'une commande **INITiate**, la mesure ne commence qu'une fois que les conditions de déclenchement spécifiées sont satisfaites. Les résultats sont placés dans la mémoire interne du multimètre (qui peut contenir jusqu'à 512 valeurs). Ils restent en mémoire jusqu'à ce que vous les récupériez.

La commande **FETCh?** permet de transférer les résultats de mesure depuis la mémoire interne du multimètre vers sa mémoire tampon de sortie, où elles peuvent être lues par votre contrôleur de bus.

Exemple de commande MEASure?

Le segment de programme suivant montre comment utiliser la commande **MEASure?** pour effectuer une mesure. Le multimètre est configuré pour une mesure de tension cc, il est mis automatiquement en "attente de déclenchement" et il se déclenche en interne ; pour finir, il envoie la mesure vers la mémoire tampon de sortie.

```
MEAS:VOLT:DC? 10,0.003  
lecture par le bus
```

Ceci est la méthode la plus simple pour effectuer une mesure. Cependant, la commande **MEASure?** n'offre aucune souplesse pour régler le nombre de déclenchements, le nombre d'échantillons, le retard de déclenchement, etc. Tous les paramètres de mesure, à l'exception de la fonction, de la gamme et de la résolution, sont pré-définis automatiquement (voir le tableau de la page 110).

Exemple de commande CONFigure

Le segment de programme ci-dessous montre comment utiliser les commandes **READ?** et **CONFigure** pour effectuer une mesure avec un déclenchement externe. Le multimètre est configuré pour une mesure de tension cc. La commande **CONFigure** ne met pas le multimètre en "attente de déclenchement" ; c'est la commande **READ?** qui s'en charge. Dès qu'il reçoit une impulsion sur la borne Ext Trig, le multimètre effectue la mesure puis envoie le résultat dans la mémoire tampon de sortie.

```
CONF:VOLT:DC 10, 0.003  
TRIG:SOUR EXT  
READ?  
lecture par la bus
```

Exemple de commande CONFigure

Le segment de programme ci-dessous est similaire au précédent, mais il utilise la commande **INITiate** pour mettre le multimètre en "attente de déclenchement". Cette commande met le multimètre en "attente de déclenchement", effectue une mesure dès que la borne Ext Trig reçoit une impulsion, puis place le résultat dans la mémoire interne du multimètre. La commande **FETCh?** transfère ensuite la mesure depuis la mémoire interne vers la mémoire tampon de sortie.

```
CONF:VOLT:DC 10, 0.003  
TRIG.SOUR EXT  
INIT  
FETC?  
lecture par la bus
```

Il est plus rapide de stocker les mesures en mémoire interne avec la commande **INITiate** que de les envoyer vers la mémoire tampon de sortie avec la commande **READ?**. Le multimètre peut stocker jusqu'à 512 mesures dans sa mémoire interne. Si vous lui demandez d'effectuer plus de 512 mesures (en réglant le nombre d'échantillons et le nombre de déclenchements), puis que vous envoyez une commande **INITiate**, cela génère une erreur de mémoire.

Après l'exécution d'une commande **INITiate**, aucune autre commande n'est acceptée tant que la séquence de mesure n'est pas terminée. Cependant, si vous sélectionnez **TRIGGER: SOURCE BUS** comme source de déclenchement, le multimètre accepte tout de même les commandes ***TRG** (déclenchement par le bus) et les messages IEEE-488 de déclenchement groupé (Group Execute Trigger).

Les commandes MEASure? et CONFigure

Voir aussi la section "Configuration de mesure" du chapitre 3 à la page 51.

- Pour le paramètre gamme, MIN sélectionne la plus petite gamme possible pour la fonction choisie ; MAX sélectionne la plus grande gamme possible; DEF correspond à la sélection automatique de gamme.
- Le paramètre résolution doit être spécifié dans la même unité que la fonction de mesure, et non en nombre de chiffres. MIN sélectionne la plus petite valeur acceptée, ce qui donne la meilleure résolution; MAX sélectionne la plus grande valeur acceptée, soit la moins bonne résolution ; DEF correspond à la résolution par défaut, qui est le mode lent à 51/12 chiffres (10 PLC).

Pour pouvoir utiliser le paramètre résolution, vous devez spécifier une gamme.

MEASure:VOLTage:DC? {<gamme> |MIN|MAX|DEF}, {<résolution> |MIN |MAX|DEF}

Prépare et effectue une mesure de tension ce avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie.

MEASure:VOLTage:DC:Ratio? {<gamme>|MIN|MAX|DEF}, {<résolution> |MIN |MAX|DEF}

Prépare et effectue une mesure de rapport cc:cc avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les mesures de rapport, la gamme spécifiée s'applique au signal relié aux bornes **Input**. La gamme de la tension de référence appliquée aux bornes **Sense** est sélectionnée automatiquement.

MEASure:VOLTage:AC? {<gamme>|MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF}

Prépare et effectue une mesure de tension ca avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les mesures en alternatif, la résolution est en fait fixée à 6 ½ chiffres. Le paramètre résolution n'affecte que l'affichage en face avant.

MEASure:CURRent:DC? {<gamme>|MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF}

Prépare et effectue une mesure de courant cc avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie.

MEASure:CURRent:AC{<gamme>|MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF}

Prépare et effectue une mesure de courant ca avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les mesures en alternatif, la résolution est en fait fixée à 6 ½ chiffres. Le paramètre résolution n'affecte que l'affichage en face avant.

MEASure:RESistance? {<gamme>|MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}

Prépare et effectue une mesure de résistance en 2 fils avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie.

MEASure:FRESistance? ? {<gamme>|MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}

Prépare et effectue une mesure de résistance en 4 fils avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie.

MEASure:FREQuency? {<gamme>|MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}

Prépare et effectue une mesure de fréquence avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les mesures de fréquence, le multimètre n'utilise qu'une seule "gamme" pour toutes les entrées comprises entre 3 Hz et 300 kHz. Lorsqu'aucun signal d'entrée n'est appliqué, la mesure de fréquence donne "0".

MEASure:PERiod? {<gamme>|MIN|MAX|DEF}, {<résolution>|MIN|MAX|DEF}

Prépare et effectue une mesure de période avec la gamme et la résolution spécifiées. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les mesures de période, le multimètre n'utilise qu'une seule "gamme" pour toutes les entrées comprises entre 3,3 ms et 0,33 seconde. Lorsqu'aucun signal d'entrée n'est appliqué, la mesure de période donne "0".

MEASure:CONTInuity?

Prépare et effectue une mesure de continuité. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les tests de continuité, la gamme et la résolution sont fixes (1 k Ω et 4 ½ chiffres).

MEASure:DIODE?

Prépare et effectue une mesure de diode. Le résultat est envoyé dans la mémoire tampon de sortie. Pour les tests de diode, la gamme et la résolution sont fixes (1 Vcc avec une source de courant de sortie de 1 mA et 4 ½ chiffres).

CONFigure:VOLTage:DC {<résolution>|MIN|MAX|DEF}

Prépare et configure le multimètre pour des mesures de tension ce avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure.

CONFigure:VOLTage:DC:

Prépare et configure le multimètre pour des mesures de rapport cc:cc avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure. Pour les mesures de rapport, la gamme spécifiée s'applique au signal relié aux bornes **Input**. La gamme de la tension de référence appliquée aux bornes **Sense** est sélectionnée automatiquement.

CONFigure:VOLTage:AC {<gamme> |MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF}

Prépare et configure le multimètre pour des mesures de tension ca avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure. Pour les mesures en alternatif, la résolution est en fait fixée à 6 1/2 chiffres. Le paramètre résolution n'affecte que l'affichage en face avant.

CONFigure:CURRent:DC {<gamme> |MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF}

Prépare et configure le multimètre pour des mesures de courant cc avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure.

CONFigure:CURRent:AC {<gamme> |MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF}

Prépare et configure le multimètre pour des mesures de courant ca avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure. Pour les mesures en alternatif, la résolution est en fait fixée à 6 1/2 chiffres. Le paramètre résolution n'affecte que l'affichage en face avant.

CONFigure:RESistance {<gamme> |MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF},

Prépare et configure le multimètre pour des mesures de résistance en 2 fils avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure.

CONFigure:FRESistance {<gamme> |MIN|MAX|DEF},

Prépare et configure le multimètre pour des mesures de résistance en 4 fils avec la gamme et la résolution spécifiées. Cette commande ne lance pas de mesure.

CONFigure:FREQuency {<gamme> |MIN|MAX|DEF},

Prépare et configure le multimètre pour une mesure de fréquence avec la gamme et la résolution spécifiées. Cette commande ne lance pas la mesure. Pour les mesures de fréquence, le multimètre utilise une même "gamme" pour tous les signaux dont la fréquence est comprise entre 3 Hz et 300 kHz. Lorsqu'aucun signal n'est appliqué en entrée, la mesure de fréquence retourne la valeur "0".

CONFigure:PERiod {<gamme> |MIN|MAX|DEF}, {<résolution> |MIN|MAX|DEF},

Prépare et configure le multimètre pour une mesure de période avec la gamme et la résolution spécifiées. Cette commande ne lance pas la mesure. Pour les mesures de période, le multimètre utilise une même "gamme" pour tous les signaux dont la période est comprise entre 0,33 seconde et 3,3 microsecondes. Lorsqu'aucun signal n'est appliqué en entrée, la mesure de période retourne la valeur "0".

CONFigure:CONTInuity

Prépare et configure le multimètre pour des mesures de continuité. Cette commande ne lance pas de mesure. Pour les tests de continuité, la gamme et la résolution sont fixes (1 k Ω et 4 1/2 chiffres).

CONFigure:DIODE

Prépare et configure le multimètre pour des mesures de diode. Cette commande ne lance pas de mesure. Pour les tests de diode, la gamme et la résolution sont fixes (1 Vcc avec une source de courant de sortie de 1 mA et 4 1/2 chiffres).

CONFigure?

Interroge le multimètre pour connaître sa configuration courante et retourne une chaîne délimitée.

Commandes de configuration de mesure

FUNCTION "<fonction>"

Sélectionne une fonction de mesure. Dans la chaîne de commande, la fonction doit être encadrée de délimiteurs comme dans (FUNC "VOLT: DC"). Indiquez l'une des chaînes ci-dessous.

VOLTage:DC	CURRent:AC	PERiod
VOLTage:DC:RATio	RESistance (ohms 2 fils)	CONTInuity
VOLTage:AC	FRESistance (ohms 4 fils)	DIODE
CURRent:DC	FREQuency	

FUNCTION?

Demande la fonction de mesure et retourne une chaîne délimitée.

<fonction>:RANGE {<gamme>|MIN|MAX}

Sélectionne une gamme pour la fonction choisie. Pour les mesures de fréquence et de période, la sélection de gamme s'applique à la tension d'entrée du signal, et non à sa fréquence (utilisez **FREQUENCY:VOLTage** ou **PERiod:VOLTage**). MIN sélectionne la plus petite gamme possible pour la fonction choisie. MAX sélectionne la gamme la plus élevée. [mémoire volatile]

<fonction>:RANGE? [MIN|MAX]

Demande la gamme sélectionnée pour la fonction choisie.

<fonction>:RANGE:AUTO {OFF|ON}

Désactive ou active la sélection automatique de gamme pour la fonction choisie. Pour les mesures de fréquence et de période, utilisez **FREQUENCY:VOLTage** ou **PERiod:VOLTage**. Seuils de transition pour la sélection automatique de gamme: seuil inférieur à 10% de la gamme ; seuil supérieur à 120% de la gamme. [mémoire volatile]

<fonction>:RANGE: AUTO?

Demande l'état de la sélection automatique de gamme. Retourne "0" (OFF) ou "1" (ON).

<fonction>:RESolution {<résolution>|MIN|MAX}

Sélectionne la résolution pour la fonction spécifiée (ne convient pas pour les mesures de fréquence, de période ou de rapport). Spécifiez la résolution dans la même unité que la fonction de mesure, et non en nombre de chiffres. MIN sélectionne la plus petite valeur acceptée, qui donne la meilleure résolution. MAX sélectionne la plus grande valeur acceptée, soit la résolution la plus faible. [mémoire volatile]

<fonction>:RESolution? [MIN|MAX]

Demande la résolution définie pour la fonction choisie. Pour les mesures de fréquence ou de période, le multimètre retourne une valeur de résolution correspondant à une fréquence «entrée de 3 Hz.

<fonction>:NPLCycles {0.02|0.2|1|10|100|MIN|MAX}

Sélectionne le temps d'intégration en nombre de cycles de la tension secteur pour la fonction courante (la valeur par défaut est de 10 PLC). Cette commande ne convient que pour les mesures de tension et de courant cc, de rapport, et de résistance à 2 et 4 fils. MIN = 0,02. MAX 100. [mémoire volatile]

<fonction>:NPLCycles? [MIN|MAX]

Demande le temps d'intégration défini pour la fonction courante.

FREQUENCY:APERTure {0.01|0.1|1|MIN|MAX}

Sélectionne le temps d'ouverture (ou temps de porte) pour les mesures de fréquence (la valeur par défaut est de 0,1 seconde). Indiquez une valeur de 10 ms (4½ chiffres), de 100 ms (valeur par défaut; 5½ chiffres) ou de 1 seconde (6½ chiffres). MIN = 0,01 seconde. MAX = 1 seconde. [mémoire volatile]

FREQUENCY:APERTure? [MIN|MAX]

Demande le temps d'ouverture pour les mesures de fréquence.

PERiod:APERTure {0.01|0.1|1|MIN|MAX}

Sélectionne le temps d'ouverture (ou temps de porte) pour les mesures de période (la valeur par défaut est de 0,1 seconde). Indiquez une valeur de 10 ms (4½ chiffres), de 100 ms (valeur par défaut; 5½ chiffres) ou de 1 seconde (6½ chiffres). MIN = 0,01 seconde. MAX = 1 seconde. [mémoire volatile]

PERiod:APERTure? [MIN|MAX]

Demande le temps d'ouverture pour les mesures de période.

[SENSe:]DETECTOR:BANDwidth {3|20|200|MIN|MAX}

Indique la plus basse fréquence prévue dans le signal d'entrée. Suivant la fréquence que vous indiquez, le multimètre sélectionne le filtre ca lent, moyen (par défaut) ou rapide. MIN = 3 Hz. MAX = 200 Hz. [mémoire volatile]

[SENSe:]DETECTOR:BANDwidth? [MIN|MAX]

Interroge le multimètre pour savoir quel filtre ca il utilise. Retourne "3", "20" ou "200".

[SENSe:]ZERO:AUTO {OFF|ONCE|ON}

Désactive ou active (par défaut) le réglage automatique de zéro. Les valeurs OFF et ONCE ont un effet similaire, mais avec le paramètre OFF, aucune nouvelle mesure de zéro n'est effectuée tant que le multimètre ne repasse pas dans l'état

d' "attente de déclenchement", tandis que le paramètre ONCE provoque une mesure de zéro immédiate. [mémoire volatile]

[SENSe:]ZERO:AUTO?

Demande l'état du réglage automatique de zéro. Retourne «0" (OFF ou ONCE) ou "1" (ON).

INPut:IMPedance:AUTO {OFF|ON}

Désactive ou active le mode de résistance d'entrée automatique pour les mesures de tension ce. Avec AUTO OFF (par défaut), la résistance d'entrée est fixée à 10 MΩ pour toutes les gammes. Avec AUTO ON, la résistance d'entrée est fixée à >10 GΩ pour les gammes 100 mV, 1 V et 10 V. [mémoire volatile]

INPut:IMPedance:AUTO?

Demande l'état du mode de résistance d'entrée automatique. Retourne "0" (OFF) ou "1" (ON).

ROUTE:TERMinals?

Interroge le multimètre pour savoir si les bornes d'entrée sélectionnées sont les bornes avant ou arrière. Retourne "FRON" ou "REAR".

Commandes d'opérations mathématiques

Il existe cinq fonctions mathématiques disponibles, mais vous ne pouvez en activer qu'une seule à la fois. Chacune d'elles effectue une opération mathématique sur chaque mesure ou stocke des données concernant une série de mesures. L'opération sélectionnée reste active tant que vous ne la désactivez pas, que vous ne changez pas de fonction, que vous ne mettez pas le multimètre hors tension et que vous ne lancez pas de ré-initialisation à distance de l'interface. Les opérations mathématiques utilisent un ou plusieurs registres internes. La valeur de certains de ces -registres peut être prédéfinie ; les autres contiennent les résultats de l'opération mathématique.

Le tableau ci-dessous vous indique les combinaisons de fonctions de mesure et d'opérations mathématiques qui sont autorisées. Les "X" indiquent les combinaisons valides. Si vous choisissez une opération mathématique qui n'est pas autorisée pour la fonction de mesure en cours, le mode mathématique est alors désactivé. Si vous sélectionnez une opération mathématique valide puis que vous passez sur une opération non valide, l'interface distante génère l'erreur "Settings conflict" (conflit de paramètres). Pour les mesures relatives et les mesures en dB, vous devez activer l'opération mathématique avant d'écrire dans le registre correspondant.

	DC V	AC V	DC I	AC I	Ω 2W	Ω 4W	Freq	Per	Cont	Diode	Ratio
Null	X	X	X	X	X	X	X	X			
Min-Max	X	X	X	X	X	X	X	X			X
dB	X	X									
dBm	X	X									
Limit	X	X	X	X	X	X	X	X			X

CALCulate:FUNCTion {NULL|DB|DBM|AVERAge|LIMit}

Sélectionne la fonction mathématique. Vous ne pouvez activer qu'une seule fonction à la fois. La fonction par défaut est la fonction de mesure relative. [mémoire volatile]

CALCulate:FUNCTion?

Demande quelle est la fonction mathématique en cours. Retourne NULL, DB, DBM, AVER ou LIM.

CALCulate:STATe {OFF|ON}

Désactive ou active la fonction mathématique sélectionnée. [mémoire volatile]

CALCulate:STATe?

Demande l'état de la fonction mathématique. Retourne "0" (OFF) ou "1" (ON).

CALCulate.AVERAge:MINimum?

Lit la valeur minimale trouvée pendant une opération min-max. Cette valeur est remise à zéro lorsque la fonction min-max est activée, après une mise hors tension ou après une ré-initialisation à distance de l'interface. [mémoire volatile]

CALCulate.AVERAge:MAXimum?

Lit la valeur maximale trouvée pendant une opération min-max. Cette valeur est remise à zéro lorsque la fonction min-max est activée, après une mise hors tension ou après une ré-initialisation à distance de l'interface. [mémoire volatile]

CALCulate:AVERage:AVERage?

Lit la moyenne de toutes les mesures effectuées depuis l'activation de la fonction min-max. Cette valeur est remise à zéro lorsque la fonction min-max est activée, après une mise hors tension ou après une ré-initialisation à distance de l'interface. [mémoire volatile]

CALCulate:AVERage:COUnT?

Lit le nombre de mesures effectuées depuis l'activation de la fonction min-max. Cette valeur est remise à zéro lorsque la fonction min-max est activée, après une mise hors tension ou après une ré-initialisation à distance de l'interface. [mémoire volatile]

CALCulate:NULL:OFFSet {<valeur>|MIN|MAX}

Stocke une valeur de référence dans le registre de référence du multimètre. Vous devez activer l'opération mathématique avant de pouvoir écrire dans ce registre. La valeur de référence peut être un nombre quelconque compris entre 0 et $\pm 120\%$ de la gamme la plus élevée pour la fonction en cours. MIN = -120% de la gamme la plus élevée. MAX = 120% de la gamme la plus élevée. [mémoire volatile]

CALCulate:NULL:OFFSet? [MIN|MAX]

Demande la valeur de référence.

CALCulate:DB:REFeRence {<valeur>|MIN|MAX}

Stocke une valeur de référence dans le registre de référence en dB. Vous devez activer l'opération mathématique avant de pouvoir écrire dans ce registre. La valeur de référence en dB peut être un nombre quelconque compris entre 0 dBm et ± 200 dBm. MIN = -200.00 dBm. MAX = 200.00 dBm. [mémoire volatile]

CALCulate:DB:REFeRence? [MIN|MAX]

Demande la valeur de référence en dB.

CALCulate:DBM:REFeRence {<valeur> |MIN|MAX}

Sélectionne la résistance de référence pour les dBm. Choisissez l'une des valeurs suivantes : 50, 75, 93, 110, 124, 125, 135, 150, 250, 300, 500, 600, 800, 900, 1000, 1200 ou 8000 ohms. MIN = 50Ω . MAX = 8000Ω . [mémoire non volatile]

CALCulate:DBM:REFeRence? [MIN|MAX]

Demande la résistance de référence pour les dBm.

CALCulate:LIMit:LOWer {<valeur>|MIN|MAX}

Définit la limite inférieure pour le test de limites. Cette limite peut être un nombre quelconque compris entre 0 et $\pm 120\%$ de la gamme la plus élevée pour la fonction en cours. MIN = -120% de la gamme la plus élevée. MAX = 120% de la gamme la plus élevée. [mémoire volatile]

CALCulate:LIMit:LOWer? [MIN|MAX]

Demande la limite inférieure.

CALCulate:LIMit:UPPer {<valeur> |MIN|MAX}

Définit la limite supérieure pour le test de limites. Cette limite peut être un nombre quelconque compris entre 0 et $\pm 120\%$ de la gamme la plus élevée pour la fonction en cours. MIN = -120% de la gamme la plus élevée. MAX = 120% de la gamme la plus élevée. [mémoire volatile]

CALCulate:LIMit:UPPer? [MIN|MAX]

Demande la limite supérieure.

DATA:FEED RDG_STORE, {"CALCulate"|" "}

Indique si les mesures effectuées par la commande INITiate doivent être stockées dans la mémoire interne du multimètre (par défaut) ou si elles ne doivent pas être stockées du tout. Cette commande est disponible à partir de la Révision 2 du micrologiciel (REV 02-01-01).

Dans l'état par défaut (DATA: FEED RDG_STORE, "CALC"), le multimètre stocke en mémoire, à l'exécution de la commande INITiate, jusqu'à 512 mesures. Les commandes MEASure? et CONFigure sélectionnent automatiquement "CALC". Lorsque la mémorisation est désactivée (DATA: FEED RDG_STORE, " "), les mesures effectuées à l'aide de la

commande `INITiate` ne sont pas stockées. Ceci peut être utile avec la fonction min-max car vous pouvez ainsi déterminer la moyenne d'une série de mesures sans avoir à stocker les valeurs individuelles. Si vous essayez de transférer les résultats de mesure vers la mémoire tampon de sortie à l'aide de la commande `FETCH?`, cela génère une erreur.

DATA: FEED?

Demande l'état de la mémoire de mesure. Retourne "CALC" ou "".

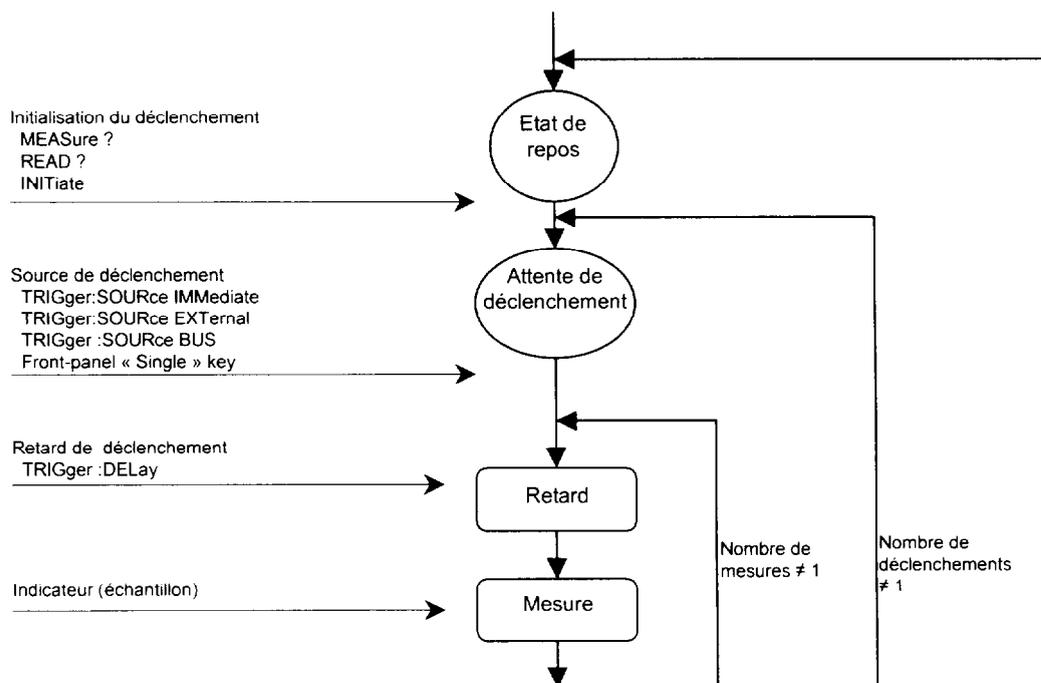
Déclenchement

Le système de déclenchement du multimètre vous permet de générer les déclenchements manuellement ou automatiquement, d'effectuer plusieurs mesures par déclenchement et d'introduire un retard avant chaque mesure. Normalement, pour chaque déclenchement, le multimètre n'effectue qu'une seule mesure ; cependant, vous pouvez lui demander d'en effectuer jusqu'à 50000.

Le déclenchement du multimètre depuis l'interface distante est un processus à plusieurs étapes qui offre une grande souplesse.

- Tout d'abord, vous devez configurer le multimètre pour la mesure à effectuer en sélectionnant la fonction, la gamme, la résolution, etc.
- Ensuite, vous devez spécifier la source de déclenchement. Celle-ci peut être logicielle (déclenchement par le bus depuis l'interface distante), matérielle (par la borne de déclenchement externe Ext Trig du panneau arrière) ou immédiate (déclenchement interne).
- Ensuite, vous devez vous assurer que le multimètre est prêt à être déclenché par la source spécifiée (il doit être dans l'état appelé attente de déclenchement).

Le diagramme suivant décrit le système de déclenchement du multimètre.



Le déclenchement du multimètre est une opération à plusieurs étapes

L'état d'attente de déclenchement

Après avoir configuré le multimètre et sélectionné une source de déclenchement, vous devez le mettre en attente de déclenchement. Aucun déclenchement n'est accepté tant que le multimètre n'est pas dans cet état. Si le multimètre est en "attente de déclenchement" et qu'un signal de déclenchement se présente, le processus de mesure démarre et les mesures sont effectuées.

Le terme «attente de déclenchement» est principalement utilisé pour le fonctionnement à distance. Lorsque vous travaillez à partir de la face avant, le multimètre est toujours en "attente de déclenchement" et accepte de se déclencher à tout moment, sauf si une mesure est déjà en cours.

Depuis l'interface distante, vous pouvez mettre le multimètre en "attente de déclenchement" en exécutant l'une des commandes suivantes:

```
MEASure ?  
READ?  
INITiate
```

Lorsqu'il reçoit une commande lui demandant de passer en "attente de déclenchement", le multimètre met environ 20 ms à se préparer. Les déclenchements qui surviennent au cours de cette période sont ignorés.

Commandes de déclenchement

INITiate

Fait passer le système de déclenchement de l'état de "repos" à l'état d'attente de déclenchement". Après réception de la commande **INITiate**, les mesures commencent dès que les conditions de déclenchement spécifiées sont satisfaites. Les résultats sont placés dans la mémoire interne du multimètre (qui peut contenir jusqu'à 512 valeurs). Ils restent en mémoire jusqu'à ce que vous les récupériez (pour cela, utilisez la commande **FETCH?**).

A partir de la Révision 2, le micrologiciel comprend une nouvelle commande qui vous permet d'effectuer des mesures avec **INITiate** sans avoir à les stocker en mémoire interne. Cette commande peut être utile avec la fonction min-max car elle vous permet de déterminer la moyenne d'une série de mesures sans avoir à stocker les valeurs individuelles.

```
DATA:FEED RDG_STORE, " "           ne stocke pas les mesures  
DATA:FEED RDG_STORE, "CALCulate"   stocke les mesures (par défaut)
```

READ?

Fait passer le système de déclenchement de l'état de "repos" à l'état d' "attente de déclenchement". Après réception de la commande **READ?**, les mesures commencent dès que les conditions de déclenchement spécifiées sont satisfaites. Les résultats sont immédiatement envoyés vers la mémoire tampon de sortie.

TRIGger:SOURce {BUS | IMMEDIATE | EXTERNAL}

Sélectionne la source de déclenchement du multimètre. Celui-ci peut accepter un déclenchement logiciel (par le bus), un déclenchement interne immédiat (source par défaut) ou un déclenchement matériel par la borne Ext Trig (déclenchement externe) du panneau arrière. [mémoire volatile]

TRIGger:SOURce?

Demande la source de déclenchement actuellement sélectionnée. Retourne "BUS", "IMM" ou "EXT".

TRIGger:DELay {<secondes> | MIN | MAX}

Introduit un retard de déclenchement après le signal de déclenchement et entre les mesures successives. Si vous n'indiquez pas de retard de déclenchement, le multimètre en choisit automatiquement un pour vous. Choisissez une valeur comprise entre 0 et 3600 secondes. MIN = 0 seconde. MAX = 3600 secondes. [mémoire volatile]

TRIGger:DELay? [MIN | MAX]

Demande le retard de déclenchement actuellement défini.

TRIGger:DELay:AUTO {OFF | ON}

Désactive ou active le retard automatique de déclenchement. La durée de ce retard dépend de la fonction, de la gamme, du temps d'intégration et du filtre ca. Le fait de définir une valeur spécifique pour le retard de déclenchement désactive la fonction de retard automatique de déclenchement. [mémoire volatile] 4

TRIGger:DELay:AUTO?

Demande l'état du retard automatique de déclenchement. Retourne "0" (OFF) ou "1" (ON).

SAMPlE:COUNT {<valeur> | MIN | MAX}

Définit le nombre de mesures (échantillons) à effectuer à chaque déclenchement. Choisissez un nombre compris entre 1 et 50000 mesures par déclenchement. MIN = 1. MAX = 50000. [mémoire volatile]

SAMPlE:COUNT? [MIN | MAX]

Demande le nombre d'échantillons qui a été défini.

TRIGger:COUNT {<valeur>|MIN|MAX|INFinite}

Définit le nombre de déclenchements que le multimètre acceptera avant de repasser à l'état de "repos". Choisissez un nombre compris entre 1 et 50 000 déclenchements. Le paramètre INFinite vous permet de déclencher le multimètre en continu (vous devez alors envoyer une commande de remise à zéro d'appareil si vous souhaitez remettre le multimètre à l'état de "repos"). En mode local, le nombre de déclenchements est ignoré. MIN = 1. MAX = 50000. [mémoire volatile]

TRIGger:COUNT? [MIN|MAX]

Demande le nombre de déclenchements qui a été défini. Si vous avez défini un nombre de déclenchements infini, cette interrogation retourne la valeur "9.90000000E+37".

Commandes système

FETCh?

Transfère les résultats stockés dans la mémoire interne du multimètre (par la commande INITiate) vers la mémoire tampon de sortie, où ils peuvent être lus par votre contrôleur de bus.

READ?

Fait passer le système de déclenchement de l'état de "repos" à l'état d'attente de déclenchement". Après réception de la commande READ?, les mesures commencent dès que les conditions de déclenchement spécifiées sont satisfaites. Les résultats sont envoyés immédiatement dans la mémoire tampon de sortie.

DISPlay {OFF|ON}

Désactive ou active l'afficheur de la face avant. [mémoire volatile]

DISPlay?

Demande l'état de l'afficheur de la face avant. Retourne "0" (OFF) ou "1" (ON).

DISPlay:TEXT <chaîne délimitée>

Affiche un message sur la face avant Le multimètre peut afficher les 12 premiers caractères du message; les caractères restants n'apparaissent pas. [mémoire volatile]

DISPlay:TEXT?

Demande le message qui a été envoyé sur l'afficheur de la face avant et retourne une chaîne délimitée.

DISPlay:TEXT:CLEAr

Efface le message affiché en face avant

SYSTem:BEEPer

Emet immédiatement un signal sonore simple.

SYSTem:BEEPer:STATE {OFF|ON}

Désactive ou active l'avertisseur sonore de la face avant. [mémoire non volatile]

Lorsque l'avertisseur sonore est désactivé, le multimètre n'émet plus de signal sonore lorsque:

- 1) la fonction min-max détecte un nouveau minimum ou un nouveau maximum ;
- 2) la fonction de maintien de résultat détecte une mesure stable ;
- 3) une mesure sort des limites lors d'un test de limites ;
- 4) la fonction de test de diode mesure une diode en polarisation directe.

SYSTem:BEEPer:STATE?

Demande l'état de l'avertisseur sonore de la face avant. Retourne "0" (OFF) ou "1" (ON).

SYSTem:ERRor?

Récupère les erreurs de la file d'erreurs du multimètre. Celle-ci peut contenir jusqu'à 20 erreurs. Les erreurs sont récupérées dans l'ordre FIFO (first-in-first-out: première entrée, première sortie). Chaque chaîne d'erreur peut contenir jusqu'à 80 caractères.

SYSTEM:VERSION?

Interroge le multimètre pour connaître la version SCPI utilisée.

DATA:POINTS?

Demande le nombre de mesures stockées dans la mémoire interne du multimètre.

***RST**

Ré-initialise le multimètre à sa configuration de mise sous tension.

***TST?**

Effectue un auto-test complet du multimètre. Retourne "0" si l'auto-test réussit et "1" s'il échoue.

***IDN?**

Lit la chaîne d'identification du multimètre (prévoyez une variable chaîne d'au moins 35 caractères).

ANNEXE D7

Programmation Réseau Syntaxe de quelques appels systèmes

Nota : Les informations qui suivent sont tirées de l'aide en ligne d'un système Linux. La conformité de ces fonctions à BSD 4.4 garantit une certaine portabilité de ces appels. Le système qui nous intéresse respecte intégralement ces appels...

IP(4)

Manuel du programmeur Linux

IP(4)

NAME

ip - Linux IPv4 protocol implementation

SYNOPSIS

```
#include <sys/socket.h>
#include <net/netinet.h>

tcp_socket = socket(PF_INET, SOCK_STREAM, 0);
raw_socket = socket(PF_INET, SOCK_RAW, protocol);
udp_socket = socket(PF_INET, SOCK_DGRAM, protocol);
```

DESCRIPTION

Linux implements the IPv4 protocol described in RFC791 and RFC1122. ip contains a level 2 multicasting implementation conforming to RFC1122. It also contains an IP router including a packet filter.

The protocol is implemented in the kernel on the basis of a BSD compatible socket interface. For more information on sockets, see socket(4).

An IP socket is created by calling the socket(2) function with a PF_INET socket family argument. Valid socket types are SOCK_STREAM to open a tcp(4) socket, SOCK_DGRAM to open a udp(4) socket, or SOCK_RAW to open a raw socket. protocol is the IP protocol in the IP header be received or sent. For TCP and UDP sockets, only 0, IPPROTO_TCP, or IPPROTO_UDP are valid. For SOCK_RAW you may specify a valid IANA IP protocol defined in RFC1700 assigned numbers.

Raw sockets may only be opened by a process with effective user id 0 or when the process has the CAP_NET_RAW capability.

When a process wants to receive new incoming packets or connections, it should be bound to a local interface address using bind(2). When INADDR_ANY is specified it will bind to any local interface. A bound TCP socket is unavailable for time after closing, unless the SO_REUSEADDR flag is set.

ADDRESS FORMAT

An IP socket address is defined as a combination of an IP interface address and a port number.

```
struct sockaddr_in
{
    sa_family_t    sin_family; /* address family: AF_INET */
    u_int16_t      sin_port;   /* port in network byte order */
    struct in_addr sin_addr;   /* internet address */
};
```

```
/* Internet address. */
struct in_addr
{
    u_int32_t    s_addr;    /* IPv4 address in network byte order */
};
```

sin_family is always set to AF_INET. This is required; in Linux 2.2 most networking functions return EINVAL when this setting is missing. sin_port contains the port in network byte order. The port numbers below 1024 are called reserved ports. Only processes with the effective user id 0 or the CAP_NET_BIND_SERVICE attribute set may bind(2) to these sockets. Note that the raw IPv4 protocol as such has no concept of a port, they are only implemented by higher protocols like tcp(4) and udp(4).

sin_addr is the host address. The addr member of struct in_addr contains the host interface address in network order. in_addr should be only accessed using the inet_aton(3), inet_addr(3), inet_makeaddr(3) library functions or directly with the name_resolver (see gethostbyname(3)). IPv4 addresses are divided into unicast, broadcast and multicast addresses. Unicast addresses specify a single interface of a host, broadcast addresses specify all host on a network and multicast addresses address all hosts in a multicast group. Datagrams to broadcast addresses are only passed to the user when the socket broadcast flag is set. To send datagrams to broadcast addresses it has to be set too. Connection oriented sockets are only allowed to use unicast addresses.

Note that the address and the port are always stored in network order, this particularity means that you need to call htons(3) on the number that is assigned to a port. All address/port manipulation in the standard library automatically convert to network order.

SEE ALSO

sendmsg(2), recvmsg(2), socket(4), netlink(4), tcp(4), udp(4), raw(4), ipfw(4)

RFC791, RFC1122, RFC1812

NOM

socket - Créer un point de communication.

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int socket(int domain, int type, int protocol);
```

DESCRIPTION

Socket crée un point de communication, et renvoie un descripteur.

Le paramètre domain indique un domaine de communication, à l'intérieur duquel s'établira le dialogue. Ceci permet de sélectionner la famille de protocole à employer. Ces familles sont définies dans le fichier linux/socket.h.

Les formats actuellement proposés sont :

AF_UNSPEC	Famille non spécifiée, laisser le système la déterminer.
AF_UNIX	Protocoles locaux internes UNIX (pipe,...)
AF_INET	Protocoles Internet (UDP, TCP, etc...)
AF_AX25	Radio amateurs AX.25
AF_IPX	Protocoles Novell.
AF_APPLETALK	Protocoles Apple.
AF_NETROM	Radio amateurs NetRom.
AF_BRIDGE	Passerelle multi-protocoles.
AF_AAL5	Réservé pour l'ATM Werner
AF_X25	Réservé pour le projet CCITT X.25
AF_INET6	Réservé pour le projet IP version 6

Les sockets ont le type, indiqué, ce qui fixe la sémantique des communications. Les types définis actuellement sont :

SOCK_STREAM Support de dialogue garantissant l'intégrité, fournissant un flux de données binaires, et intégrant un mécanisme pour les transmissions de données hors-bande.

Les sockets de ce type sont des flux full-duplex, similaires à des tubes (pipes).

SOCK_DGRAM Transmissions sans connexion, non garantie, de datagrammes de longueur fixe, généralement courte.

SOCK_RAW Transmissions internes au système, le type **SOCK_RAW**, ne peut être utilisé que par le Super-User.

SOCK_RDM Transmission garantie de datagrammes

SOCK_SEQPACKET Dialogue garantissant l'intégrité, pour le transport de datagrammes de longueur fixe. Le lecteur peut avoir à lire le paquet de données complet à chaque appel système read

Le protocole à utiliser sur la socket est indiqué par l'argument protocol. Normalement il n'y a qu'un seul protocole par type de socket pour une famille donnée. Néanmoins

rien ne s'oppose à ce que plusieurs protocoles existent, auquel cas il est nécessaire de le spécifier.

Le numéro de protocole dépend du domaine de communication de la socket. Voir `protocols(5)`.

Une socket de type stream doit être connectée avant que des données puisse y être lues ou écrites. Une connexion sur une autre socket est établie par l'appel système `connect(2)`. Une fois connectée les données y sont transmises par `read(2)` et `write(2)` ou par des variantes de `send(2)` et `recv(2)`.

Quand une session se termine, on referme la socket avec `close(2)`.

Les données hors-bande sont envoyées ou reçues en utilisant `send(2)` et `recv(2)`.

Le protocole de communication utilisé pour implémenter les sockets stream garantit qu'aucune donnée n'est perdue ou dupliquée. Si un bloc de données, pour lequel le correspondant a suffisamment de place dans son buffer, n'est pas transmis correctement dans un délai raisonnable, la connexion est considérée comme inutilisable, et les appels systèmes renverront une valeur -1 en indiquant une erreur ETIMEDOUT dans la variable globale `errno`.

Eventuellement les protocoles peuvent maintenir les sockets en service en forçant des transmissions directes toutes les minutes en l'absence de toute autre activité. Une erreur est indiquée si aucune réponse n'est reçue sur une socket inactive pendant une période prolongée (par exemple 5 minutes).

Un signal SIGPIPE est envoyé au processus tentant d'écrire sur une socket inutilisable, forçant les programmes ne gérant pas ce signal à se terminer.

Les sockets de type SOCK_SEQPACKET emploient les mêmes appels systèmes que celles de types SOCK_STREAM, à la différence que la fonction `read(2)` ne renverra que le nombre d'octets requis, et toute autre donnée restante sera éliminée.

Les sockets de type SOCK_DGRAM ou SOCK_RAW permettent l'émission de datagrammes à des correspondants indiqués au moment de l'appel système `send(2)`. Les datagrammes sont généralement lus par la fonction `recvfrom(2)`, qui fournit également l'adresse du correspondant.

Un appel à `fcntl(2)` permet de préciser un groupe de processus qui recevront un signal SIGURG lors de l'arrivée de données hors-bande. Cette fonction permet également de valider des entrées/sorties non bloquantes, et une notification asynchrone des évènements par le signal SIGIO.

Les opérations sur les sockets sont représentées par des options du niveau socket. Ces options sont définies dans `sys/socket.h`. Les fonctions `setsockopt(2)` et `getsockopt(2)` sont utilisées respectivement pour fixer ou lire les options.

VALEUR RENVOYÉE

socket retourne un descripteur référençant la socket créée en cas de réussite. En cas d'échec -1 est renvoyé, et `errno` contient le code d'erreur.

ERREURS

EPROTONOSUPPORT	Le type de protocole, ou le protocole lui-même n'est pas disponible dans ce domaine de communication.
EMFILE	La table des descripteurs par processus est pleine.
ENFILE	La table des fichiers est pleine.
EACCES	La création d'une telle socket n'est pas autorisée.
ENOBUFS	Pas suffisamment d'espace pour allouer les buffers nécessaires.

NOM

bind - Fournir un nom une socket.

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>
```

```
int bind(int sockfd, struct sockaddr *my_addr, int addrlen);
```

DESCRIPTION

bind fournit à la socket sockfd, l'adresse locale my_addr. my_addr est longue de addrlen octets. Traditionnellement cette operation est appelée "assignation d'un nom à une socket" (Quand une socket est créée, par l'appel-système socket(2),nelle existe dans l'espace des noms mais n'a pas de nom assigné).

NOTES

Assigner un nom dans le domaine UNIX crée une socket dans le système de fichier, qui devra être détruite par le créateur une fois qu'il n'en a plus besoin, en utilisant unlink(2).

Les règles d'assignation de nom varient suivant le domaine de communication. Consultez le manuel du programmeur Linux section 4 pour de plus amples informations.

VALEUR RENVOYÉE

bind renvoie 0 s'il réussit, ou -1 s'il échoue, auquel cas errno contient le code d'erreur.

ERREURS

EBADFF	sockfd n'est pas un descripteur valide.
EACCESS	L'adresse est protégée et l'utilisateur n'est pas le Super-User.
ENOTSOCK	L'argument est un descripteur de fichier, pas une socket.
EADDRINUSE	La socket a déjà une adresse assignée.

Les erreurs suivantes sont spécifiques au domaine UNIX(AF_UNIX):

EINVAL	La longueur addr_len est fausse, ou la socket n'est pas de la famille AF_UNIX.
EROFS	L'i-noeud de la socket se trouverait dans un système de fichiers en lecture seule.
EFAULT	my_addr pointe en dehors de l'espace d'adresse accessible.
ENAMETOOLONG	my_addr est trop long
ENOENT	Le fichier n'existe pas.
ENOMEM	pas assez de mémoire pour le noyau.
ENOTDIR	Un composant du chemin d'accès n'est pas un répertoire.
EACCESS	L'accès à un composant du chemin d'accès n'est pas autorisé.
ELOOP	my_addr contient des références circulaires (à travers un lien symbolique).

NOM

send, sendto, sendmsg - Envoyer un message sur une socket.

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int send(int s, const void *msg, int len, unsigned int flags);
int sendto(int s, const void *msg, int len, unsigned int flags,
           const struct sockaddr *to, int tolen);
int sendmsg(int s, const struct msghdr *msg, unsigned int flags);
```

DESCRIPTION

Send, sendto, et sendmsg permettent de transmettre un message à destination d'une autre socket. Send ne peut être utilisé qu'avec les sockets connectée alors que sendto et sendmsg peuvent être utilisés tout le temps.

L'adresse de la cible est donnée par to avec la longueur tolen. La longueur du message est indiquée dans len. Si le message est trop long pour être transmis intégralement au protocole sous-jacent, l'erreur EMSGSIZE sera déclenchée et rien ne sera émis.

Aucune indication d'échec de distribution n'est fournie par send. Seules les erreurs locales sont détectées, et indiquées par une valeur de retour -1.

Si la socket ne dispose pas de la place suffisante pour le message, alors send va bloquer, à moins que la socket ait été configurée en mode d'entrées/sorties non-bloquantes. On peut utiliser l'appel système select(2) pour vérifier s'il est possible d'émettre des données.

Le paramètre flags peut contenir une ou plusieurs des options suivantes

```
#define MSG_OOB      0x1 /* Traiter les données hors-bande */
#define MSG_DONTROUTE 0x4 /* Contourner le routage          */
```

L'option MSG_OOB est utilisée pour émettre des données hors-bande sur une socket qui l'autorise (par ex : SOCK_STREAM). Le protocole sous-jacent doit également autoriser l'émission de données hors-bande.

MSG_DONTROUTE est utilisée par les programmes de diagnostic ou de routage. Voir recv(2) pour une description de la structure msghdr.

VALEUR RENVOYÉE

Ces appels systèmes renvoient le nombre de caractères émis, ou -1 s'ils échouent, auquel cas errno contient le code d'erreur.

ERREURS

EBADF	Descripteur de socket invalide.
ENOTSOCK	L'argument s n'est pas une socket.
EFAULT	Un paramètre pointe en dehors de l'espace d'adressage accessible.
EMSGSIZE	La socket nécessite une émission intégrale du message mais la taille de celui-ci ne le permet pas.
EWOULDBLOCK	La socket est non-bloquante et l'opération demandée bloquerait.
EPIPE	L'écriture est impossible (correspondant absent), et le signal SIGPIPE a été ignoré par le processus.
ENOMEM	Pas assez de mémoire pour le noyau.
ENOBUFS	La file d'émission de l'interface réseau est pleine. Ceci indique généralement une panne de l'interface réseau, mais peut également être dû à un engorgement passager.

NOM

recv, recvfrom, recvmsg - Recevoir un message sur une socket.

SYNOPSIS

```
#include <sys/types.h>
#include <sys/socket.h>

int recv(int s, void *buf, int len, unsigned int flags);
int recvfrom(int s, void *buf, int len, unsigned int flags,
             struct sockaddr *from, int *fromlen);
int recvmsg(int s, struct msghdr *msg, unsigned int flags);
```

DESCRIPTION

recvfrom et recvmsg sont utilisés pour recevoir des messages depuis une socket s, et peuvent servir à la lecture de données que la socket soit orientée connexion ou non.

Si from est non nul, et si la socket n'est pas orientée connexion, l'adresse de la source du message y est insérée. Fromlen est un paramètre résultat, initialisé à la taille du buffer from, et modifié en retour pour indiquer la taille réelle de l'adresse enregistrée.

L'appel recv est normalement utilisé sur une socket connectée (voir connect(2)) et est équivalent à recvfrom avec un paramètre from nul. Comme ceci est redondant il est possible que recv ne soit plus maintenu dans le futur.

Ces trois routines renvoient la longueur du message si elles réussissent. Si un message est trop long pour tenir dans le buffer, les octets supplémentaires peuvent être abandonnés suivant le type de socket utilisé (voir socket(2)).

Si aucun message n'est disponible sur la socket, les fonctions de réception se mettent en attente, à moins que la socket soit non bloquante (voir fcntl(2)) auquel cas la valeur -1 est renvoyée, et errno est positionnée à EWOULDBLOCK.

Les fonctions de réception renvoient normalement les données disponibles dans la limite du paramètre len sans attendre d'avoir reçu le nombre exact réclamé. Ce comportement peut être modifié avec les options de socket SO_RCVLOWAT et SO_RCVTIMEO décrites dans getsockopt(2). L'appel select(2) peut permettre de déterminer si des données supplémentaires sont disponibles.

L'argument flags de l'appel recv est constitué par un OU binaire (|) entre les valeurs suivantes

MSG_OOB traiter les données hors-bande
MSG_PEEK lire sans enlever les données
MSG_WAITALL attendre le nombre exact ou une erreur

L'option MSG_OOB permet la lecture des données hors-bande qui ne seraient autrement pas placées dans le flux de données normales. Certains protocoles placent ces données hors-bande en tête de la file normale, et cette option n'a pas lieu d'être dans ce cas.

L'option MSG_PEEK permet de lire les données en attente dans la file sans les enlever de cette file. Ainsi une lecture ultérieure renverra à nouveau les mêmes données.

L'option MSG_WAITALL demande que l'opération de lecture soit bloquée jusqu'à ce que la requête complète soit satisfaite. Toutefois la lecture peut renvoyer quand même moins

de données que prévu si un signal est reçu, ou si une erreur ou une déconnexion se produisent.

L'appel `recvmsg` utilise une structure `msg_hdr` pour minimiser le nombre de paramètres à fournir directement. Cette structure a la forme suivante, définie dans `<sys/socket.h>`

```
struct msg_hdr {
    caddr_t    msg_name;        /* optional address */
    u_int     msg_namelen;     /* size of address */
    struct     iovec *msg_iov;  /* scatter/gather array */
    u_int     msg_iovlen;     /* # elements in msg_iov */
    caddr_t    msg_control;    /* ancillary data, see below */
    u_int     msg_controllen;  /* ancillary data buffer len */
    int       msg_flags;      /* flags on received message */
};
```

Ici `msg_name` et `msg_namelen` spécifient l'adresse destination si la socket n'est pas connectée, `msg_name` peut être un pointeur nul si le nom n'est pas nécessaire. `msg_iov` et `msg_iovlen` décrivent les buffers de réception comme décrit dans `readv(2)`. `msg_control`, de longueur `msg_controllen`, pointe sur un buffer utilisé pour les autres messages relatifs au protocole, ou à d'autres données. Les messages ont la forme

```
struct cmsghdr {
    u_int     cmsgh_len;      /* data byte count, including hdr */
    int       cmsgh_level;   /* originating protocol */
    int       cmsgh_type;    /* protocol-specific type */
    /* followed by
    u_char    cmsgh_data[]; */
};
```

Par exemple, on peut utiliser ceci pour être informé des changements du flux de données avec XNS/SPP, ou pour obtenir des données d'identification en demandant un `recvmsg` sans buffer immédiatement après l'appel de `accept` avec ISO.

Les descripteurs de fichiers annexes sont désormais passés en tant que données annexes dans le domaine `AF_UNIX` avec `cmsgh_level` valant `SOL_SOCKET` et `cmsgh_type` valant `SCM_RIGHTS`.

Le champ `msg_flags` est rempli au retour avec les informations concernant le message. `MSG_EOR` indique une fin d'enregistrement, les données reçues terminent un enregistrement (utilisé généralement avec les sockets du type `SOCK_SEQPACKET`). `MSG_TRUNC` indique que la portion finale du datagramme a été abandonnée car le datagramme était trop long pour le buffer fourni. `MSG_CTRUNC` indique que des données de contrôle ont été abandonnées à cause d'un manque de place dans le buffer de données annexes. `MSG_OOB` indique que des données hors-bande ont été reçues.

VALEUR RENVOYÉE

Ces fonctions renvoient le nombre d'octets reçus si elles réussissent, ou -1 si elles échouent, auquel cas `errno` contient le code d'erreur.

ERREURS

<code>EBADF</code>	L'argument <code>s</code> n'est pas un descripteur valide.
<code>ENOTCONN</code>	La socket est associée à un protocole orienté connexion et n'a pas encore été connectée (voir <code>connect(2)</code> et <code>accept(2)</code>).
<code>ENOTSOCK</code>	L'argument <code>s</code> ne correspond pas à une socket.
<code>EWOULDBLOCK</code>	La socket est non-bloquante et aucune donnée n'est disponible, ou un délai de timeout a été indiqué, et il a expiré sans que l'on ait reçu quoi que ce soit.
<code>EINTR</code>	Un signal a interrompu la lecture avant que des données soient disponibles.
<code>EFAULT</code>	Un buffer pointe en dehors de l'espace d'adressage accessible.