Université Paul Sabatier, Toulouse

INITIATION À L'INSTRUMENTATION NUMÉRIQUE :

Utilisation du logiciel LabVIEW 2012 pour la mise en œuvre de cartes multifonctions et le pilotage d'instruments

Annexe (TP)

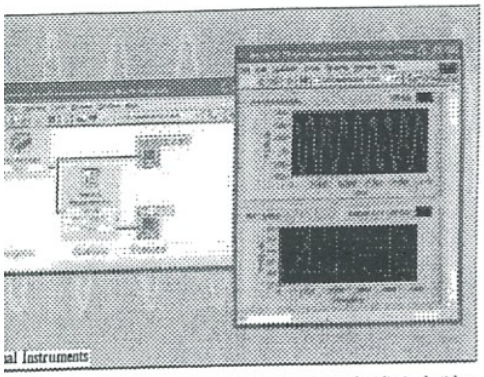
Initiation à l'instrumentation numérique

Annexe aux travaux pratiques.

Table des matières

A-1 Articles de presse "Mesures"

Tendances



al Instruments

W 7 Express intègre des VIs Express qui sont des assistants rapides pour la réalisation des tâches fréquentes comme l'acquisition de données, l'analyse et la visualisation des résultats... Cette version supporte aussi un module pour la programmation d'applications à télécharger dans des PDA.

ACQUISITION ET TEST

Quoi de neuf dans les logiciels de test et mesure ?

Les logiciels d'acquisition de données et plus généralement de test et mesure évoluent sans cesse pour proposer de nouvelles fonctionnalités : intégration logicielle-matérielle de plus en plus immédiate, acquisition et traitement en temps réel des signaux, téléchargement d'applications dans des PDA, etc. Cet article fait un tour d'horizon des principaux environnements du marché avec leurs dernières évolutions.

Il y a maintenant quelques années, l'intégrateur qui désirait mettre au point un banc d'essais se trouvait souvent confronté à un vrai casse tête : il fallait réunir les cartes et les instruments, pouvoir engager la communication avec chacun, c'est-à-dire connaître les instructions de base de chaque instrument ou bien écrire soi-même le driver de la carte ou de l'instrument pour pouvoir l'utiliser avec un langage évolué.
Une fois les essais réalisés, il fallait encore pouvoir rapatrier les mesures sur un poste de travail, les convertir dans un format adéquat, utiliser peut-être un autre logiciel pour le traitement et l'analyse. Bref, chaque étape était un obstacle à franchir.
Aujourd'hui, les logiciels spécialisés dans le test et la mesure permettent de gagner beaucoup de temps dans chacune des phases de la conception et du pilotage d'un moyen d'essai : que ce soit au moment de l'intégration des cartes et des instruments, de l'écriture du programme de test, de l'acquisition des signaux, du traitement et de l'analyse des données ou encore de la visualisation et de la rédaction de rapports, les logiciels de test et mesure simplifient vraiment la vie des techniciens et des ingénieurs d'essais.
L'intégration entre le logiciel et le matériel reste néanmoins le nerf de la guerre : il a été facilité grâce à la standardisation de plusieurs interfaces d'instruments comme le SCPI, l'interface VISA (connue aussi sous le nom de VXI Plug and Play) et plus récemment des drivers de classe d'instruments IVI qui permettent de s'affranchir du matériel utilisé. Cette intégration conditionne encore largement l'achat du matériel comme celui du logiciel.

En bref

- Les logiciels d'acquisition de données et de pilotage d'instruments évoluent
- En plus de l'ajout de nouvelles fonctionnalités, les logiciels gagnent en "ouverture" et simplicité d'utilisation
- National Instruments domine toujours le marché. Matlab de The Mathworks affiche de plus en plus ses ambitions

National Instruments, leader opiniâtre

Depuis plusieurs années, le marché des logiciels de test et mesure et d'acquisition de données est phagocyté par l'Américain National Instruments dont les deux figures de proue sont LabVIEW et LabWindows/CVI. Le premier est un langage graphique assez simple qui permet de programmer à l'aide de blocs fonctionnels et de " tirer des fils" entre chaque bloc. Le second est un environnement de développement en C ANSI conçu spécifiquement pour le test et la mesure ; il est plus difficile d'accès que le premier, et pourtant « il est plus vendu en France que LabVIEW, contrairement aux autres pays du monde », témoigne Patrick Renard, directeur de la communication de National Instruments France.
La version la plus récente de LabVIEW est la version 7 introduite en juillet 2003. Parmi les nouveautés les plus marquantes de cette version, on retiendra l'ajout d'un module de programmation pour PDA. L'utilisateur pourra désormais développer des applications destinées à être exécutées sur un PDA. Ces applications sous forme de VIs (Virtuals Instruments) sont compilées en code élémentaire compréhensible par l'assistant. Elles sont ensuite téléchargées dans la mémoire de l'assistant via une liaison infrarouge, USB ou Ethernet sans fil. Elles apparaissent enfin dans les menus de l'assistant comme n'importe quelle autre application. Rappelons que certains PDA disposent d'emplacements au format PC Card (PCMCIA), format pour lequel un grand nombre d'instruments sont disponibles (cartes d'acquisition, multimètres et même oscilloscopes!). Le champ des applications mobiles ouvert par cette technologie est donc énorme : campagnes de tests sur site, relevés de mesures, systèmes d'acquisition de données portables, mais aussi supervision à distance ; il suffit que le PDA soit doté d'un modem radio et connecté à distance au système de supervision.
NI a d'ailleurs de la suite dans les idées puisqu'il s'est associé récemment à Dap Technologie pour

proposer une tablette PC durcie (sous Windows CE) disponible avec l'environnement LabVIEW. A mi-chemin entre le PDA, l'ardoise électronique et le notebook, la Microflex CE8640 dispose en effet de deux emplacements pour cartes PCMCIA type II.

L'innovation principale de la version 7 de LabVIEW concerne l'entrée en scène de Vis Express. Conçus pour réduire les temps développement, ces assistants de programmation encapsulent des fonctionnalités dans des instruments virtuels de haut niveau, pour les applications de mesure et d'automatisation les plus courantes. L'utilisateur dispose ainsi de 40 Vis Express pour accélérer la réalisation de tâches comme l'acquisition de données, l'analyse du signal, la visualisation ou encore la lecture et l'écriture de fichiers. L'exploitation de ces Vis Express qui se configurent au travers de boîtes de dialogues, nécessite très peu de programmation, voire aucune.

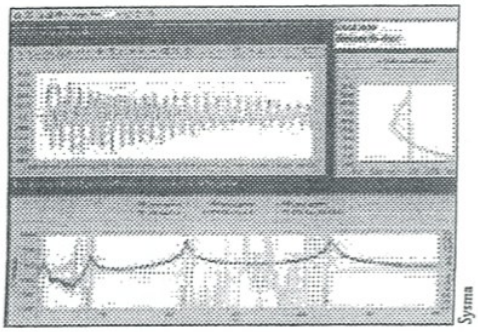
Parmi les autres nouveautés marquantes de LabVIEW 7, on trouve la présence d'un module de configuration de matériel architecturé autour de circuits FPGA. Cela concerne principalement les cartes d'entrées/sorties du constructeur comme la NI PXI-7831R dotées de 8 entrées et de 8 sorties analogiques et de 96 E/S numériques. Le Langage LabVIEW par flux de données se substitue ici à un langage comme VHDL en permettant d'inscrire dans le silicium la définition des entrées/sorties numériques (compteurs/timers par exemple), les relations de synchronisation entre ces entrées/sorties, les cadences de synchronisation, le contrôle en boucle fermée. Le parallélisme logiciel des tâches LabVIEW peut ainsi être implémenté de façon matérielle avec des performances optimales.

La version 7.0 de LabWindows/CVI introduite en septembre 2003 intègre désormais un plan de travail totalement intégré. Auparavant, les utilisateurs avaient parfois plusieurs fenêtres sous les yeux. Désormais c'est fini, le logiciel présente dans un environnement de travail unique l'ensemble des modules de développement : arborescence des bibliothèques et drivers d'instruments, résultats de recherches, messages de débogage, messages d'erreur, etc. De nombreux assistants permettent de générer du code automatiquement, des outils de conversion XML simplifient la création de panneaux de fonctions.

Pour ceux qui préfèrent programmer en Visual C++ ou en Visual Basic, Measurement Studio 7.0 (proposé lui aussi par NI) regroupe un jeu complet de classes et de contrôles pour développer des applications de mesure et d'automatisation dans Visual Studio .NET 2003 de Microsoft avec lequel il offre une compatibilité native.

Tout comme LabVIEW, Measurement Studio et LabWindowsCVI s'appuient sur le nouveau driver de troisième génération NI-DAQmx 7.0 venant succéder au pilote NI-DAQ. Rappelons que ce driver permet de s'interfacer avec l'ensemble du matériel d'acquisition de données de National Instruments. « *Le driver DAQ est en réalité notre principal cheval de bataille. L'équipe qui travaille sur ce pilote est notre plus grosse équipe de développement. Elle est plus importante que celle qui développe LabVIEW* », explique Philippe Baucour, directeur Marketing de National Instruments France. NI estime en effet que pour gagner des parts de marchés dans l'instrumentation modulaire, la solution réside dans l'intégration la plus simple possible du logiciel et du matériel. Selon l'Américain, le nouveau driver de cartes est en effet 10 à 20 fois plus rapide pour les opérations monopoint comme la régulation où l'on doit acquérir un point, faire un traitement et générer une consigne. Le driver supporte également le multitâche (multithreading). Il est possible par exemple de mettre en œuvre des applications pour lesquelles sur une même carte un processus s'occupe des lignes numériques alors qu'un autre gère l'acquisition sur des lignes analogiques.

Il y a une alternative à National Instruments en matière de logiciel de test et mesure : c'est National Instruments ! En effet, ceux qui ne seraient pas séduits par LabVIEW, LabWindows/CVI ou Measurement Studio pourront se tourner vers DasyLab. En réalité, ce logiciel fait partie du catalogue de l'Américain même s'il n'y est pas effectivement. La société qui l'édite a été rachetée par NI, il y a quelques années mais elle n'a pas été intégrée. Dasylab est aujourd'hui considéré comme un concurrent de LabVIEW et se trouve être distribué par des sociétés concurrentes comme SM2I en France par exemple. « *Notre stratégie est de servir nos clients, qui ont opté il y a quelques années pour ce logiciel, en continuant de faire évoluer le produit et non de l'intégrer à LabVIEW* », explique M. Renard. Une preuve de cette bonne volonté est que la version 8 de ce logiciel sera bientôt disponible. Pour Daniel Leymarie, président de SM2I, « *il s'agit du logiciel le plus intuitif du marché. Nous formons nos clients en une seule journée. Cet environnement ne nécessite pas de programmation. Il permet de développer des applications en quelques clics de souris en insérant les icônes appropriées dans le plan de travail puis en les reliant par des fils* ». Dans DasyLab, de très nombreuses icônes



Syrma

WinATS et Marathon offrent la possibilité de faire de l'acquisition en temps réel de signaux analogiques issus de capteurs et de trames CAN synchronisés entre eux.

sont prédéfinies ; elles représentent des modules d'entrée/sortie, des afficheurs, ou des calculateurs.

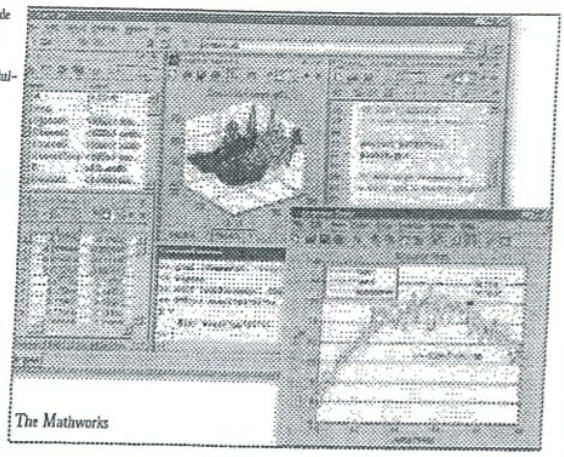
Parmi les nouveautés de la version 8, on retrouve les diagrammes polaires, un module d'enregistrement CAN, la personnalisation des boutons de fonctions et des touches de raccourci, l'aperçu des variables globales, etc. Ce logiciel supporte aussi les drivers IVI, l'interface OPC et le protocole Modbus. Il permet de s'interfacer à plus d'une soixantaine de fournisseurs de matériels dans le domaine de l'acquisition de données dont DataTranslation, Measurement Computing, Adlink, Microstar Laboratories et bien sûr National Instruments.

Plug Adlink, Play LabVIEW

" Plug Adlink, Play LabVIEW " : tel est en effet l'un des slogans d'Adlink en qui NI trouve une concurrence réelle au niveau matériel. Le Taiwanais, distribué en France par SM2I et Ecrin Systems, a lui aussi bien compris que l'intégration logicielle-matérielle est cruciale sur ce marché. Il propose un ensemble de " drivers logiciels " , baptisé PCIS-LVIEW PnP qui permettent de substituer une carte PCI ou PXI de chez National Instruments dans un banc d'essai et de la remplacer par une carte Adlink sans aucune modification du programme logiciel LabVIEW ! Ces drivers fournis sous forme de VIs sont entièrement compatibles avec les drivers des cartes NI DAQ. Ainsi, sur Internet des vidéos d'Adlink circulent où l'on voit la substitution logicielle en question mais où l'on se garde bien d'évoquer la substitution matérielle. Toute la stratégie d'Adlink est basée sur les prix des cartes qui sont cassés continuellement.

Agilent Technologies, le concurrent historique de National Instruments, avec HPVEE a rebaptisé son logiciel VEE Pro (suite à sa séparation d'avec Hewlett-Packard). Cet environnement permet lui aussi d'automatiser les applications de test

Tendances



Matlab permet désormais de faire du pilotage de cartes d'acquisition, du contrôle d'instruments et de la conduite de bancs d'essais.

The Mathworks

et mesure, de piloter des bancs d'essais, de faire du traitement du signal, en temps réel ou en post-acquisition. Il s'agit d'un environnement complet de développement graphique créé pour des ingénieurs pas forcément développeurs. VEE Pro est orienté tâche. Chaque objet du schéma d'ensemble représente une tache particulière que l'on peut paramétrer. Les objets correspondent à un morceau de code du programme. « *Cet environnement permet de contrôler tous les types d'instruments, provenant de n'importe quel fournisseur et au moyen de n'importe quelle interface (GPIB, RS-232, USB ou Ethernet LAN) grâce à l'interface VISA. Il permet même de piloter des cartes d'acquisition au format PCI, des instruments au format VXI et même PXI. Il existe de nombreux drivers pour VEE, qui supporte des standards de l'industrie tels que VXI Plug and Play et IVI-COM. On peut se les procurer chez la plupart des vendeurs d'instruments tels que Tektronix et Rohde&Schwarz. Quant aux drivers pour les instruments Agilent, on peut les télécharger sur notre site, ainsi qu'une version d'évaluation du logiciel. Les clients nous disent que l'avantage le plus flagrant de VEE Pro est le temps nécessaire à l'apprentissage de son utilisation. Un de nos clients n'a mis que 30 minutes pour commencer à programmer depuis le moment il a reçu la copie d'évaluation de VEE Pro* », argumente Ghislain Tietcheu Moukoue, ingénieur d'applications chez *Agilent Technologies*.

La version 7.0 de VEE Pro vient de paraître. Elle offre un accès direct à la plate-forme .NET de Microsoft à travers l'environnement VEE Pro. Il s'agit d'un groupe de composants (appelés classes) qui ont été rangés par catégorie et qui sont disponibles dans VEE Pro pour faciliter la réalisation de certaines tâches informatiques. Prenons l'exemple d'un ingénieur qui a rassemblé et analysé une série de données et qui doit envoyer ces résultats par email à un autre ingénieur. Avec les versions précédentes de VEE Pro, il pouvait le faire, mais il devait passer par une série d'étapes complexes. Avec cette nouvelle version, il doit seulement trouver la tâche dont il a besoin dans le menu, initialiser l'assemblée .NET, saisir les destinataires et le tour est joué.

L'apparence et l'utilisation de VEE Pro ont également été modifiées pour ressembler aux environnements de programmation de Microsoft, tels que celui de Visual Basic. Des fonctionnalités telles que le "undo" (ou défaire en français) et un certain nombre de dispositifs d'édition de propriétés ont été ajoutés pour permettre à l'utilisateur de programmer plus rapidement. Par exemple, l'utilisateur peut choisir une série d'objets en une seule fois, changer la couleur, effectuer des alignements simples, et changer des paramètres. Tout cela, à l'aide d'une table d'édition de propriétés. Ceci permet une prise en main rapide pour la plupart des utilisateurs qui sont familiers avec les outils Microsoft.

Signalons aussi l'existence du kit d'outils T&M d'Agilent qui a été développé pour permettre aux programmeurs de se connecter aux instruments et d'automatiser facilement leur pilotage lorsqu'ils choisissent d'utiliser des langages de programmation comme Visual Basic, ou C++. Tout comme Measurement Studio, il offre la possibilité de faire de Visual Studio .NET un environnement de programmation de test et mesure.

Matlab descend vers l'acquisition et le pilotage

Matlab de *The MathWorks* est à l'origine un langage mathématique évolué comprenant un très grand nombre de fonctions pour faire du calcul numérique complexe (calcul matriciel, transformée de Fourier rapide, transformée en z, calcul vectoriel, traitement d'image, etc.). Il est très largement utilisé au sein des équipes de recherche et développement dans les phases de simulation et de conception des systèmes et ce dans toutes les industries (automobile, aéronautique, télécommunications, etc). En test et mesure, il était traditionnellement utilisé pour faire de l'analyse et du calcul sur les résultats d'essais. Dans la version 6.0 de VEE Pro, il y avait déjà la possibilité d'exécuter des scripts Matlab au sein même de l'environnement VEE Pro, donnant ainsi à l'utilisateur un accès facile aux capacités de calcul et d'algorithmique de Matlab (en particulier la Signal Processing Toolbox). Beaucoup d'utilisateurs de VEE ont profité de cette intégration de Matlab dans VEE. Mais désormais, les inconditionnels de Matlab peuvent ne plus sortir de leur environnement favori puisque celui-ci leur permet, au même titre que les environnements traditionnels, de faire de l'acquisition de signaux sur carte et de piloter l'ensemble des instruments d'un banc d'essai. Outre l'environnement logiciel de base, il faut pour cela se doter de certaines options logicielles appelées Toolbox. Ainsi, la "*Data Acquisition Toolbox*" permet de faire de l'acquisition sur carte PCI. Cette "boîte logicielle" permet de se connecter à la plupart des cartes du marché et de taper des scripts Matlab pour déclencher l'acquisition des cartes, rapatrier les mesures et visualiser immédiatement les résultats dans Matlab. Environ 500 cartes du marché sont supportées, dont un grand nombre de chez *National Instruments*. « *Il suffit de connecter la carte au PC. Si celle-ci fait partie du répertoire de la Data Acquisition Toolbox, elle est immédiatement reconnue par Matlab. La communication entre la carte et Matlab est archi simple : elle s'effectue au moyen de 5 ou 6 commandes, ce n'est pas une usine à gaz. Si la carte n'est pas supportée par la Toolbox, on peut s'adresser au constructeur de celle-ci, qui aura sans doute déjà écrit le driver pour la carte* », explique Bertrand De Labrouhe, directeur Marketing de *The Mathworks*. Le principe est le même pour l'"*Instrument Control Toolbox*" qui permet de piloter des instruments comme des oscilloscopes et pour l'*Image Acquisition ToolBox* qui ouvre la voie aux cartes d'acquisition d'images. La version 7.0 de Matlab introduite récemment supporte l'interface VISA ainsi que les drivers de classe IVI.

Acquisition temps réel sur bus CAN avec WinATS et Marathon

La société *Sysma* basée à Aix en Provence édite WinATS et Marathon, deux logiciels d'acquisition, de pilotage d'instruments et d'analyse des données, qui sont à l'heure actuelle installés sur plus de 1 500 systèmes chez de grands groupes industriels. Parmi eux, on ne citera que *Renault, PSA* et *Valeo* dans l'automobile, *EADS* dans l'aéronautique, le *CEA, EDF* et

GDF dans l'Energie, Alstom et la SNCF dans le ferroviaire.

WinATS séduit vraiment par sa facilité d'utilisation : ce logiciel graphique ne nécessite aucune programmation : tout n'est que paramétrage et configuration. Pas de synoptiques tentaculaires dans l'éditeur de projet : on visualise en un clin d'œ il les différentes voies d'acquisition représentées en parallèle à l'horizontal. Sur chacune de ces voies, des blocs fonctionnels définissent chaque phase du cycle d'essai : acquisition, stockage, traitement, condition d'arrêt du cycle. Selon son éditeur, deux jours de formation suffisent pour le prendre en main complètement.

Développé en collaboration avec le secteur automobile, Marathon est quant à lui un logiciel spécialisé dans le suivi d'essais d'endurance et de caractérisation avec régulation PID. Il assure le pilotage de plusieurs bancs d'essais en parallèle et permet le stockage des données et la visualisation en cours d'essai sur des entrées/sorties analogiques, numériques et de type CAN.

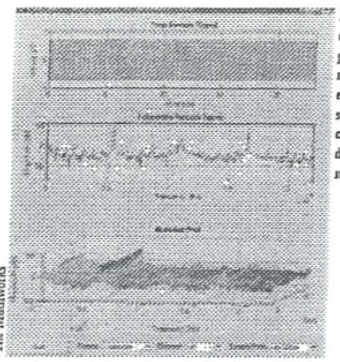
Le plus gros point fort de ces logiciels est de pouvoir réaliser des acquisitions et du pilotage de manière déterministe grâce au noyau temps réel RTX de *Venturcom* se substituant à l'horloge Windows. « *La migration de DOS vers Windows a été une régression au point de vue du déterminisme. Nous avons donc opté pour le noyau RTX de Venturcom qui a l'avantage de se greffer sur l'environnement Windows, tout en lui apportant des caractéristiques temps réel* », explique Eric Cerruti, directeur commercial de Sysma. Ce noyau permet ainsi de cadencer la prise et la génération d'échantillons à une fréquence maximale de 10 kHz avec une gestion prioritaire de l'horloge sur le système d'exploitation. Suivant le besoin, l'utilisateur peut également configurer une fréquence d'échantillonnage plus faible. L'écart temporel est ainsi déterminé et fixe.

Actuellement, la plupart des véhicules sont équipés du multiplexage sur bus CAN pour permettre un dialogue entre les différents calculateurs du véhicule (injection, ABS, ESP, etc.). L'acquisition des nombreuses informations circulant sur ce bus représente un intérêt indéniable pour étudier le comportement, la fiabilité et les performances des véhicules et de leurs différents organes. Avec WinATS et Marathon, les ingénieurs et les techniciens d'essais ont la possibilité de mesurer des paramètres analogiques (température, accélération, pression, vitesse...) sur des systèmes d'acquisition classiques (cartes, centrales d'acquisition) mais aussi les paramètres circulant sur le bus CAN. L'ensemble de ces données sera horodaté avec la même base de temps. Cette solution a plusieurs avantages. D'une part, elle permet de comparer les données analogiques et les trames CAN sur une même échelle de temps. Elle permet aussi d'éviter de multiplier les entrées analogiques, de rajouter des câbles et des capteurs en récupérant un certain nombre de paramètres sur le bus CAN.

Traitement statistique avec FAMOS

Famos proposé par IMC et distribué en France par *Johne et Reilhofer* n'est pas à proprement parler un logiciel d'acquisition et de pilotage de systèmes ; il est plutôt utilisé pour faire de l'analyse et du traitement des données d'essais. Il peut cependant être connecté à l'ensemble des systèmes du catalogue de l'Allemand (centrale d'acquisition, boîtiers, cartes) au moyen de l'extension logicielle IMC Device. Cette option permet de réaliser des systèmes d'acquisition et de traitement temps réel évolués grâce à la présence de DSPs intégrés sur certains matériels. Le grand atout de Famos est qu'il intègre de nombreuses fonctions pour faire du calcul statistique sur les données d'essais. Il offre également la possibilité de visualiser de manière synchronisée une vidéo d'un essai (crash test par exemple) et les paramètres mesurés lors de cet essai (courbe de déformation mécanique par exemple, avec curseur de temps). Cette possibilité permet de corréler avec exactitude les résultats de l'expérience avec ce qui s'est réellement produit.

Bertrand Braux

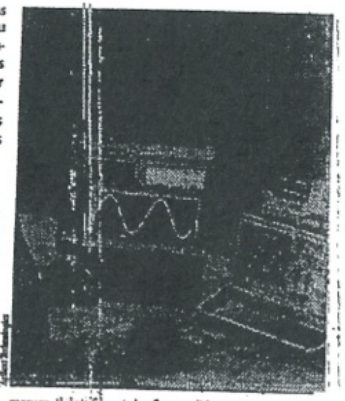


The Mathworks

Le Matlab Compiler est une extension qui permet de transformer le code Matlab en un code exécutable sur n'importe quelle cible ne disposant pas de Matlab Et ce sans royalties...

Le logiciel standardise le contrôle d'instrument

Au sein d'un banc de tests, la communication inter instruments ne dépend pas uniquement du bus physique utilisé. Elle réclame une compréhension au niveau de l'application des commandes utilisées pour contrôler les instruments. A l'heure où différentes connexions (RS-232, GPIB, Ethernet, USB) sont disponibles sur les instruments, les protocoles logiciels prennent de plus en plus d'importance pour réaliser l'interface entre l'application de tests et les instruments. Rendre les différentes couches indépendantes, s'affranchir du matériel, pouvoir interchanger les instruments, les bus et les applications : une quête du Graal qui dure depuis plus de 25 ans…



L'invention du GPIB a facilité l'interconnexion physique des instruments mais il n'a pas facilité, pour le programmeur, la discussion avec chaque instrument. Adopté en 1987, le protocole IEEE-488.2 a standardisé les formats des messages d'instruments, les protocoles des contrôleurs et défini un ensemble de commandes, et une structure de renvoi d'état standard. Cette interface a permis d'unifier le contrôle d'instruments fabriqués par des centaines d'entreprises différentes. Néanmoins, chaque famille d'instruments d'un même constructeur avait encore son propre jeu d'instructions, qui pouvait différer largement selon les constructeurs. Il était alors impossible d'intervertir les instruments sans modifier les scripts en profondeur. Le programmeur qui avait développé un programme conséquent avec un jeu d'instructions d'une famille d'instruments se trouvait lié au constructeur de cette famille. En 1990, a été créé le CPI, une initiative ayant pour but de standardiser les commandes entre les instruments provenant de tous les constructeurs. Le projet a permis d'harmoniser un grand nombre d'instructions entre les différentes milles d'appareils mais à l'heure actuelle s instruments de chaque constructeur ont ncore des commandes spécifiques. Avec le veloppement du VXI, l'apparition de nouaux bus comme MXI ou Ethernet et la mplexité rémanente de l'utilisation des us série RS-232, le besoin s'est fait sentir pouvoir s'affranchir de la connexion utiée pour contrôler les équipements. C'est alors que fut développée l'API (ou Application Program Interface) VISA. Encore aujourd'hui, l'intérêt de la Virtual Software Architecture est évident. Cette API que l'on peut installer sur un PC ou une station Sun fournit un jeu d'instruction pour le contrôle d'instruments permettant de s'affranchir de la connexion utilisée (RS-232, MXI, GPIB, Ethernet…). Elle utilise des services de plus bas niveaux pour piloter ces différentes liaisons (VISA utilise le driver IEEE-488.2 par exemple). Prenons l'exemple d'une application logicielle de tests développée avec VISA, qui piloterait des équipements grâce à des liens série. On peut substituer les liens série à des liens GPIB sans modifier le programme de tests. Il suffit juste de changer quelques variables permettant de prendre en compte le changement. Pour Guy Knapp, d'*Agilent Technologies*, *« VISA est la plate-forme logicielle incontournable quand on veut utiliser plusieurs connexions différentes ou des adaptateurs de connexion. »*

Les drivers d'instruments simplifient la programmation

A l'heure actuelle, les applications de tests conséquentes ne sont plus écrites avec des instructions bas niveau de type VISA ou IEEE-488 mais elles sont développées grâce à des environnements de développement. Le marché de ces logiciels est phagocyté par *National Instruments* avec LabView (environnement graphique) et LabWindows/CVI (language C) qui sont les grosses références en la matière. Lorsqu'un constructeur vend un équipement de mesure, il doit fournir les fameux "drivers d'instruments" pour ces logiciels qui sont en général téléchargeables sur son site Internet. *« Si nous ne fournissons pas les drivers d'instruments pour LabView ou LabWindows/CVI qui sont avec nos équipements, les clients ne nous les achètent pas »*, témoigne Laurent Gilli de Quality Source. Un driver d'instrument comporte un jeu de fonctions haut niveau servant à initialiser l'instrument, à configurer les paramètres de mesure, à aller lire les mesures, à éventuellement les traiter, puis à fermer la session de communication. Les fonctions du driver sont construites à partir de plusieurs commandes bas niveau de type SCPI. En choisissant de développer un programme avec les drivers des Instruments utilisés, le développeur gagne du temps et s'affranchit des contraintes inhérentes à la programmation bas niveau.

En bref…

- Les couches logicielles gérant la communication entre les instruments d'un banc d'essais permettent de se libérer des contraintes inhérentes au matériel utilisé (câbles et instruments).
- Avec l'API VISA, on s'affranchit de la connectique utilisée (RS-232, GPIB, Ethernet…).
- Avec les drivers d'instruments IVI, on peut intervertir les appareils des différents constructeurs.

Il n'empêche que les principales remarques adressées par les programmeurs à l'encontre des drivers d'instruments sont les suivantes :

- Chaque constructeur étant libre de développer son driver comme il l'entend, on se retrouve avec des drivers très hétérogènes, tant d'un point de vue qualité que d'un point de vue fonctionnalité (en fait, certains drivers sont loin d'être achevés). Or, ces drivers se développent à un rythme soutenu, il y en a plus de 1 600 actuellement.
- Ils n'offrent pas de mode de simulation qui permettrait de développer l'application même si l'instrument n'est pas physiquement présent.
- Ils sont élaborés pour un seul instrument et ne permettent pas d'interchanger les instruments.

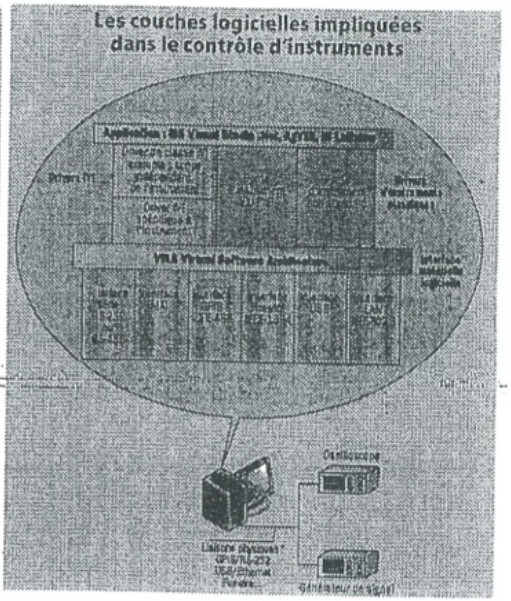
Afin de pallier à tous ces défauts, les instrumentiers se sont ralliés en 1998 pour créer la fondation IVI (InterchangeableVirtual Instruments) dont l'objectif est de définir un certain nombre de règles pour la fabrication de drivers d'instruments. Cette fondation propose la réalisation de drivers en comportant deux parties distinctes : les drivers de classe et les drivers spécifiques d'instruments.

Les drivers de classe sont bâtis sur un jeu de fonctions communes à tous les instruments d'une même famille (par exemple les multimètres). Cela veut dire concrètement que le développeur appelle les mêmes fonctions pour programmer un multimètre Fluke 45 que pour programmer un HP34401A. Les drivers spécifiques comprennent un ensemble de fonctions spécifiques à l'instrument susceptibles de s'interfacer avec les drivers de classe. Autrement dit les fonctions de classe génériques à toute une famille d'instrument appellent des fonctions spécifiques permettant de contrôler réellement l'instrument. A chaque fois qu'un constructeur propose un nouvel instrument, il doit écrire uniquement la partie spécifique du driver d'instruments. Pour cela, il doit se plier à des spécifications d'écriture (ou modèle d'attribut) définies par le consortium IVI, garantissant la qualité et le niveau fonctionnel du driver écrit. Ces drivers IVI permettent également de fonctionner en mode simulé, c'est-à-dire sans la présence physique des instruments, ce qui est unique aujourd'hui. Ils autorisent également l'utilisation de drivers spécifiques traditionnels ou même de commandes bas niveau de type SCPI au sein d'une même application logicielle. Et comme leur nom l'indique la programmation avec des fonctions de classe IVI permet d'interchanger les instruments, et ce même lorsque l'application fonctionne.

Bertrand Braux

Le point sur…

- **IEEE-488.2.** Cette spécification définit le protocole de communication entre tous les équipements d'un bus GPIB. Il s'agit d'une interface logicielle définissant la structure des échanges sur le bus (protocole des contrôleurs, ouverture de lignes, renvoi d'états standard…).
  Renseignements : www.lee.org
- **SCPI Standard Commands for Programmable Instrumentation.** Crée en 1990, cette initiative a pour but de standardiser les jeux de commande bas niveau auxquels répondent les instruments dans le but de pouvoir les intervertir et ce indépendamment du constructeur.
  Renseignements : www.scpiconsortium.org
- **VISA Virtual Software Architecture.** Inventée par *National Instrument* et standardisée par l'alliance VXI Plug & Play (en cours de normalisation sous la référence IEEE P1226.5), la librairie VISA fournit une interface entre le logiciel et la connectique utilisée (VXI, GPIB, Ethernet) permettant de s'affranchir à la fois de l'application (C, C ++, LabView) et de l'environnement (Windows/Unix).
  Renseignements : www.vxipnp.org
- **IVI Instrument Virtual.** Créée en 1998, la fondation IVI regroupe les plus grands noms du domaine du test et de la mesure (40 au total) dans le but de mettre au point un standard pour les drivers d'instruments. Ces drivers comportent une partie commune à toute une classe d'instruments (Ex. : les oscilloscopes) et une partie spécifique à chaque instrument. L'objectif est encore une fois de pouvoir les intervertir.
  Renseignements : www.ivifoundation.org



A-2 La commande d'instruments

A-2.1 Introduction

Un système de mesure informatisé exploite les ressources d'instruments programmables dans le but de réaliser des mesures automatiquement (sans intervention humaine). Les instruments sont contrôlés par un dispositif particulier appelé **contrôleur** (souvent un PC associé à une carte spécialisée). Dans un système modèle, les instruments et un seul contrôleur utilisent un **protocole** commun de communication (ensemble commun de moyens logiciels et matériels permettant le dialogue entre les appareils).

La liaison physique entre appareils est réalisée par un ensemble de câbles (ou lignes) appelé **bus**. Le bus véhicule des signaux de commandes et des signaux de données d'un appareil émetteur (talker) à un ou éventuellement plusieurs récepteurs (listeners). Les extrémités du bus sont équipées de connecteurs spéciaux qui s'emboîtent dans des connecteurs duaux situés généralement sur la face arrière des instruments. Au niveau des dispositifs (contrôleur ou instrments), le connecteur transmet un signal émis ou reçu par un circuit électronique spécialisé programmable appelé **interface**. L'interface, constituée de plusieurs blocs fonctionnels, assure, entre autres, une adaptation de type électrique entre le bus et les circuits internes de l'instrument (ou du contrôleur) et une mise en forme des signaux selon le cahier des charges fixé par le protocole.

A-2.2 Notion de liaison et d'interface

Une **liaison** est un système de communication entre **deux** systèmes. Quand le nombre de systèmes interconnectés est supérieur à deux, il est d'usage de parler de **bus**. Indépendamment du système de communication, la connexion est qualifiée de **série** (resp. **parallèle**) si l'information (codée sous 2^n bits) est transmise en **n** (resp. **1**) coups d'horloge. Ceci découle de l'option matérielle choisie pour transmettre la donnée : le support série utilise **un** seul fil alors que le support parallèle utilise **n** fils. Il existe une multitude de liaisons et de bus (RS232, IDE, RS485, USB, I2C, GPIB, MIDI, CAN, ...) [†].

Dans un appareil (exemple un PC), les données sont échangées entre les circuits intégrés (CI) qui le constituent (processeur, mémoires, ...) via un support constitué de n conducteurs qui assure la transmission simultanée (en parallèle) des n bits constituant la donnée. Si l'on souhaite envoyer une donnée vers un autre appareil via une liaison série, ces n bits ne pourront pas être transmis simultanément (un conducteur unique). Il est donc indispensable d'insérer, **sur chaque appareil**, un dispositif spécialisé (UART – Universal Asynchronous Receiver/Transmitter) qui assure, entre autres fonctions, à l'émission une conversion parallèle-série et, à la réception, une conversion série-parallèle (utilisation de registres à décalage). Ce CI fait partie d'un circuit plus complexe désigné sous le nom d'**interface**.

Le rôle de l'interface ne se limite pas à une conversion série-parallèle. D'un point de vue électrique, elle évite, en particulier, que la présence de la connexion de perturbe pas le fonctionnement

†. IDE = integrated drive electronics ; USB = universal serial bus ; GPIB = general purpose interface bus ; CAN = controller area network ; I2C = inter-integrated circuit ; RS = recommended standard ; MIDI = musical instrument digital interface

des appareils (problème d'adaptation classique). L'interface est donc un point de rencontre d'environnements différents. Un terme synonyme d'interface est **port**.

Ces TP exploitent deux interfaces couramment utilisées dans le monde de l'instrumentation : l'**interface GPIB** et l'**interface RS232**. Dans le standard GPIB, la partie du bus transmettant les données (bus de données) est constitué de 8 lignes. On peut donc transmettre simultanément (**liaison parallèle**) les 8 bits associés au codage d'une donnée élémentaire (octet ou byte). Le standard RS232 utilise par contre un bus de données comportant un seul fil (**liaison série**). Par conséquent, à fréquence d'horloge identique, la transmission d'octets sous GPIB doit être huit fois plus rapide que sous RS232.

La programmation directe (dite aussi de **bas niveau**) d'une interface (et donc des appareils) nécessite des connaissances approfondies et passe par la manipulation et/ou l'élaboration de logiciels spécialisés appelés **drivers** de bas niveau. Pour des personnes non spécialistes et/ou uniquement intéressées par le contrôle d'instruments, cette programmation se fait de manière aisée et transparente par des "interfaces" logicielles dites de **haut niveau**.

Vous utiliserez le logiciel LabVIEW et l'interface logicielle de haut-niveau VISA. Cette dernière est spécialisée dans le contrôle d'appareils. Les appareils mis à votre disposition adoptent un "standard" appelé **SCPI** (prononcer "SKIPI"). Les fonctions VISA sont utilisées pour envoyer les instructions SCPI aux différents appareils.

A-2.3 VISA et SCPI

Définition et intérêts

VISA est une interface de programmation d'application d'E/S (Entrées/Sorties) standard pour la programmation d'instrumentation. L'interface (logicielle) VISA est une interface de haut niveau qui appelle des drivers de bas niveau. VISA est le standard industriel pour le développement de drivers d'instruments. Outre son intérêt présenté dans l'introduction, cette interface utilise des fonction **identiques** pour communiquer avec les instruments **quel que soit** le type d'interface (GPIB, VXI, RS232C).

Architecture VISA

Le gestionnaire de ressource par défaut se trouve au niveau le plus élevé des opérations VISA. LabVIEW établit automatiquement une communication avec le gestionnaire de ressources par défaut lors du premier appel de VI VISA. On appelle "**ressource**" une entité avec laquelle vous pouvez communiquer (instruments ou mémoires) et "**session**" une connexion (lien) à n'importe quelle ressource existante. Vous devez utiliser le gestionnaire de ressources par défaut VISA pour ouvrir des sessions vers d'autres ressources. Vous devez ouvrir des sessions vers des instruments avant qu'un VI ne puisse communiquer avec eux.

Recherche des ressources

La fonction **VISA Find Resource** recherche les ressources disponibles dans le système. Cette fonction est un point de départ classique pour un programme VISA. Vous pouvez l'utiliser pour déterminer si toutes les ressources nécessaires à l'exécution de l'application sont disponibles. La

seule entrée nécessaire est une **chaîne de caractères** appelée l'**expression de recherche**. Par exemple, l'expression de recherche **?*INSTR** permet de trouver toutes les ressources d'instruments. La fonction VISA fournit une liste sous forme de **tableau de chaîne de caractères**. Chaque chaîne contient la description de l'une des ressources trouvées. Ces chaînes sont appelées des **descripteurs d'instruments**.

Classe VISA

Une **classe** VISA est un groupe qui contient certaines ou l'ensemble des opérations VISA. **Instr** est la classe la plus courante qui couvre toutes les opérations VISA pour un instrument. La sélection de la classe **GPIB Instr** implique que seules les fonctions d'opérations associées à la classe de périphériques GPIB peuvent être câblées correctement à cette session.

Ouverture d'une session

Les descripteurs d'instruments sont utilisés pour ouvrir des sessions vers les ressources du système via la fonction **VISA Open**. Dans la plupart des applications, les sessions n'ont besoin d'être ouvertes qu'une fois pour chaque instrument avec lequel l'application communique. Cette session peut être transmise à travers l'application, puis fermée à la fin de l'application.

Fermeture d'une session

Une session ouverte vers une ressource VISA utilise également les ressources système de l'ordinateur. Pour fermer correctement un programme VISA, toutes les sessions VISA ouvertes doivent être fermées. Pour cela, utiliser la fonction **VISA Close**. L'entrée de session VISA de la fonction VISA Close est la session à fermer. Cette session provient en général d'un terminal de sortie VISA Session d'une session VISA Open.

Gestion d'erreurs

Chacun des VI VISA contient des terminaux d'entrée et de sortie d'erreur utilisés pour passer des clusters d'un VI VISA à un autre dans un diagramme. Le cluster d'erreur contient :

a) un drapeau (flag) booléen indiquant si une erreur s'est produite

b) un code d'erreur VISA numérique

c) une chaîne de caractères contenant l'emplacement du VI dans lequel l'erreur s'est produite.

Si une erreur se produit, les prochains VIs n'essaient pas de s'exécuter et passent simplement le cluster d'erreur. Une aide en ligne permet d'obtenir la description de l'erreur associée au code affiché (ex. : code = -1073807246 Name = VI_ERROR_RSRC_BUSY Description = La ressource est valide mais VISA ne peut pas l'atteindre actuellement.
N.B. : dans une application VISA, l'ordre d'exécution des différentes fonctions VISA est fixé par le cheminement imposé au cluster d'erreur.

Communication "basée message"

Les périphériques série, GPIB et de nombreux périphériques VXI reconnaissent différentes **commandes "basées** (sur des) **messages"**. Les VI utilisés pour réaliser ces opérations sont **VISA Write** et **VISA Read**. Comme mentionné plus haut, les **mêmes** VI sont utilisés pour écrire des

commandes basées messages aux instruments GPIB, série et VXI. VISA repère automatiquement les fonctions de driver à appeler, selon le type de ressource utilisée.

N.B. : certains instruments exigent, pour une communication série, un symbole ASCII non affichable accolé immédiatement après le dernier caractère du message.

Les commandes réelles "basées messages" reconnues par l'instrument varient selon les fabricants. Néanmoins les instruments récents de même fonctionnalité partagent un jeu de commandes "basées messages" **commun s'ils adhèrent au standard SCPI**.

Propriétés VISA

En plus des opérations de communication de base précédentes, les ressources VISA ont différentes propriétés (appelées **attributs**) dont les valeurs peuvent être lues ou définies dans le programme. Dans un programme LabVIEW, ces propriétés sont gérées de la même façon que les propriétés des indicateurs et des commandes de la face avant (**attribute nodes**). Les nœuds de propriétés (**property nodes**) sont utilisés pour lire ou définir les valeurs des propriétés VISA (exemples : le taux de transfert ou baud rate dans une liaison série). On distingue deux types fondamentaux de propriétés VISA : les **propriétés globales** et les **propriétés locales**. Les propriétés globales sont spécifiques à une ressource alors que les propriétés locales sont spécifiques à une session.

Evénements

Un événement est un moyen de communication VISA entre une ressource et ses applications. C'est une façon pour la ressource de notifier à l'application qu'une condition s'est produite et qu'une action de la part de l'application est nécessaire.

Verrouillage

VISA introduit des verrous pour le contrôle d'accès des ressources. Dans VISA, les application GPIB et VXI peuvent ouvrir simultanément plusieurs sessions pour la même ressource et accéder en même temps à la ressource grâce à ces différentes session. Dans certains cas, des applications qui accèdent à une ressource doivent empêher d'autres sessions d'accéder à cette même ressource. Par exemple, une application peut avoir besoin d'exécuter une opération de lecture et une opération d'écriture immédiatement après, de sorte qu'aucune opération ne doit avoir lieu entre les opérations de lecture et d'écriture. L'application peut donc verrouiller la ressource avant d'appeler l'opération d'écriture et la déverrouiller après l'opération de lecture.

N.B. : comme mentionné précédemment, dans une application VISA, l'ordre d'exécution des différentes fonction VISA est généralement fixée par la chaîne d'erreur.

A-2.4 Standards IEEE488.1, IEEE488.2, SCPI

Jusqu'au milieu des années soixante-dix, la communication entre ordinateur et instruments était désespérante. Chaque type d'ordinateur avait sa propre structure de bus et son propre protocole d'interface. Il fallait acheter (voire construire) des cartes d'interface spécifiques à l'ordinateur ainsi que les câbles associés. Ce défaut général de compatibilité s'étendait aux périphériques eux-mêmes. On ne pouvait généralement pas connecter deux instruments entre eux car les appareils fournis

par différents fabricants utilisaient souvent des signaux différents et des conventions différentes concernant le transfert des données.

Norme IEEE488.1

Pour remédier à ceci, un standard fut proposé sous l'impulsion de la firme Hewlett-Packard (HP). Le standard IEE488.1 définit les caractéristiques électriques et mécaniques du General Purpose Interface Bus (GPIB). Des protocoles sont fournis pour établir une liaison (lien de communication) entre un parleur (talker ou sender) et un ou plusieurs écouteurs (listeners). Le standard indique aussi comment synchroniser le transfert des données (handshake). Il définit aussi une relation hiérarchique entre les dispositifs du système : un soi-disant **contrôleur** répertorie les dispositifs qui participent à la liaison et détermine leurs rôles (distinction entre **talker** et **listener**). De par les contraintes de charge, un maximum de 15 dispositifs peuvent être connectés au GPIB. Chaque dispositif est identifié par son adresse unique qui doit avoir une valeur comprise entre 0 et 30.

Norme IEEE488.2

Quoique le standard IEEE488.1 définisse **comment** les données doivent être transmises, il n'était pas spécifié **quelles** données devaient être transmises. En pratique, ceci conduisait au problème suivant : deux instruments réalisés par des constructeurs différents se comprenaient rarement. Cette situation peut être comparée à une liaison téléphonique : la standardisation internationale permet d'établir une liaison d'excellente qualité vers l'étranger mais si les deux interlocuteurs ne parlent pas la même langue, la communication est impossible.
De cette préoccupation naquit un deuxième standard appelé IEEE488.2 et **non limité** aux instruments GPIB. Ce standard définit la syntaxe pour **tous** les types de données qui pourraient être utilisés dans des systèmes de mesure. De plus, il définit les protocoles de communication pour **l'échange de messages** (suite de caractères ASCII) et fait la distinction entre **commandes** et **requêtes** (ces dernières étant identifiées par un point d'interrogation placé à la fin de la commande). Il définit également un système précis de report d'état (status) et d'erreurs ainsi qu'un nombre limité de commandes appelées "**common commands**". Le jeu des common commands est reconnu par tous les instruments dits compatibles IEEE488.2. Par exemple, la commande ***RST** a pour effet de placer l'instrument dans un état connu (réinitialisation) défini par le constructeur. La requête ***IDN ?** demande et permet d'obtenir la "carte d'identité" d'un instrument.
Un des objectifs de ce standard (qui complète donc le standard IEEE488.1) fut de sécuriser les chaînes d'acquisition. En effet, les systèmes d'instrumentation doivent fonctionner parfaitement sans intervention humaine. Le standard gère les conditions d'erreur afin d'obtenir une réaction correcte de la part des instruments. D'autre part, il permet de synchroniser des appareils ou/et de déclencher une action après que les précédentes aient été achevées.

Norme SCPI

Malgré l'amélioration apportée par la norme IEEE488.2, il n'existait pas encore de message standard dans le sens où une opération de paramétrage d'un GBF (par exemple fixer l'amplitude de sortie à 10V) nécessitait une syntaxe propre au constructeur. Ceci impliquait la consultation d'un nouveau manuel si ce GBF devait être remplacé (panne par exemple) par GBF d'une autre marque. Pour remédier à ce problème, un standard s'appuyant sur le standard IEEE488.2 fut officialisé en 1990. Il s'agit du standard **SCPI** (Standard Commands for Programmable Instruments – prononcer

"SKIPI"). Les commandes (messages) SCPI utilisent une **syntaxe intuitive**, donc simple à mettre en œuvre et à retenir. Le standard définit une règle utilisant des **mot-clés** (keywords) écrits sous forme abrégée ou non et associés selon une structure arborescente Si deux générateurs de deux constructeurs différents sont compatibles SCPI (et donc a fortiori au standard IEEE488.2) la commande **FREQuency 5.0E+3** (forme longue) ou **FREQ 5.0E+3** (forme abrégée) sera comprise par les deux appareils et aura pour effet de fixer à 5000 Hz la fréquence des signaux générés.
Généralement, un appareil est doté de quelques commandes SCPI confirmées mais aussi (parfois) de commandes spécifiques qui, bien que non officielles, sont conçues dans l'esprit de la norme SCPI (et qui seront peut-être un jour officialisées car le jeu de commandes SCPI s'enrichit au fil des ans).

A-3 La liaison RS232

Bibliographie :
— L'interface RS232 de John Campbell (Sybex)
— La liaison RS232 de Philippe André (ETSF)
— Aide enn ligne LabVIEW

A-3.1 Origine de la norme RS232

Sans rentre trop dans les détails, on retiendra que la liaison RS232 a été développée historiquement pour relier des terminaux alphanumériques à un ordinateur (calculateur) par l'intermédiaire du réseau téléphonique.

Classification des appareils en deux types :

a) Le **DTE** qui reçoit ou qui envoie des informations (terminal vidéo ordinaire constant en un clavier et un écran de visualisation).

b) Le **DCE** ou équipement de communication ou de transfert de données, c'est-à-dire un équipement servant d'interface avec un réseau de communication. Le DCE se contente donc de transférer les données du DTE vers le réseau téléphonique ou du réseau téléphonique vers le DTE.
En pratique, la communication entre deux appareils via l'interface série RS232 peut s'avérer difficile. La difficulté consiste à déterminer la structure du "câble" RS232 qui reliera les deux appareils : liaison DCE/DTE ou liaison DTE/DTE (cf. paragraphe ci-dessous).

A-3.2 Connectique

Sur un appareil, un connecteur spécifique trahit la présence d'une interface RS232. Le connecteur le plus répandu est un connecteur en forme de D appelé SUB-D9 (9 broches ou 9 douilles selon le "sexe"). Un observateur constatera que les connexions sont numérotées selon la convention représentée sur la figure A-1. Remarquer la cohérence de la numérotation sachant qu'un connecteur mâle ne peut s'emboîter que dans un connecteur femelle. Un autre type de connecteur, le SUB-D25, contient 25 broches ou douilles.

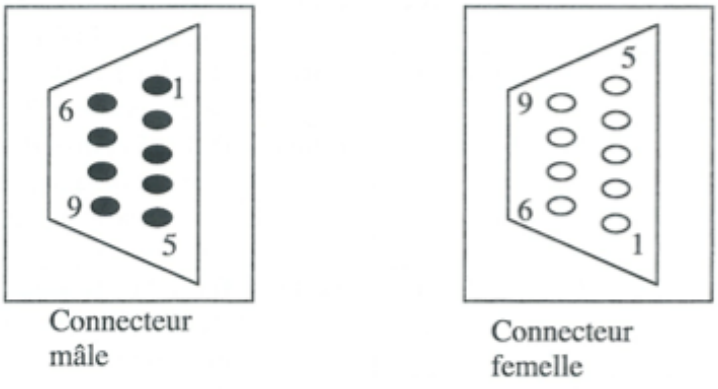


Figure A-1 – Numérotation des broches/douilles d'une interface RS232

A-3.3 Repérage des broches ou des douilles

Chaque connexion (broche ou douille) possède un nom spécifique qui ne dépend pas du type de l'appareil (DTE ou DCE). Cependant, la nomenclature est historiquement adaptée à un DTE.

Nom	**N° SUB-D9**	**N° SUB-D25**	**Sens pour un DTE**	**Sens pour un DCE**	**Rôle**
TX	3	2	Sortie	Entrée	**Transmitted Data** – Transmet les données du DTE vers le DCE
RX	2	3	Entrée	Sortie	**Received Data** – Transmet les données du DCE vers le DTE
DTR	4	20	Sortie	Entrée	**Data Terminal Ready** – Utilisé le plus souvent pour signaler au DCE que le DTE a été mis sous tension et qu'il est prêt à fonctionner
DSR	6	6	Entrée	Sortie	**Data Set Ready** – Signale au DTE que le DCE a été mis sous tension et qu'il est prêt à fonctionner
RTS	7	4	Sortie	Entrée	**Request To Send** – Utilisé par le DTE pour demander s'il peut émettre
CTS	8	5	Entrée	Sortie	**Clear To Send** – Ouverture de transmission (prêt à recevoir !)
DCD	1	8	Entrée	Sortie	**Data Carrier Detect** – Détection de message de données (liaison établie). Sur un DTE, est très fréquemment utilisé pour interdire les réceptions de données
RI	9	22	Entrée	Sortie	**Ring Indicator** – Indicateur de communication (demande de communication du DCE)
Gnd	5	7			Masse

— Noter l'inversion des signaux RX et TX entre le SUB-D25 et le SUB-D9 et la règle incontournable qui consiste à relier les masses des deux interfaces entre elles.

— Comme indiqué précédemment, la norme a été prévue, historiquement, pour connecter un DTE à un DCE (liaison DTE/DCE). Dans ce cas, la situation est extrêmement simple : pour deux appareils équipés de connecteurs SUB-D9, le "câble" RS232 reliant les deux interfaces doit contenir 9 fils (cas d'un SUB-D9), le fil i reliant la connexion i du DTE à la connexion i du DCE.

— Pour la connexion DTE/DTE, par exemple, le tableau précédent montre qu'il est aberrant de conserver le connecteur précédent (il réalise en effet une liaison entre deux entrées ou entre deux sorties !).

— **Comment reconnaître un DTE d'un DCE ?** La norme exige que le transmetteur doit

être **toujours** à une tension **négative** (1 logique) lorsque les données ne sont **pas** effectivement transmises. Comme les données doivent être transmises soit sur le contact 2 soit sur le contact 3 suivant le sexe de l'appareil (pour un SUB-D9 **et** pour un DTE la transmission se fait par la patte 3), on peut donc, rien qu'en testant ces deux contacts (au moyen d'un voltmètre) se rendre compte si l'appareil en essai est un DTE ou un DCE.

N.B. : Si une interface est équipée d'un connecteur SUB-D25, les opérations principales utiliseront les 9 contacts indiqués dans le tableau ci-dessus. Les autres contacts ne joueront qu'un rôle secondaire voire facultatif défini par le fabricant.

A-3.4 Niveaux logiques

La norme spécifie des niveaux logiques **bipolaires**. Ceci signifie que le signe (positif ou négatif) de la tension détermine son niveau logique. Une tension positive à l'interface représente un 0 tandis qu'une tension négative représente un 1 (logique dite "inversée").

Pour garantir l'état 0, **un contact de sortie** doit établir une tension comprise +5 et +15 volts. De même, un état garanti doit se situer entre −5 et −15 volts. Dans la plage intermédiaire, appelée **région de transition**, les niveaux logiques ne sont pas définis : ceci signifie que toute tension de sortie comprise entre +5 et −5 volts risque d'être interprété soit comme 0 soit comme 1 (résultat aléatoire).

Pour les entrées, la seule différence est celle de la largeur de la zone de transition. La zone logique non définie ne s'étend que sur 6 volts (de −3 à +3). Cette différence qui pourrait paraître accessoire est extrêmement importante. Pour éviter toute confusion, il suffit d'adopter la convention suivante :

— Les **entrées** sont **validées** quand elles sont positives, **invalidées** quand elles sont négatives.
— Les **sorties** sont **actives** lorsqu'elles sont positives, **inhibées** lorsqu'elles sont négatives.

N.B. : Commes les UART sont alimentés par la même tension +5 volts que les circuits intégrés de l'ordinateur, il existe entre le connecteur et l'UART un circuit intégré (l'émetteur de ligne RS232C). Ce CI convertit les tensions de sortie de l'UART en valeurs requises par la norme tandis qu'un autre (le récepteur de ligne RS 232C) convertit les tensions RS232C en niveaux convenant aux circuits de l'UART. Ce circuit est aussi appelé **translateur** ou **driver** de niveau (exemple : le CI 1489).

A-3.5 Communication & trame RS232

Les caractéristiques importantes concernant la communication sont la vitesse de transfert, le nombre de bits de données, le nombre de bits de stop et la parité. **Deux interfaces ou ports ne pourront communiquer que si ces paramètres coïncident.**

— **Baud rate** : la durée de transmission d'un bit est normalisée. La vitesse de transfert exprimée en quantum d'information (ici des bits) par seconde peut être choisie de 50 à 19200 bauds avec comme valeurs intermédiaires 75, 110, 150, 300, 600, 1200, 2400, 4800, et 9600.
— **Data bits** : l'information est envoyée sous la forme d'un paquet de bits (un octet généralement, c'est-à-dire 8 bits).
— **Stop bits** : utilisés pour indiquer la fin d'un paquet d'informations.
— **Parity** : un moyen (non infaillible) pour contrôler la qualité de la transmission d'un paquet.

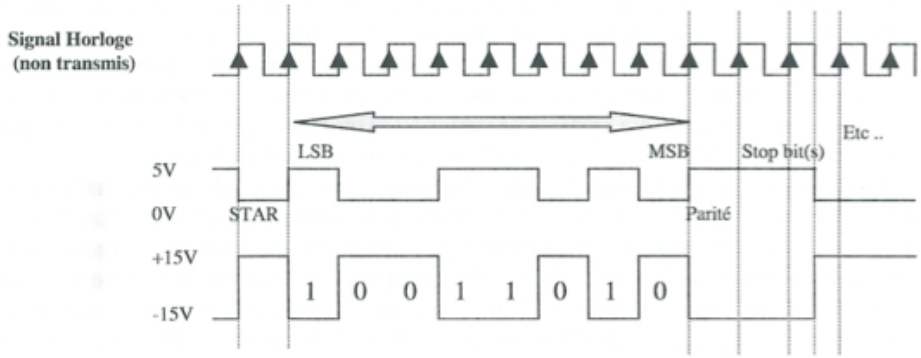


Figure A-2 – Trame

Transmettre une donnée en mode série impose l'utilisation d'une structure de **trame**. Chaque bit envoyé définit une durée. Cette durée est le temps de maintien de chaque bit et correspond à **une** période d'horloge. Les explications de la trame représentée dans la figure A-2 sont les suivantes :

— L'état initial correspond à l'état logique 1 qui est aussi l'état de repos de la ligne.
— A un instant donné, un bit dit de "start" est émis ; il correspond à un état logique 0.
— Le bit de start est suivi des bits codant la donnée en commençant par le bit de poids le plus faible (LSB ou Least Significant Bit).
— Le bit de poids le plus fort est alors, **éventuellement**, suivi d'un bit de parité.

Bit de parité ? *La présence de ce bit permet de vérifier* **sommairement** *l'intégrité du transfert de données. Le principe est simple : le bit de parité est calculé par l'émetteur à partir du champ de données à transmettre. Le récepteur effectue sur le champ reçu un calcul analogue et compare son résultat avec le contenu du bit de parité. Toute différence traduit une altération de l'information transmise. On appelle* **parité paire** *l'opération dans laquelle l'émetteur place un 1 ou un 0 dans le bit de parité, pour que le mot formé par l'ensemble des bits (données + bit de parité) soit constitué par un nombre pair de 1 logique. La* **parité impaire** *résulte du même procédé, mais dans ce cas, le bit de parité sera évalué de telle manière que le nombre de 1 logique soit impair.*

— Enfin, on transmet **1** ou **1,5** ou **2** bits de stop. L'un des intérêts de faire varier la durée de stop (**une**, **une et demie** ou **deux** périodes d'horloge), est de laisser au récepteur davantage de temps pour être à nouveau prêt à recevoir une trame. Ce délai est impératif car les données sont envoyées à un rythme imposé par l'émetteur. Etant donné que chaque dispositif a sa propre horloge, il est possible que l'émetteur et le récepteur soient légèrement asynchrones. Les bits de stop n'indiquent donc pas seulement la fin de la transmission, mais permettent aussi de générer un délai qui permet de conserver l'intégrité de la transmission même en présence d'un écart (au maximum 5%) entre les fréquences des deux horloges.

A-3.6 Tampon

Un ordinateur peut transmettre des caractères beaucoup plus vite que le dispositif périphérique récepteur ne peut les traiter. Dans le cas d'une imprimante, les caractères arrivent à une vitesse importante et ne peuvent être imprimés immédiatement. Les caractères doivent donc être stockés

quelque part, dans l'ordre de leur arrivée, en attendant leur tour d'être imprimés. Ce lieu de stockage, appelé **tampon** ou **buffer**, est localisé sur l'imprimante. Les données étant reçues par l'imprimante cette mémoire est appelée **tampon d'entrée** (un appareil pouvant, en général, émettre ou recevoir possède donc un tampon d'entrée et un **tampon de sortie**).
Comme les caractères sont mis dans le tampon beaucoup plus vite qu'ils ne sont retirés pour être imprimés, il va tôt ou tard arriver un moment où il n'y aura pas de place pour les caractères qui se présentent. Une fois le tampon d'imprimante plein, tout nouveau caractère sera perdu. Ce cas est appelé **débordement de tampon**. Pour éviter de perdre des données, le récepteur surveille le niveau de remplissage de son tampon d'entrée. Quand le tampon est plein à environ 90%, l'imprimante envoie un signal (matériel ou logiciel) à destination du PC pour qu'il stoppe provisoirement le flux de données. Le débit restera suspendu tant que le tampon d'entrée ne sera pas vidé à 90%. A cet instant le récepteur demandera d'autres données.

A-3.7 Protocole de communication

Le paragraphe précédent montre que, lors d'une communication, nous sommes confrontés à un problème de régulation de données. Dans un premier cas de figure, la communication est parfaitement maîtrisée et aucun artifice n'est nécessaire. Dans un deuxième cas, le récepteur peut être occupé à d'autres tâches et doit pouvoir réguler lui-même le flux de données.
Le protocole de communication ou **handshaking** indique comment le flot de données passant dans l'interface est régulé et commandé. Cette gestion peut être réalisée en appliquant des niveaux de tensions adaptés sur certaines broches (**handshaking matériel**) ou en insérant dans le flot de donnée des "marques" spéciales (**handshaking logiciel**).

Handshaking matériel

Le chronogramme suivant montre la chronologie et le déroulement du mécanisme d'une communication :

1	L'émetteur demande si le récepteur peut recevoir des données
2	Le récepteur répond, oui je peux recevoir des données
3	L'émetteur demande s'il peut transmettre des données
4	Le récepteur rṕond, oui je suis prêt
Data	Transmission de la trame de données Une fois la donnée transmise, on remet les signaux dans l'état initial tout en accusant réception des informations
5	Fin de transmission
6	OK fin de transmission
7	Fin de requête
8	OK protocole terminé

Handshaking logiciel

L'émetteur transmet ses données au récepteur. Lorsque le récepteur est "plein" (tampon d'entrée à 90% de sa valeur maximale), il émet vers l'émetteur un **Xoff** (code ASCII non affichable désignant Ctrl-S ou 19 en décimal). L'émetteur à la réception de ce Xoff, arrête d'émettre les données et attend un **Xon** (code ASCII non affichable ou Ctrl-Q ou 17 en décimal) de la part du récepteur

pour reprendre l'émission.

N.B. : bien entendu, il existe plusieurs façons de réaliser un handshaking matériel ou logiciel.

A-3.8 Aspects logiciels

Une fois que le bon "câble" RS-232 relie l'interface RS232 du PC à l'interface RS232 de l'instrument, il faut aborder l'aspect programmation. Mentionnons tout d'abord que le microprocesseur de l'ordinateur alloue à chaque port série (et plus généralement à chaque périphérique) une zone mémoire. Chaque octet d'une zone est repéré par une adresse et son contenu peut être modifié ou lu. Etant donné que certains octets sont associés aux registres de l'UART (cf Introduction), on devine comment le microprocesseur agit sur l'interface. Il est à noter que les adresses réservées n'ont pas changé depuis l'avènement des PC et que les ports série (s'ils existent) sont désignés par le Gestionnaire de périphériques par le diminutif COM suivi de leur numéro d'ordre (1 à 4 maximum).

N.B. : la désignation du port dépend souvent de l'application gérant la liaison.

Pour "piloter" un appareil deux aspects sont à examiner : les commandes compréhensibles par l'appareil et le logiciel qui a envoyé ces commandes et qui assure simultanément la gestion de l'interface.

Commandes

Si l'appareil est conforme à la norme IEEE488 (cf. documentation sur le GPIB), il reconnaîtra un ensemble de commandes courantes (exemple *RST). Ces commandes permettent d'exécuter des fonction telles que la réinitialisation, l'autotest et des opérations d'état (certaines instructions accèdent à des registres dont le contenu décrit ou fixe l'état de l'appareil). Ces commandes "communes" commencent toujours par un astérisque (*), ont un nom de trois, quatre ou cinq caractères et peuvent comprendre un ou plusieurs paramètres, y compris le point d'interrogation.
Le jeu de commandes précédent étant insuffisant, il est complété par un jeu de commandes "plus ou moins" spécifique à l'appareil. Par exemple les constructeurs d'appareils adhérant au consortium "SCPI" proposent pour des instruments **de fonctionnalité identique**, des commandes communes (appelées commandes SCPI confirmées), tout en les complétant par des commandes spécifiques appelées plus généralement commandes "propriétaires" (sous-entendu du constructeur).

Logiciel

De nombreux langages de programmation sont utilisés pour gérer une communication (RS232 ou autres) entre appareils. On distingue deux catégories :

- **langages "textuels"** où le programme est une succession de lignes de codes, par exemple BASIC et dérivés (QBASIC, VisualBASIC, HPBASIC, ...), C et dérivés (C++, LabWindowsCVI, ...), PASCAL et dérivés (TurboPascal, ...).
- **langages "graphiques"** où le programme se présente sous la forme d'icônes matérialisant des blocs fonctionnels reliés par des fils matérialisant les connexions et le passage du flux de données, par exemple LabVIEW, HPVee, Igor.

A-4 Code ASCII et caractères non affichés

La communication entre le PC et l'appareil est une communication basée "messages", c'est-à-dire que l'utilisateur écrit, indépendamment du logiciel utilisé, un message à partir d'un jeu (ou table) de caractères. Un code très répandu est le **code ASCII**. Chaque caractère est codé sous 8 bits (ASCII dit "étendu") et que certains vous sont familiers : les caractères de l'alphabet (les majuscules étant parfaitement distinguées des minuscules), la numérotation décimale, quelques signes de ponctuation. Ces signes familiers sont des caractères dits "affichables" (en enfonçant la touche A sur un clavier d'ordinateur, vous voyez apparaître A sur l'écran). A l'opposé, d'autres caractères ne sont pas affichables au sens strict du terme. Par exemple, le caractère noté LF (LineFeed) est "généré" en appuyant sur la touche Entrée d'un clavier. Ce caractère n'est pas affichable, mais représente une commande sensée provoquer deux actions sur votre écran d'ordinateur : décaler le curseur et tous les caractères qui le suivent une lignes d'écran en dessous **puis** glisser l'ensemble complètement à gauche de la ligne.

Dans de nombreux cas, une commande basée messages commence par une suite de caractères ASCII affichables **suivie** d'un ou plusieurs caractères ASCII non affichables. Ces caractères non affichables sont utilisés pour indiquer la fin du message (affichable) : ils sont **absolument** indispensables. Leur absence peut souvent expliquer un problème de communication entre appareils. Les caractères non affichables à utiliser dépendent de l'appareil (consulter sa fiche technique et réaliser des tests).

Remarque importante : sous LabVIEW, l'option "Affichage des codes \" sélectionnée par un click droit à partir d'une commande de chaîne de caractères permet d'informer le logiciel qu'il doit interpréter les caractères suivants immédiatement le backslash (\) comme un code associé à des caractères ASCII non affichables. La table suivante montre comment le langage graphique G de LabVIEW interprète ces codes.

Code	**Interprétation G**
\00 – \FF	Valeur hexadécimale d'un octet
\b	Backspace (ASCII BS, équivalent à \08)
\f	Form feed (ASCII FF, équivalent à \0C)
\n	Line feed (ASCII LF, équivalent à \0A)
\r	Carriage return (ASCII CR, équivalent à \0D)
\t	Tab (ASCII HT, équivalent à \09)
\s	Space (ASCII SP, équivalent à \20)
\\	Backslash (ASCII \, équivalent à \5C)

Différents types de liaisons courantes dans un ordinateur de type PC

Nom	**Type**	**Bits/CLK**	**Liaison**	**Bus**	**Débit max**	**Matériels (quelques exemples)**
RS232	Série	1	×		115 kB/s	Souris, appareils de mesure
Centronics	Paral	8	×		500 ko/s	Imprimante, scanner
RS485	Série	1		×	460 kb/s	Appareils de mesure
IDE	Paral	16	×		17 Mo/s	Disque dur, CD-ROM (mémoire de masse)
Ultra640SCSI-3	Paral	16		×	640 Mo/s	Disque dur, scanner
USB	Série	1		×	12 Mo/s	Souris, clavier, scanner
I2C	Série	1		×	3,4 Mb/s	Connexions entre CI, télévision
GPIB/IEEE488	Paral	8		×	1 Mo/s	Appareils de mesure
MIDI	Série	1	×		31,5 kb/s	Synthétiseurs audio
CAN	Série	1		×	1 Mo/s	Automobile

A-5 Interface GPIB

Bibliographie :
— Catalogue National Instruments 2001
— "Le bus IEEE-488" de Christophe Basso (Dunod)

A-5.1 Origine du bus GPIB

En 1965, la firme Hewlett-Packard a conçu le Bus d'Interface Hewlett-Packard (HPIB) pour connecter ses instruments programmables à ses ordinateurs. Grâce à son débit élevé (1 Mo/s), cette interface est devenue rapidement populaire. Plus tard acceptée comme nouveau standard référencé sous IEEE488-1975, elle a évolué pour devenir le standard ANSI/IEEE 488.1-1987.

Aujourd'hui, le nom de GPIB (General Purpose Interface Bus) est beaucoup plus répandu que HPIB. La norme ANSI/IEEE488.2-1987 a renforcée le standard original en définissant de manière précise comment les contrôleurs et les instruments communiquent. Les Commandes Standards pour Instruments Programmables (SCPI) utilisent les structures de commandes définies dans l'IEEE488.2 et créent un jeu de commandes communes à utiliser pour un instrument SCPI.

A-5.2 Types de messages GPIB

Les dispositifs GPIB communiquent avec d'autres dispositifs GPIB en envoyant *(a)* des messages dépendant de l'instrument à atteindre, et *(b)* des messages d'interface au travers d'un bu spécifique :
— **les messages dépendant de l'appareil à atteindre**, souvent appelés données ou messages de données, contiennent des informations spécifiques à l'appareil telles que des instructions de programmation, des résultats de mesure, des indicateurs d'état de dispositifs et des fichiers de données.
— **les messages d'interfaces** gèrent le bus. Habituellement appelés commandes ou messages de commandes, les messages d'interfaces réalisent les fonctions telles que l'initialisation du bus, l'adressage des dispositifs et le mode de programmation des appareils (distant ou local).

N.B. : une donnée n'intéressant pas forcément tous les appareils, il est possible d'empêcher les appareils non concernés par le message de participer à la communication (notion de désadressage).

A-5.3 Emetteurs, récepteurs et contrôleurs

Les dispositifs GPIB peuvent être des émetteurs (Talkers), des Auditeurs (listeners) ou Récepteurs et/ou des Contrôleurs (Controllers). Un émetteur envoie des messages de données à un ou plusieurs Auditeurs qui reçoivent les données. Le Contrôleur gère le flux d'informations circulant sur le GPIB en envoyant des commandes à tous les appareils. Un voltmètre numérique par exemple, peut être soit un émetteur, soit un auditeur.

Le GPIB est comme un bus d'ordinateur ordinaire excepté que les circuits de l'ordinateur sont interconnectés sur un même support (la carte "mère"). Le GPIB présente des dispositifs autonomes interconnectés par des câbles normalisés.

Le rôle du contrôleur GPIB est de définir parmi les appareils connectés au bus ceux qui écoutent et celui qui parle. Certaines configurations GPIB ne nécessite pas de Contrôleur. Par exemple, la

connexion entre un dispositif qui est toujours un émetteur et un dispositif qui est un récepteur. Une fois les rôles clairement établis, l'information arrive au dispositif intéressé. L'envoi des information à tel ou tel équipement s'effectue par le biais d'une adresse qui circule sur le bus. Un ordinateur équipé d'une interface GPIB représente le parfait exemple du Contrôleur.

A-5.4 Contrôleur actif et système de contrôle

Un système peut utiliser plusieurs ordinateurs. Bien que chacun puisse se comporter comme un Contrôleur, un seul est actif à un instant donné. Les autres peuvent être alors des émetteurs ou des récepteurs. Le contrôle actif peut être transféré d'un contrôleur à l'autre, mais le tout reste sous la supervision d'un contrôleur général.

A-5.5 Adressage sur le bus

Chaque instrument possède, au minimum, une adresse d'émetteur ou de récepteur. On peut sélectionner individuellement l'adresse de chaque instrument (émetteur, récepteur ou bien les deux à tour de rôle) de plusieurs façons :

- **matériel** : en basculant des interrupteurs miniatures placés en général à l'arrière de l'équipement.
- **logiciel** : en entrant un numéro de 0 à 30 inclus via les touches du panneau frontal de l'appareil.

N.B. : il faut parfois éteindre et rallumer l'appareil pour valider cette séquence, surtout dans le premier cas.

A-5.6 Lignes et signaux GPIB

Les fils ou lignes du bus GPIB au nombre de 16 (+8 de masse) peuvent être regroupées selon trois catégories :

- **Lignes de données DI** : elles constituent un bus bidirectionnel, 8 bits. Les octets transitent en exploitant un code. Généralement, il s'agit du code 7 bits ASCII. Toutes les commandes et la plupart des données utilisent ce code ASCII 7-bits. Le signal DI08 est soit inutilisé, soit utilisé pour le test de parité.

- **lignes d'handshaking ou de contrôle** : au nombre de trois, elles garantissent la fiabilité du transfert des données circulant sur le bus :
  - **NRFD** (Not Ready For Data) : indique quand un dispositif est prêt pour recevoir un octet. Tous les dispositifs qui peuvent recevoir des commandes peuvent piloter (fixer le potentiel de) cette ligne.
  - **NDAC** (No Data ACcepted) : indique si un dispositif accept ou refuse l'octet. Tous les dispositifs qui peuvent recevoir des messages peuvent piloter cette ligne.
  - **DAV** (DAta Valid) : indique quand les signaux sur les lignes de données sont stables et donc valides et peuvent donc être acceptés en toute sécurité par les dispositifs. Le Contrôleur comme tout émetteur connecté sur le bus, pilote cette ligne lorsqu'il envoie des octets.

- **Lignes de gestion du bus** : cinq lignes gèrent le flux d'information au travers de l'interface :

— **ATN** (ATteNtion) : le Contrôleur fixe ATN à vrai quand il utilise les lignes de données pour envoyer des commandes et fixe ATN à faux quand un Emetteur peut envoyer des messages de données.
— **IFC** (InterFace Clear) : cette ligne uniquement pilotée par le contrôleur actif permet à ce dernier d'arrêter à tout instant (de manière asynchrone) l'opération en cours sur le bus.
— **REN** (Remote ENable) : son niveau exclusivement géré par le contrôleur en charge, force chaque appareil concerné à être piloté par le bus. A l'état bas (vrai), chaque récepteur adressé et capable d'être piloté passe en mode remote. Lorsque le signal REN repasse à un niveau haut (faux), l'appareil concerné retourne en mode local et l'utilisateur peut alors agir sur la configuration de l'appareil par les commandes de son panneau frontal.
— **SRQ** (Service ReQuest) : tout dispositif peut piloter SRQ pour une requête de service asynchrone auprès du Contrôleur (demande d'interruption immédiate liée à un problème de fonctionnement par exemple). Pour déterminer l'équipement qui réclame son attention, le contrôleur effectue un **polling** (de l'anglais sonder) : il interroge individuellement chaque appareil (serial polling) ou tous les appareils à la fois (parallel polling).
— **EOI** (End Or Identify) : EOI a deux fonctionnalités : *(a)* l'émetteur pilote EOI pour signaler la fin d'une chaîne de message à l'aide d'un caractère particulier ; *(b)* piloté à l'état bas conjointement avec la ligne ATN, le signal EOI autorise le contrôleur à effectuer des parallel polling.

A-5.7 Caractéristiques électriques et physiques

Les dispositifs sont habituellement reliés connectés par un câble blindé à 24 fils équipé de connecteurs dont les extrémités sont normalisées. On peut relier les dispositifs selon une **configuration linéaire** ou une **configuration en étoile** ou un combinaison des deux. Pour des connexions spéciales, un câble adaptateur non standard devra être utilisé.
Le GPIB utilise une logique négative à niveaux TTL. Quand DAV est vraie, par exemple, cela correspond à un niveaux TTL bas (<0.8 V) et quand DAV est haut, cela correspond à un niveau TTL haut (>2.0 V)

A-5.8 Spécifications

Pour atteindre les taux de transfert pour lesquels le GPIB a été conçu, la distance physique entre dispositifs et le nombre de dispositifs interconnectés sont limités. Les restrictions suivantes sont typiques pour une opération normale :
— Une séparation maximale de 4m entre deux dispositifs et une séparation moyenne de 2m sur tout le bus.
— Une longueur maximale de câble égale à 20m.
— Pas plus de 15 appareils connectés sur le bus avec au moins deux tiers des appareils allumés.

A-6 Présentation sommaire des acronymes VISA, SCPI, IVI

VISA est l'acronyme de Virtual Instrument Software Architecture. La bibliothèque VISA est un ensemble de fonctionnalités qui permet de dialoguer (I/O) avec un instrument. Elle respecte un cahier des charges précis (spécifications VISA) élaboré par l'organisme **VXIplug&play System Alliance**.

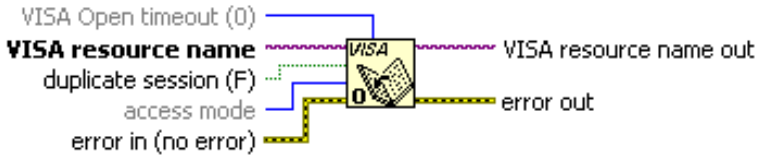
Une fonctionnalité de cette bibliothèque s'appelle **viOpen**. Le programmeur souhaitant la créer doit obligatoirement définir les paramètres d'entrée **sesn, rsrcName, accessMode, timeout** et le paramètre de sortie **vi** (à ne pas confondre avec un VI LabVIEW). En utilisant un langage textuel, la syntaxe de cette commande est donc de la forme :

viOpen(sesn, rsrcName, accessMode, timeout, vi)

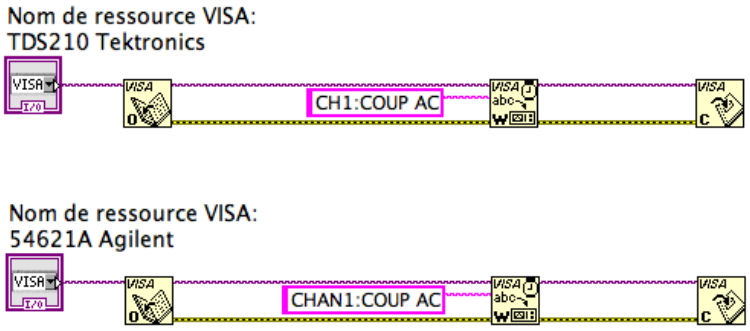
Cette routine ouvre une session (chemin de communication unique) entre un logiciel (LabVIEW par exemple) et une ressource (ensemble de fonctionnalités permettant de contrôler un dispositif par exemple).

Chaque entreprise, qui construit un appareil ou/et qui propose un logiciel permettant le contrôle d'instruments, fournit généralement sa bibliothèque VISA. Dans la bibliothèque VISA d'Agilent ou dans celle de National Instruments, vous trouverez le **viOpen** mentionné précédemment.

On notera que l'entreprise ne fournit pas obligatoirement toutes les fonctionnalités définies par l'organisme VXIplug&play System Alliance. De plus, l'aspect des fonctions voire le nombre de paramètres peuvent différer d'un constructeur à l'autre. Par exemple le viOpen de LabVIEW (National Instruments) se présente sous la forme d'une icône graphique (cf. ci-dessous) qui propose, outre les quatre paramètres d'entrée obligatoires, un paramètre d'entrée supplémentaire (entrée d'erreur). L'icône présente aussi un paramètre de sortie supplémentaire (sortie d'erreur) non exigé par la norme VISA.



L'intérêt de la bibliothèque VISA est de proposer un ensemble de fonctionnalités indépendantes des constructeurs et donc des matériels. Ceci permet de faciliter la programmation ou l'adaptation d'un programme ùn matériel différent (généralement, il suffit de modifier la syntaxe des commandes). Pour illustrer ceci, on présente ci-dessous un programme LabVIEW qui utilise trois fonctions VISA basiques : **viOpen, viWrite** et **viClose**. Ce VI permet de sélectionner le mode de couplage (ici AC) des voies d'entrée n°1 de deux oscilloscopes différents. On constate que la programmation est identique excepté la syntaxe de la commande (CH1 : COUP AC pour le TDS210 et CHAN1 :COUP AC pour le modèle 54621A).



N.B. : dans le cas d'oscilloscopes se conformant à la norme SCPI (Standard Commands for Programmable Instruments), la syntaxe d'une commande SCPI certifiée aurait été identique. Plus d'infos à l'URL `http://www.scpiconsortium.org/`

N.B. : depuis 2001, un nouveau consortium (fondation **IVI** pour Interchangeable Virtual Instruments) a été officiellement constitué. Il a absorbé VXIplug&play System Alliance et promeut les spécifications pour les instruments qui simplifient l'interchangeabilité, l'accroissement des performances et la réduction des coûts des programmes de développement et de maintenance.

A-7 Exemples de commande SCPI : programmation du GBF HP33120A

Notes :

— Un signal dit arbitraire doit avoir un nom. Un nom de signal arbitraire peut contenir jusqu'à huit caractères. Le premier doit être une lettre. Les espaces ne sont pas autorisés. Le constructeur a déjà enregistré dans la mémoire du GBF cinq signaux ineffaçables et originaux appelés SINC, NEG_RAMP, EXP_RISE, EXP_FALL et CARDIAC. Ces noms sont donc réservés et ne doivent pas être utilisés pour nommer votre propre signal (vous pouvez définir 4 signaux différents au maximum).

— Les commandes suivantes ont été écrites dans des cases afin d'en améliorer leur lisibilité et de montrer sans ambiguïté la présence d'espaces. Il y a exactement **un seul** caractère par case.

— Les commandes "basées message" seront envoyées au GBF **en les regroupant dans une chaîne de caractères unique** reliée à l'entrée d'une fonction VISA Write implémentée par LabVIEW.

Commande 1 : mettre le GBF dans un état de fonctionnement standard

*	R	S	T

Commande 2 : mettre le GBF en mode remote

S	Y	S	T	:	R	E	M

Commande 3 : effacer les signaux arbitraires éventuels

D	A	T	A	:	D	E	L	:	A	L	L

Commande 4 : télécharger dans la mémoire effaçable appelée VOLATILE les valeurs du signal. On donne ici un exemple de téléchargement de dix valeurs à virgule flottante représentant un signal (entre 8 et 16000 points par signal). Ces valeurs doivent être **impérativement** comprises entre -1 et $+1$. Généralement, la commande suivante a été créée au préalable en utilisant une fonction mathématique générant des nombres qui sont convertis au format ASCII puis concaténés à la chaîne de caractères DATA VOLATILE.

D	A	T	A		V	O	L	A	T	I	L	E	,		1	,		0	,		.	5	,

	.	2	,		0	,		.	2	,		.	5	,		0	,		.	3	,		0

Commande 5 : copier la mémoire effaçable VOLATILE dans une mémoire non effaçable appelée MYSIG2

D	A	T	A	:	C	O	P	Y		M	Y	S	I	G	2

Commande 6 : sélectionner le signal que vous souhaitez voir apparaître en sortie du GBF (ex : MYSIG2)

F	U	N	C	:	U	S	E	R		M	Y	S	I	G	2

Commande 7 : envoyer ce signal numérique sur le convertisseur numérique-analogique (CNA) du GBF

F	U	N	C	:	S	H	A	P		U	S	E	R

Commande 8 : fixer la fréquence du signal à 5 kHz par exemple

F	R	E	Q		5	.	0	E	+	3

Commande 9 : fixer l'amplitude crête-à-crête à 3 V par exemple

V	O	L	T		3	.	0