A l'occasion de NI Days 2002, il nous semblait important de donner quelques points de repère et d'expliquer la démarche de National Instruments vis-à-vis du temps réel. Il n'est pas question ici de rentrer dans les détails techniques (ceci fait l'objet d'autres présentations) mais plutôt de donner les clés qui permettent de comprendre la stratégie de NI, les produits actuels et les orientations futures.

Questions d'utilisateurs

- ~~Je souhaite acquérir en temps réel et en continu à 100 000 points par seconde~~
- Je souhaite asservir un moteur à 1000 pts/s
- Je souhaite garantir l'exécution de mon application quelle que soit l'activité de ma station
- Je souhaite garantir les temps de réponse de mon application (trigger, événements…)

ni.com/france

NATIONAL INSTRUMENTS

Puisque National Instruments est essentiellement tourné vers les clients/utilisateurs (vente en direct, services, support localisé, antennes commerciales régionales…), commençons par étudier quelques unes des demandes que nous avons reçu. Historiquement, elles sont issues d'ingénieurs qui, ayant fait le choix du PC, ayant prouvé tous les avantages qu'il y avait à bâtir une solution de test et mesure sur cette plate-forme, se sont naturellement tournés vers NI pour faire part de leur envie d'aller plus loin.

En ce qui concerne la première requête, notons que l'offre matérielle et logicielle actuelle de NI permet de répondre. Les cartes disposent en effet de toute l'électronique nécessaire (mémoire, buffer, transfert haute vitesse etc.) et les drivers facilitent la mise en œuvre de ce type d'application.

Pour le reste, remarquez que les questions posées ne sont pas extravagantes mais plutôt frappées au coin du bon sens. Par exemple, on sait qu'une carte d'acquisition est capable d'acquérir ou de générer très rapidement des données. Ceci dit, dès que l'utilisateur s'intéresse aux application de contrôle-commande (acquisition, traitement, génération, le tout synchronisé et en continu), tout se passe comme si NI ne mettait pas à la disposition des développeurs les moyens de répondre facilement au problème.

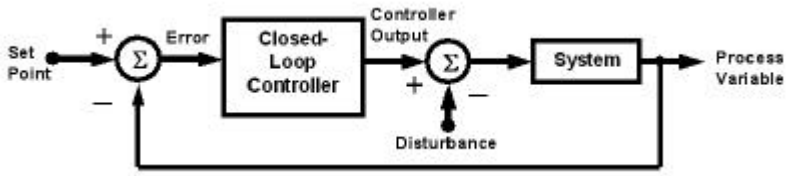
De plus, même si les systèmes d'exploitation se sont beaucoup améliorés du point de vue de la stabilité, il n'empêche que tous ceux qui utilisent un PC ont presque toujours une petite voix intérieure qui semble dire : « Oui, mais qu'est-ce qui se passe si le système d'exploitation (pas l'application) se bloque ? ». Là aussi, mais c'est à la limite plus compréhensible, NI et tous les fournisseurs d'outils de test et mesure sur PC ne semblent pas avoir de réponse.

Attention, il faut bien garder en tête que cela concerne non seulement les fournisseurs d'outils de développement mais aussi tous ceux qui par exemple enfouissent un système d'exploitation classique (type Windows ou autre) dans un instrument de mesure ou dans une boîte noire.

Notez aussi que la réponse qui consisterait à dire « oui mais l'instrument ou la boîte noire ne fait que de la mesure, et à ce titre les risques sont limités », est plus ou moins recevable. En effet, pour peu que sur le système en question il y ait un port Ethernet, on réalise que l'outil de mesure va certes mesurer, mais aussi recevoir des trames TCP/IP qu'il faudra traiter. Autrement dit, même installé dans un « instrument » un OS classique reste un OS classique et les problèmes sont les mêmes que sur un PC. Il faut en être conscient et accepter le fait que certaines applications puissent s'en accommoder et d'autres pas.

La dernière classe de question intéresse au premier chef les ingénieurs du test et de la mesure puisqu'il s'agit de trouver un moyen de garantir des temps de réponse après déclenchement. Attention, cela ne signifie pas qu'actuellement, une carte oscilloscope PXI qui reçoit un signal de trigger ne va pas immédiatement prendre les mesures. Non, on s'intéresse par exemple aux applications susceptibles de réaliser une action sur une carte DAQ après qu'un mot particulier ait été reçu sur un port série. On le voit, il s'agit des applications pour lesquelles une action doit être effectuée après qu'un ordre ait été reçu. Pensez par exemple à la prise en compte d'un signal d'arrêt d'urgence. Là aussi, NI (comme les autres) ne peut apporter de réponse ferme et définitive puisque « cela dépend de la charge de votre système d'exploitation ».

Il faut un système temps réel !



ni.com/france

NATIONAL INSTRUMENTS

On le voit et on le comprend, sur un système d'exploitation classique (32 bits, mutithread de type Linux, Mac Os X ou Windows), il n'est pas toujours évident d'apporter des réponses satisfaisantes à des questions simples. Ceci dit, une solution existe et elle consiste à utiliser un système d'exploitation temps réel.

Toutefois, avant de changer de système d'exploitation, autrement dit de casser un standard, il va falloir étudier les «bonnes» raisons d'un tel changement, étudier les différentes possibilités, voir si il n'y a pas moyen de minimiser les risques associés à un tel changement. On peut penser par exemple aux risques liés à l'utilisation d'autres méthodes de développement, aux retards dans le codage qui peuvent en résulter et aux méthodes de test et de validation de l'application qui vont nécessairement évoluer si l'utilisateur change de système d'exploitation et d'outils de développement.

Ah oui… un système embarqué

- Non !
- Un système embarqué n'est pas nécessairement un système temps réel
- Vice et versa
- Attention à la notion de système enfoui

ni.com/france

NATIONAL INSTRUMENTS

Ici quelques rappels semblent nécessaires. Un système embarqué n'est pas nécessairement un système temps réel. De même (mais c'est généralement mieux compris) un système temps réel n'est pas nécessairement un système embarqué. Précisons un peu les choses, séquence décryptage…

Les systèmes embarqués : il peut s'agir d'un PC complet sur une carte qui au sein d'un matériel plus important prend en charge la gestion d'un écran ou la gestion d'un ensemble de disques durs. D'autres exemples existent tels que les terminaux Internet ou bien encore certains systèmes télématiques (équipement électronique de confort embarqués dans un véhicule). Notez en plus qu'ici, la notion de système embarqué est synonyme de système enfoui alors que dans nos applications de test et mesure, système embarqué signifie système de mesure dans un avion, une voiture ou une station spatiale.

Les systèmes d'exploitations embarqués : en fait la confusion tient au fait que très longtemps, les OS temps réel étaient aussi utilisés dans les systèmes embarqués. Cela est principalement dû au fait que la taille du cœur de ces systèmes est généralement très faible. Par exemple, le cœur de QNX fait 12 Ko à peu près. De plus, ces systèmes ayant une empreinte mémoire faible, les différents fournisseurs les rendent disponibles pour différents processeurs dont entre autres les processeurs utilisés dans les systèmes embarqués. Ceci étant dit, bien qu'embarqués, leurs fonctionnalités temps réel ne sont pas toujours ni nécessaires ni utilisées.

A titre d’exemple :

pSos est un système d’exploitation temps réel généralement utilisé sur des systèmes enfouis (cœur d’imprimante, gestionnaire de clavier d’oscilloscope…)

QNX et RTX sont des systèmes d’exploitation temps réel qui peuvent être embarqués et/ou utilisés sur PC

Palm OS ou Windows XPE sont des systèmes d’exploitation dédiés à l’embarqué (assistant personnel par exemple).

Un OS temps réel ?

- C’est un système d’exploitation qui garantit la réponse à un événement ou l’exécution d’une certaine tâche dans un intervalle de temps donné.
- Temps réel = déterminisme
- Temps réel != rapide

- Un OS temps réel répond aux trois questions de l’utilisateur précédent

ni.com/france

NATIONAL INSTRUMENTS

On peut maintenant se demander en quoi un système d’exploitation temps réel représente la réponse aux questions posées par les utilisateurs.

En fait, la grande spécificité d’un OS temps réel, c’est qu’il garantit la prise en compte des événements dans un intervalle de temps donné. Et alors ? Et alors, cela signifie que si un système temps réel bancaire garantit qu’il prendra en compte une transaction, au maximum, au bout d’une minute, tous les clients de la banque sont assurés que quelle que soit la charge du système, les opérations s’effectueront en temps et en heure. Si techniquement l’OS en question doit jouer dynamiquement sur les différentes priorités (suspendre par exemple l’impression de certains rapports qui sont jugés moins prioritaires que les transactions) peu importe…

Il existe de très nombreuse manières de bâtir un système d’exploitation temps réel, mais ce qu’il faut retenir, c’est que l’aspect essentiel d’un tel système est le déterminisme.

Cela signifie bien que système temps réel ne veut pas dire système rapide et à ce titre, l’exemple bancaire précédent est un bon exemple. Par contre, système temps réel rapide veut dire système temps réel plus cher. En effet, posséder un système temps réel rapide, cela signifie que l’on dispose d’un système qui garantit des temps de réponse plus courts. Techniquement il est sans doute plus évolué que les autres et il bénéficie de technologies qu’il faut bien payer.

Encore une fois, temps réel ne signifie pas rapide mais déterministe.

Voyons maintenant pourquoi un système temps réel répond aux trois questions des utilisateurs :

Question 1, boucle de régulation : je peux être assuré que ma routine de contrôle (acquisition d'un point, calcul, génération d'un point) s'exécutera toujours à la même vitesse. Si ma routine est rapide, le système pourra, changer de contexte et passer la main à une autre tâche jusqu'à ce qu'il soit nécessaire que mon application acquiert un autre point. Si la routine est trop lente, je suis cependant assuré qu'elle s'exécutera toujours à la même vitesse. Ce type de cas peut se présenter lorsque par exemple l'application doit traiter un point toute les secondes alors que la boucle de régulation s'exécute en 2 secondes. Il faut alors modifier les algorithmes afin qu'ils s'accommodent de la vitesse de boucle effective ou mettre la main au porte monnaie et investir dans un système temps réel aux performances supérieures.

Question 2, application qui s'exécute quelle que soit l'activité de la station. Généralement, dans un système d'exploitation temps réel il est possible de jouer sur les priorités des différents processus qui s'exécutent en parallèle. On peut ainsi rendre toutes les tâches du système moins prioritaires que l'application critique. Ceci peut rendre l'interface graphique moins réactive mais garantit la bonne exécution de la tâche qui nous intéresse.

Question 3, garantir les temps de réponse. C'est justement tout l'objet des OS temps réel.

Quelques unes des possibilités…

- OS 9
- QNX
- RTX
- RT Linux
- VxWorks
- Lynx OS
- PSoS
- …


ni.com/france

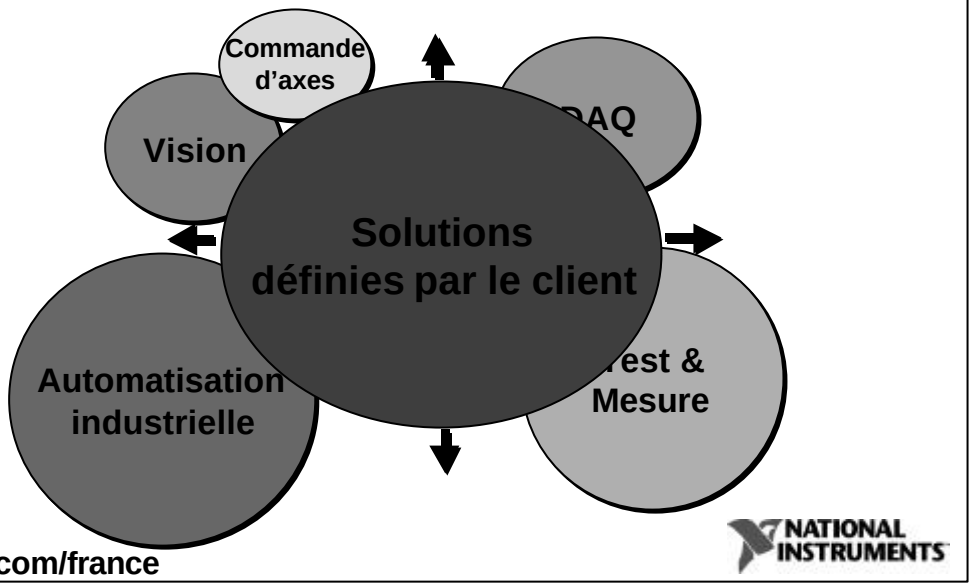
Nous avons justifié l'utilisation d'un système d'exploitation temps réel et nous avons expliqué les distinctions qui existaient entre systèmes embarqués et systèmes temps réel. Il est temps maintenant d'expliquer pas à pas la démarche de National Instruments.

Ci-dessus, on trouve quelques uns des systèmes temps réels les plus connus et ce qui frappe de prime abord, c'est la richesse de l'offre et le manque de standard établi.

Ceci dit, dans la démarche NI, l'utilisation de standards est une règle d'or et quelques exemples ne sont peut être pas inutiles. Dans le cas du GPIB, c'est une initiative de Hewlett Packard à laquelle National Instruments a tout de suite adhéré puisque c'est un moyen de contrôler de manière standard tous les instruments de tous les constructeurs. Dans le cas du VXI, NI joue pleinement le jeu et supporte pleinement ce standard (la sortie récente du kit MXI3 pour VXI en est la preuve flagrante). Ceci dit, on remarque que National Instruments possède une offre minimale en VME car le manque de standardisation du bus de fond de panier ne garantit pas des temps de mise en œuvre rapide. En ce qui concerne le PXI, même si au départ c'est une initiative NI, c'est maintenant un standard international qui se trouve sous la responsabilité de PICMG (organisme qui gère le standard CompactPCI).

En conclusion, et c'est le premier élément qui permet de comprendre la stratégie de NI, il faut trouver un moyen de répondre aux attentes des clients/utilisateurs dans le domaine du temps réel en évitant de tomber dans le piège du manque de standardisation des systèmes d'exploitation. De plus, il faut sans doute avant d'aller plus loin, calmer les ardeurs des ingénieurs R&D de NI et revenir un peu sur le métier de National Instruments afin de bien comprendre les besoins spécifiques des différents marchés supportés.

La position de NI sur ses marchés



ni.com/france

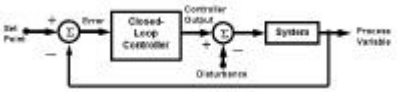
National Instruments possède une position unique en son genre… En effet, si l'on regarde attentivement les différentes lignes de produits qu'il propose et donc les marchés qu'il touche, on se rend compte que sa situation est centrale. En fait cette situation est en parfaite adéquation avec celle des utilisateurs à qui l'on demande d'intégrer dans des temps records des applications toujours plus rapides, plus précises et susceptibles d'intégrer des moyens de test et contrôle en constante évolution.

Autrement dit, il faut que la solution aux problèmes temps réel que trouvera National Instruments soit homogène avec l'esprit des lignes de produits existantes. De plus, il faudra que cette solution s'architecture autour de matériels utilisables et réutilisables dans différents type d'applications.

Ceci étant posé, on peut maintenant essayer de restreindre le champ des solutions possibles, en étudiant les principaux paramètres de l'équation. Si l'on était en train de résoudre une équation différentielle, nous dirions que dans les diapositives qui suivent, nous allons étudier les conditions aux limites…

Les types d'applications

- C'est la dimension logicielle du problème
- Régulation et contrôle
- Stimulation
- Simulation



ni.com/france

NATIONAL INSTRUMENTS

Dans les grandes classes d'applications de test, mesure et automatisation industrielle où le temps réel est nécessaire on peut distinguer :

1/ Les application de régulation et de contrôle : typiquement, sur un nombre plus ou moins important de voies l'application va acquérir des données, effectuer un certains nombre de calculs et générer des consignes. Classiquement ce sont toutes les applications de contrôle PID.

2/ La classe stimulation recouvre toutes les applications qui doivent envoyer des signaux à un système dont on veut vérifier/étudier le comportement. Par exemple, une application fait varier la température d'un habitacle de véhicule et on étudie le comportement de la climatisation. Un autre exemple concerne les applications qui après modélisation, sont chargées d'envoyer des vecteurs de test à un système ABS afin de vérifier si ce dernier répond correctement dans le temps qui lui est imparti.

3/ La classe simulation recouvre toutes les applications dont le comportement doit simuler une unité sous test. Par exemple, en reprenant les exemples précédents, l'application devrait ici, recevoir des stimulis de la part d'un banc de test et se comporter comme une climatisation ou un système ABS.

On le voit, ces applications sont différentes des applications où l'on acquiert un son pour tester l'acoustique d'une salle ou bien encore d'une application générant des vecteurs de test pour valider une carte électronique. En effet, dans ces derniers exemples, l'aspect temporel est moins critique voire totalement ignoré. Oui, c'est vrai, il faut tester rapidement les cartes électronique, mais fondamentalement, que le test prenne une seconde ou une minute ne remet pas en cause le fait que la carte soit bonne ou mauvaise. Par contre, dans les trois premiers exemples le temps (pas la vitesse) est un facteur primordial. Par exemple, il faut non seulement que le système ABS réponde correctement aux sollicitations mais aussi que la réponse se fasse dans le temps imparti.

Conséquences

- Il n'y a rien de fondamentalement différent par rapport à ce qui existe déjà
- Certes il y a le déterminisme…
- NI doit disposer d'une offre qui, dans l'esprit, est compatible avec ce qui existe déjà

La conclusion concernant les applications précédentes est que rien de fondamental (à part le déterminisme) ne les différencie de ce qui existe déjà. Ce sont des entrées, des sorties, du test… rien de révolutionnaire si ce n'est la contrainte liée au temps.

Du point de vue de la réflexion que doit conduire NI, il faut proposer une solution qui garantisse que ce qui se fait déjà sur les OS classiques soit au moins aussi facile à faire sur un OS temps réel. Cela passe très certainement par le fait que NI doit garantir que la plupart des outils déjà utilisés le seront encore en version temps réel.

Le cœur de cible !

- C’est la dimension humaine du problème
- C’est l’utilisateur qui pose les trois questions de la première diapositive
- C’est un ingénieur de test, pas un ingénieur de développement temps réel
- C’est un ingénieur qui souhaite capitaliser son savoir faire et à qui on ne peut pas demander de tout remettre en cause

Oh le joli terme marketing… En français dans le texte cela signifie tout simplement que dès le départ, dans les réflexions conduites par National Instruments, les besoins du client/utilisateur ont été au cœur des préoccupations et qu’à ce titre plusieurs contraintes importantes ont été identifiées.

On l’a vu, l’offre, en termes de systèmes d’exploitation est pléthorique. Toutefois, à y regarder de plus prêt, on réalise rapidement que la mise en œuvre et la maintenance de ces systèmes, même si elle s’est grandement améliorée, n’en reste pas moins délicate à plus d’un titre. Dans tous les cas, et si l’on suppose que la majorité des clients travaillent sous Windows ou Mac OS, on peut imaginer que le fait de devoir changer d’OS puisse poser certains problèmes. Par exemple, si on suppose qu’un OS temps réel se présente comme un Unix, on imagine que le simple fait de vouloir gérer les fichiers de son application puisse poser problème à des utilisateurs Windows.

Maintenant, du point de vue des outils de développement la situation est pire… En effet, si nombre de clients réalisent des applications LabVIEW et Measurement Studio non triviales, il n’est pas évident que de leur proposer de changer d’outils de développement soit une bonne chose. En effet, un changement d’OS associé à un changement d’outils peut paraître rédhibitoire à plus d’un développeur. On touche ici un point essentiel de la démarche et qui concerne la capitalisation du savoir-faire. Par exemple, NI s’engage déjà sur le fait qu’un code LabVIEW soit portable. Cela signifie que l’expérience acquise en LabVIEW sur Mac pourra être utilisée sur un autre développement LabVIEW sous Linux. De même, les fonctions de l’API GPIB sont disponibles sur de très nombreuses plate-formes et ont très peu changé. Cela signifie que NI a toujours fait en sorte que le savoir-faire acquis sur cette API puisse être capitalisée par le développeur. Enfin, et toujours à titre d’exemple, quand LabWindows est devenu LabWindows/CVI en passant de DOS à Windows, là aussi l’expertise acquise n’a pas été remise en cause et des outils de conversion ont permis un passage en douceur d’un système d’exploitation à l’autre.

Conséquences

- On ne peut pas demander à l'utilisateur de passer en programmation de trop bas niveau
- Il faut garantir des temps de mise en œuvre comparables avec ce qu'il connaît

L'expérience acquise par NI sur d'autres systèmes d'exploitation ainsi que les retours d'informations des utilisateurs imposent donc des contraintes supplémentaires. Ainsi, puisque l'on s'adresse à des ingénieurs de test et mesure qui sont plus intéressés par le résultat des mesures que par les moyens de mesures, il faudra appliquer la recette qui a fait le succès de NI sur d'autres plate-formes (Windows, Mac…) et proposer une solution qui isole les développeurs des détails de trop bas niveau.

La deuxième contrainte tient au fait que l'on s'adresse à des industriels qui ont des contraintes vis-à-vis des temps de mise sur le marché qui ne peuvent pas être négligés. Autrement dit, il faudra absolument que les temps de mise en œuvre des systèmes temps réel soient comparables voir identiques à ce qui ce fait déjà sous Mac ou Windows. Attention, ces temps comprennent non seulement le développement de l'application mais aussi l'intégration des différents matériels, les tests, la maintenance du code etc.

Des systèmes hétérogènes

- C'est la dimension matérielle du problème
- Des entrées/sorties variées et multiples
- Problèmes de synchronisation
- Boîte noire ou systèmes intégrés ?

ni.com/france

NATIONAL INSTRUMENTS

Nous nous sommes intéressé aux applications et aux utilisateurs, il est temps d'étudier les matériels mis en œuvre et force est de constater que depuis quelques années, ces derniers sont devenus véritablement hétérogènes.

Cela tient d'une part au fait que, dans le cas du test par exemple, on en demande toujours plus et que pour cette raison, les systèmes actuels sont amenés, non seulement à traiter des signaux électriques classiques mais aussi, pourquoi pas, à conduire des inspections visuelles et à contrôler des axes moteur pour mettre tel ou tel produit sous une caméra.

Comme il s'agit d'applications de test et de mesure, la notion de synchronisation est importante (générer tel signal, générer tel autre, corréler les résultats…) et c'est naturellement que le PC a été promu chef d'orchestre. En effet, c'est un composant standard (architecture, système d'exploitation…) et c'est le seul qui permette aux utilisateurs de définir leurs propres systèmes, de synchroniser le tout et de bénéficier d'une vitesse de calcul et d'options de communication dignes des bancs de test du XXI^{ème} siècle.

Pour prendre un contre exemple, on peut imaginer un PC relié à un ou plusieurs boîtiers (acquisition, vision, commande d'axe, instruments de mesure…). Outre les problèmes de connectique avec les boîtiers (liaison série, USB, FireWire…) et même si des signaux de déclenchement peuvent être échangés d'un boîtier à un autre, les questions suivantes vont se poser : chaque boîtier va venir avec sa propre API, pourrais-je facilement en tirer partie dans mon environnement de développement ? J'ai des boîtiers, des câbles entre le PC et les boîtiers et d'autres encore entre les boîtiers (trigger). Y a-t-il moyen d'avoir un système plus compact et surtout plus homogène ? Chaque boîtier vient avec ses propres outils de diagnostic et de test. Pourrais-je, sans avoir à développer quoi que ce soit, tester complètement mon système ? Le système est fonctionnel, mais il faut que je rajoute une ou deux voies de mesure. Comment pourrai-je faire évoluer mon système ? Etc.

Conséquences

- Il faut que le matériel existant soit compatible avec la solution RT proposée
- Il faut proposer un système modulaire

En conséquence, et compte tenu de l'expérience acquise, la solution temps réel de NI devra nécessairement être de type modulaire afin de permettre aux utilisateurs de ne payer que les fonctionnalités qu'ils utilisent et de pouvoir faire évoluer leurs systèmes. Autrement dit, il faut que la démarche NI dans le domaine du temps réel soit similaire avec celle conduite sur les stations Mac ou Windows.

De plus, il faut que la solution temps réel puisse à terme tirer parti de tous les types de moyens d'acquisition disponibles au catalogue NI puisque les systèmes des utilisateurs sont hétérogènes.

La résolution de l'équation

- Pas de standard donc on supprime l'OS !
- Oui, bien sûr, il y a un OS RT mais non, on en parle pas
- On s'appuie sur les outils de configuration classiques et le modèle de programmation LabVIEW
- On développe un minimum de nouveaux matériels

ni.com/france

NATIONAL INSTRUMENTS

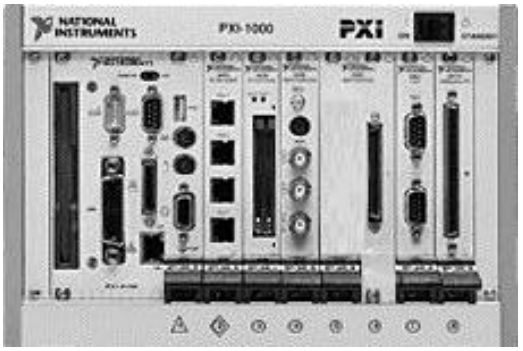
Il est temps de résoudre le problème posé.

On l'a vu, il n'y a pas de véritable standard en terme d'OS temps réel. La solution consiste donc pour National Instruments à occulter le problème en parlant de solution pour le développement d'applications temps réel plutôt que de vendre telle ou telle spécificité de tel ou tel OS. Ce faisant, cela permet aux utilisateurs et à NI de rester focalisés sur les problèmes de mesures tout en laissant la possibilité à NI de changer d'OS.

En ce qui concerne le développement logiciel, compte tenu de la base installée et du fait que LabVIEW, dans le domaine du test et de la mesure, est un standard à lui tout seul, c'est sur ce dernier que l'on s'appuie. Techniquement, la solution consiste à venir compléter un LabVIEW classique avec un « plug in » qui lui permet de générer du code qui sera exécuté en temps réel. Ainsi, deux cibles sont atteintes. On ne change pas les méthodes de développement des utilisateurs (ce qui rassure et facilite le portage des applications) mais surtout on garantit des temps de mise en œuvre similaires à ce qui se fait déjà.

Pour finir, du point de vue matériel, les seuls développements pour NI consistent en la création de modules susceptibles d'exécuter le code LabVIEW en temps réel et de faire l'interface entre ce dernier et tous les modules d'acquisition et de contrôle du catalogue National Instruments (le 7 février 2002 tous les modules ne sont pas encore pilotables en temps réel mais c'est vraiment le but).

En pratique…



ni.com/france

Voyons sur un exemple pratique comment cela peut se passer. Imaginons une application mettant en œuvre des moyens d'acquisition et de contrôle hétérogènes, écrite avec LabVIEW et tournant sous Windows 2000 sur un châssis PXI.

Supposons maintenant que pour des raisons liées à la sécurité (quoiqu'il arrive à Windows, il faut que l'application s'exécute) ou aux contrainte de temps (déterminisme), l'utilisateur ait besoin de faire passer son application en version temps réel. Voici pas à pas la méthode à suivre :

1/ J'appelle National Instruments France, je fais le point avec mon interlocuteur pour savoir si tous les modules qui sont dans le châssis sont contrôlable en temps réel.

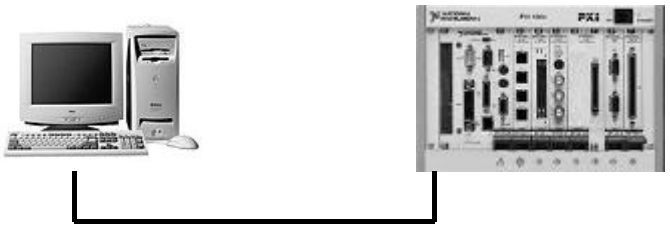
2/ Comme il n'y a pas de problème, je passe commande sur le web d'une extension temps réel pour LabVIEW. Deux jours après je reçois mon package LabVIEW Real Time.

3/ Sur le châssis PXI, je désinstalle ma licence LabVIEW 6 et je la réinstalle sur un PC de bureau classique équipé de Windows 2000. Je n'ai donc pas de licence supplémentaire à acheter.

4/ J'installe l'extension LabVIEW Real Time sur le PC de bureau (là ou je viens d'installer LabVIEW)

5/ Avec l'extension temps réel, et sans sortir de LabVIEW, je crée une disquette de démarrage. Une fois qu'elle est créée, je l'insère dans le lecteur du contrôleur PXI et je redémarre ce dernier. Je n'ai donc pas de nouveau matériel à acheter.

En pratique…



ni.com/france

NATIONAL INSTRUMENTS

6/ Je tire un câble réseau entre le PC de bureau et le contrôleur PXI.

7/ Avec MAX (Measurement and Automation Explorer), depuis le PC de bureau, je retrouve automatiquement le contrôleur PXI sur le réseau et je procède via le réseau à l'installation du firmware (le moteur RT, DAQ RT etc.). Cette opération n'est fait qu'une seule fois.

9/ J'en profite pour configurer, depuis le PC de bureau, les différents modules d'acquisition qui sont dans le châssis PXI.

10/ Je récupère sur le serveur mon entreprise le code source de mon application LabVIEW et je recompile le tout en version temps réel sur le PC de bureau.

11/ Sur le PC de bureau, depuis l'environnement LabVIEW, je demande à télécharger le code nouvellement compilé dans le contrôleur PXI. L'application s'exécute alors en temps réel sur le châssis.

12/ Je déconnecte le câble Ethernet et j'éteins le PC de bureau. Ma journée est terminée, je laisse l'application temps réel s'exécuter sur le contrôleur PXI et je rentre chez moi.

Facultatif :

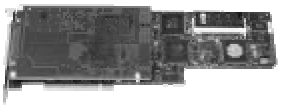
13/ Le lendemain matin, je rallume le PC de bureau, branche le câble réseau entre les 2 stations et relance LabVIEW. Quand je lance l'exécution du VI sur le PC de bureau, LabVIEW Real Time établit automatiquement la liaison entre le PC et le code qui s'exécute en temps réel. Je peux alors, depuis le PC de bureau, utiliser les contrôles de la face avant. Les changements (consignes) sont automatiquement transmises à l'application temps réel qui tourne sur le contrôleur PXI.

Bien sûr, cette description est un peu réductrice et il existe tout un ensemble d'options et de configurations possibles. Ces dernières font l'objet de présentations spécifiques lors de NIDays 2002.

Quoiqu'il en soit l'état d'esprit de la démarche est bien celui expliqué et l'on peut se rendre compte de la façon dont toutes les contraintes exprimées précédemment ont

Prise en compte des demandes ?

1999 : **Carte DAQ RT**



2000 : **Système PXI RT**

2001 : **Module FP-20xx**

ni.com/france

NATIONAL INSTRUMENTS

On retrouve ici la liste des principaux produits temps réel dont les sorties respectives ont jalonné l'historique de la ligne de produit Real Time.

Il est à noter que l'on retrouve essentiellement des contrôleurs. Qu'il s'agisse du PXI ou de FieldPoint, ces derniers sont là pour permettre l'exécution du code temps réel et faire l'interface avec les modules standards d'acquisition et de contrôle. Le terme standard est ici très important puisque lors de l'établissement d'une nouvelle configuration temps réel ou l'évolution d'un système existant, les modules de mesure proprement dit resteront les mêmes. Autrement dit, du point de vue matériel et financier, rien ne permet de distinguer une configuration temps réel d'une autre.

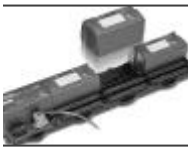
On voit donc ici que l'une des plus importantes requêtes de la part des utilisateurs a été prise en compte et l'on peut se demander si les autres retours d'informations qui ont été formulés l'ont été eux aussi.

Moins cher !

- Les nouveaux contrôleurs PXI
- Les systèmes FieldPoint



- Systèmes modulaires
- Le coût global de mise en oeuvre

ni.com/france

NATIONAL INSTRUMENTS

C'est une demande constante des clients vis-à-vis de leur fournisseurs et NI n'échappe pas à la règle. Ceci dit, il faut peut être là aussi expliquer les différentes réponses qu'apporte National Instruments au problème du coût.

On peut attaquer le problème de front et c'est ce qui est fait avec les nouveaux contrôleurs PXI dont le prix a quasiment été divisé par deux en moins d'un an. De même, et de manière plus spécifique, des nouveaux contrôleurs spécialement conçus pour les applications temps réel voient leur prix passer sous la barre des 1500 €.

On peut aussi envisager le problème du prix sous l'angle du prix de revient par voie de mesure. Afin d'augmenter le nombre de voies et diminuer d'autant le coût par voie NI a dans un premier temps rendu possible le contrôles de plusieurs modules à partir d'un contrôleur PXI. En 2001, il fut possible de faire tourner des applications temps réel sur des contrôleurs FieldPoint ce qui a permis non seulement de réaliser des systèmes temps réel de test et mesure distribués mais aussi, compte tenu du prix des contrôleurs et du nombre de voies utilisables, de diminuer le coût de chaque voies d'E/S.

Par la suite, on peut regarder le prix en terme d'efficacité. Par exemple, est il efficace de payer 1000 € un système temps réel d'acquisition de données dont on utilise que 30% des voies ? Oui, c'est sûr, pour le coup, on a de quoi faire évoluer le système. Ceci dit, comme le module en question n'était disponible qu'avec ces options, on a pas eu le choix. Enfin, demain, si il faut ajouter des voies dont les spécificités ne sont pas supportées par le module en question, faudra t il acheter un autre système « économique ». On l'a dit, afin d'optimiser l'efficacité de ses investissements, il faut passer par un système modulaire. Historiquement, le 5B est un bon exemple. De manière plus récente et plus moderne, le SCXI, le VXI, les PXI et FieldPoint sont d'autres très bons exemples.

Le point ici est que l'utilisateur ne paie que les fonctionnalités dont il a besoin et dispose d'options pour ajouter, en toute liberté, d'autres voies d'acquisition selon les besoins du moment.

Le dernier point, et c'est très certainement le plus important, consiste à étudier le coût global de mise en œuvre de la solution. En effet, il n'est pas raisonnable de ne comparer que le coût du matériel. En effet, ce qui compte, ce n'est pas le coût d'achat mais bien le coût de mise en œuvre. Ainsi, il n'est sans doute pas inutile de comparer ce qui est vraiment comparable et d'inclure les coûts du matériel, des licences logicielles, du temps de test et de validation de la configuration et enfin le coût du temps de développement de la toute première application. A l'extrême, on pourrait envisager que vous receviez gratuitement un PC, une licence logicielle (OS et outils de développement temps réel) ainsi qu'une carte d'acquisition mais qu'il n'existe pas d'outil de configuration ni de driver pour utiliser la carte en question. On voit bien qu'il faudra alors soit payer soit réaliser en interne un driver de bas niveau et ensuite réaliser l'application proprement dite. Tout ceci, qu'on le veuille ou non, a un prix et si on se place encore à l'extrême en considérant que le coût ingénieur est négligeable, de toute façon il faudra plus de temps. Quand on met tout ceci avec les contrainte industrielles de nos utilisateurs et, entre autres les temps qui leur sont alloués pour développer leurs applications (le PC est un outil, ce qui compte, c'est la mesure, pas les moyens de mesure) il y a fort à parier que le projet sera en retard et ça, c'est tangible, concret et cela se paie par des pénalités.

Plus de possibilités matérielles

- DAQ et conditionnement du signal
- Série
- GPIB
- CAN
- Commande d'axes
- …

ni.com/france

NATIONAL INSTRUMENTS

On l'a dit, la stratégie de National Instruments est de faire en sorte qu'à terme l'ensemble des modules d'acquisition et de contrôle soient intégrables au sein d'un système temps réel. Si aujourd'hui tout n'est pas encore disponible la liste ci-dessus permet de couvrir les besoins les plus importants. Bien sûr, même si il n'est pas possible de les mentionner ici, la liste et les priorité associées aux modules qu'il faut rendre disponible sous RT est déjà établie.

D'autre part, et c'est sans doute le plus important, l'historique de NI devrait rassurer ceux qui sont peut être encore dubitatifs. En effet, National Instruments a identifié le temps réel comme étant l'une des priorités majeures pour les prochaines années et tout sera fait pour simplifier et démocratiser la création d'applications de test, mesure et contrôle temps réel.

Plus d’options logicielles

- Tous les toolkit type PID, Logique floue etc.
- Analyse Point par Point de la version 6.1
- LabWindows/CVI (ANSI C)
- Visa

ni.com/france

NATIONAL INSTRUMENTS

Il est à noter qu’à partir du moment où un toolkit LabVIEW est développé avec LabVIEW (et sous réserve qu’il ne comporte pas de parties spécifiques à tel ou tel OS) ce dernier est utilisable pour le développement d’applications temps réel.

Ce point remarquable qui est une conséquence directe du fait que dans sa stratégie, NI s’est attaché plutôt à l’environnement de développement qu’à l’OS temps réel. En d’autres termes, si NI avait décidé de réaliser des développements spécifiques pour tel ou tel OS, oui sans doute certaines performances auraient pu être supérieures d’entrée de jeu. Ceci dit, en s’attachant à faire de LabVIEW Real Time, la couche d’interface entre le développement et un OS temps réel, l’utilisateur est gagnant sur toute la ligne : outre un portage aisé des application existantes, l’ensemble des toolkits sont utilisables.

A l’occasion d’une des tables ronde organisée pendant de NIDays 2001, il avait été clairement demandé de trouver un moyen pour que certains codes écrits en C puissent être utilisés au sein d’applications temps réel. Avec la sortie de Measurement Studio 6 (Aout 2001) le problème fut résolu et il est possible dorénavant de créer, à partir de LabWindows/CVI, un code qui peut être facilement utilisable dans un diagramme LabVIEW. Si en première approximation, cela ne répond pas à toutes les questions, c’est un moyen très efficace de réutiliser tout ou partie de code que l’on ne souhaite pas ré-encoder sous forme de diagramme LabVIEW.

Ce qu’il faut remarquer, ce qu’à l’instar de ce qui se fait au niveau du matériel, la machine NI est lancé et que le but est de rendre, grâce au logiciel, la programmation temps réel aussi facile et aussi riche que sur d’autres OS.

Conclusion

- La solution proposée est efficace car elle focalise sur un domaine
- Support natif des systèmes d'E/S hétérogènes

ni.com/france

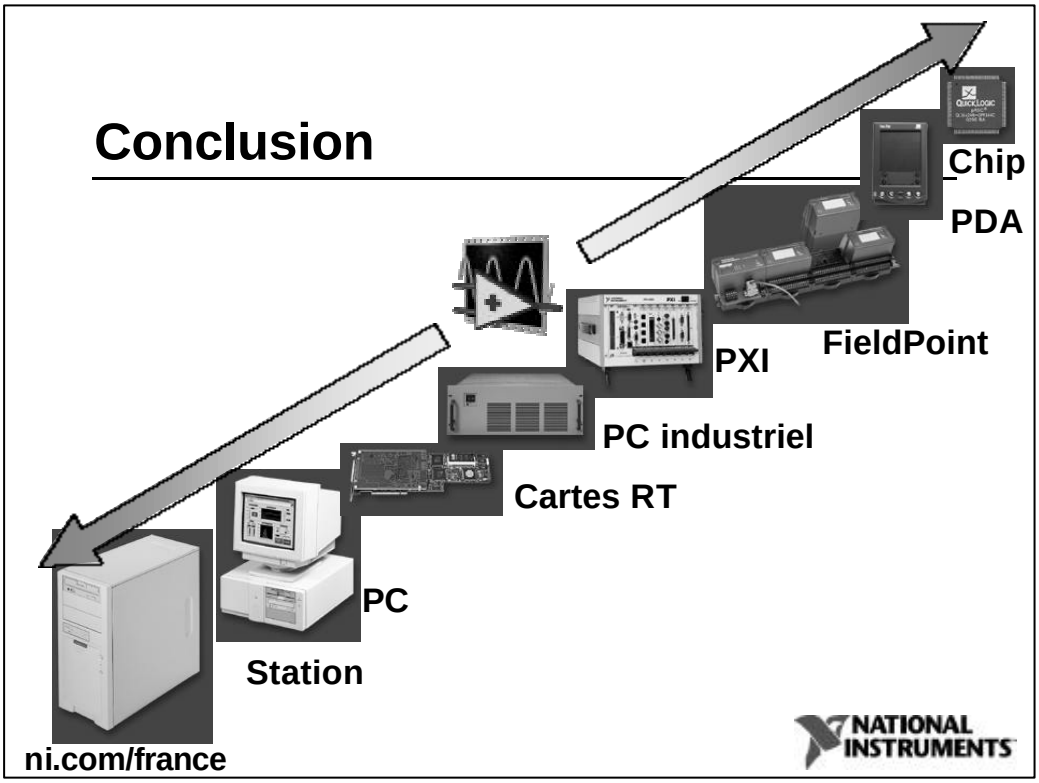
NATIONAL INSTRUMENTS

Il est temps de conclure et s'il n'y avait qu'une seule chose à retenir, cela devrait être que la solution NI pour la réalisation d'applications temps réel de test, de mesure et de contrôle est efficace car elle focalise sur un domaine précis.

En effet, à l'instar de ce qui se fait sous Windows par exemple où LabVIEW ne s'intéresse pas au développement d'applications de gestion, ou de traitement de texte, LabVIEW RT ne focalise que sur les applications de contrôle commande, de simulation et de stimulation. Le point important, c'est qu'en ne focalisant que sur un domaine précis, l'équipe NI est capable de prendre en compte les spécificités et les difficultés que rencontrent les utilisateurs, de les intégrer et donc de rendre plus facile tel ou tel type de développement. Par exemple, par rapport à des outils de développement temps réel génériques, la solution NI vient avec tous les outils pour faire du contrôle PID, des entrées/sortie, de la synchronisation entre modules etc. C'est toute la différence qui existe entre des outils de développement à usage général et d'autres qui sont et restent spécialisés sur un métier particulier.

Le second point important qui différencie l'offre National Instruments est le fait qu'elle est spécialisée dans le pilotage et le contrôle de matériels de mesures hétérogènes. Il n'y a rien de révolutionnaire a priori ici. Ceci dit, quand on compare cette offre avec ce qui existe sur le marché, il n'y a pas de doute, l'offre NI est certainement l'une des plus riche. Quand on remarque que dans le cadre du marché de l'embarqué (différent mais proche de celui du temps réel), plus de 20% des développeurs sont à la recherche de moyen d'acquisition, il n'est pas étonnant, d'une part de voir les utilisateurs classiques de LabVIEW, faire migrer leurs applications mais surtout de voir les développeurs temps réel s'intéresser à l'offre NI.

Conclusion


Chip
PDA
FieldPoint
PXI
PC industriel
Cartes RT
PC
Station
ni.com/france
NATIONAL INSTRUMENTS