CHOIX D'UN SYSTÈME D'EXPLOITATION POUR APPLICATIONS EMBARQUÉES

COLIN WALLS
TECHNICIEN LOGICIELS EMBARQUÉS

LIVRE BLANC

Mentor®
A Siemens Business

LOGICIELS EMBARQUÉS

www.mentor.com

INTRODUCTION

De nos jours, il est rare de trouver un système embarqué dépourvu de système d'exploitation (SE). Seul le type d'appareil le plus simple peut être construit efficacement sans un type de noyau quelconque. Toutefois, cette possibilité ne doit pas être écartée. La gamme complète d'appareils embarqués peut être représentée par un graphique de la complexité du processeur (généralement la largeur du bus de données) par rapport à la complexité du logiciel (figure 1).

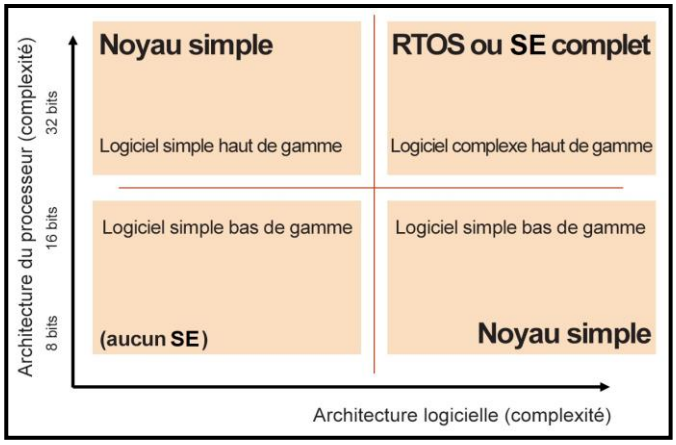

Figure 1 : *Les quatre quadrants : basés sur la complexité du processeur par rapport à la complexité de l'architecture logicielle.*

Cette dernière peut être grossièrement divisée en quatre quadrants. La partie supérieure droite (logiciels complexes exécutés sur un processeur haut de gamme) représente le champ traditionnel des systèmes d'exploitation temps réel (RTOS) et d'autres systèmes d'exploitation. Sur un processeur moins puissant, il peut être toujours utile de déployer un noyau de base si le logiciel est relativement complexe. Parfois, un processeur puissant est employé pour exécuter un logiciel relativement simple, lorsque la performance du processeur est nécessaire pour accélérer la vitesse d'exécution. Dans ce cas, l'utilisation d'un noyau n'est pas strictement obligatoire, mais peut être une approche prudente car elle accroît l'évolutivité de l'architecture logicielle et permet ultérieurement de prendre en charge une augmentation de la complexité. Ce n'est véritablement que lorsqu'un logiciel simple est exécuté sur un appareil bas de gamme qu'il est possible de se passer d'un noyau quelconque.

Après avoir conclu qu'un SE est nécessaire pour un projet, reste la question du choix. Généralement, quatre options sont possibles : opter pour un système d'exploitation haut de gamme, tels que Linux® ou une variante embarquée de Windows, ou choisir un système d'exploitation temps réel, pour lequel un large choix est disponible, déployer l'un des systèmes d'exploitation libres aisément accessibles ou mettre en œuvre un noyau en interne. On trouvera ci-dessous une évaluation de ces différentes options.

SYSTÈMES D'EXPLOITATION COMMERCIAUX

Le marché foisonne de bons systèmes d'exploitation. Ces produits présentent un certain nombre d'avantages et d'inconvénients par rapport aux autres alternatives.

AVANTAGES

De nombreux produits RTOS commerciaux sont disponibles, la plupart provenant de fournisseurs à la réputation bien établie. Toutefois cet aspect doit être soigneusement évalué. La taille de l'entreprise, la maturité du produit et la base d'utilisateurs sont des facteurs clés. L'une des principales exigences est la disponibilité du support technique.

Dans le processus de sélection d'un RTOS, l'acheteur et le vendeur prennent un engagement à long terme. La capacité migration future vers un nouveau processeur constitue l'un des aspects essentiel de la relation. Vous pouvez faire confiance à un fournisseur RTOS de renom pour prendre en charge de nouveaux appareils en respectant les échéances de votre projet. Leur produit est probablement conçu pour simplifier le processus de portage.

Une bonne documentation est essentielle et peut être exigée d'un fournisseur de RTOS commercial. Il n'y a aucune raison pour ne pas demander à voir un échantillon des manuels. Autrefois, l'expédition dans le monde entier de manuels imprimés aurait été certainement difficile. Aujourd'hui, l'envoi d'un fichier PDF est une attente relativement facile à satisfaire.

Le code source est disponible pour de nombreux SE commerciaux, parfois gratuitement ou moyennant des frais spécifiques. Il est préférable de s'assurer que le code est lisible car le code source peut rapidement devenir incompréhensible pour un humain. Aussi, ce code doit être suffisamment commenté. Il n'est pas censé remplacer une solide documentation, mais s'avère un complément utile.

Le développement d'une application parallélisée (multithread) comporte de nombreux défis, l'un d'eux étant le débogage. Disposer d'un type quelconque de débogueur compatible avec le RTOS peut être très utile. La prise en charge de point d'arrêt global (*où le processeur est momentanément interrompu)* est recommandée. Le mode exécution (*où seule la tâche en cours est arrêtée)* peut être également utile dans d'autres cas.

Il est rare qu'une application embarquée se contente d'un simple noyau parallélisé. D'autres composants logiciels sont généralement nécessaires. Un SE commercial est susceptible d'en compter un large éventail, notamment des protocoles de communication qui doivent être entièrement validés, des systèmes de fichiers (y compris Flash) et un module graphique pour faciliter la conception de l'interface utilisateur.

L'un des défis propres au développement d'applications embarquées est le travail relatif à l'exploitation des périphériques du matériel. Avec un SE, ce dernier implique le développement de pilotes spécifiques. Un SE commercial disposera probablement d'une vaste gamme de pilotes pour les périphériques les plus standards et permettra le développement de pilotes personnalisés.

INCONVÉNIENTS

En termes techniques, chaque système embarqué est différent. Le processeur, la mémoire et les périphériques varient tous d'un appareil à l'autre. Les systèmes diffèrent également en fonction du marché visé, du prix final de l'équipement, ainsi que les volumes produits. Ces différences ont une incidence sur les options de licence du SE. Dans certains cas, le paiement de droits de reproduction (quelques euros par un appareil) est raisonnable. Dans d'autres cas, notamment lorsque des volumes très importants sont prévus, un modèle sans redevance de recopie peut être idéal.

L'une des objections couramment avancées contre l'utilisation d'un SE commercial est le manque de connaissances internes du développeur quant à ses fonctions. Ce qui peut être

vrai, mais est-ce réellement important ? Si le SE se comporte conformément à la documentation, pourquoi importe-t-il de savoir exactement comment il parvient au résultat ? Il est légitime de penser que, dans la mesure où l'exploitation d'un SE exige une certaine spécialisation, la plupart des équipes de développement de logiciels embarqués n'ont pas les moyens de maintenir une telle expertise. Si le code source est disponible, il apporte une garantie au cas où le fonctionnement interne du SE devrait être vérifié.

Les acheteurs de systèmes d'exploitation, comme toute personne achetant un produit, ne veulent pas dépendre d'un seul fournisseur ; même si ce dernier accueillerait favorablement cette éventualité. Avec les produits de SE commerciaux, l'une des principales différences entre les diverses options est l'interface de programmation, l'API. En réalité, ces dernières varient peu. Un portage ultérieur, si un changement de fournisseur de SE s'avérait nécessaire, ne représentera donc probablement pas un problème majeur. Le respect des normes, bien sûr, engendre pour le moins un certain degré d'autonomie vis-à-vis du fournisseur. Dans ce cas, l'API POSIX constitue probablement la meilleure option.

Une autre objection courante aux SE commerciaux est leur nombre excessif de fonctionnalités, car ils doivent répondre aux besoins d'une importante base de clients. Bien que cela soit vrai, le coût des fonctionnalités est largement amorti et n'a donc pas d'impact réel sur un client donné. Il est également probable qu'un SE soit proposé sous la forme d'un ensemble de composants, dont bon nombre sont optionnels. Aux débuts des RTOS commerciaux, les produits n'étaient qu'un assemblage monolithique de blocs de code. Face à l'augmentation du nombre de fonctionnalités disponibles, la notion de configurabilité est apparue. Un SE entièrement configurable permet d'intégrer uniquement les fonctionnalités requises dans l'image exécutable finale.

SYSTÈMES D'EXPLOITATION « LIBRES »

Dans ce contexte, les SE gratuits ou « libres » n'incluent pas vraiment Linux, car la plupart des développeurs de systèmes embarqués sont susceptibles d'investir dans l'achat d'une distribution prise en charge et intégrée de Linux. Ces distributions ne sont donc pas réellement gratuites. Cette section tient compte des RTOS plus compacts et plus facilement téléchargeables, particulièrement prisés.

AVANTAGES

L'intérêt majeur ici est clairement l'absence de coûts initiaux, également après le déploiement, car il n'y a aucun frais de licence à prévoir.

Les SE libres ont toujours tendance à inclure le code source. Ce dernier est certainement utile à titre de référence, car la documentation peut être limitée et l'assistance difficile à obtenir. Il constitue également une exigence pour la configuration et le portage vers un nouvel environnement matériel, ce qui, bien sûr, incombe à l'utilisateur.

Comme pour de nombreux types de logiciels pris en charge par l'utilisateur, les RTOS libres attirent souvent une forte clientèle, ce qui se traduit par une communauté en ligne dynamique à partir de laquelle l'assistance est disponible gratuitement. Bien que cet aspect soit particulièrement attrayant et utile, l'une des préoccupations est leur longévité. Les logiciels embarqués ont tendance à suivre les technologies ou produits à la mode. Que se passe-t-il si le RTOS que vous avez choisi devient obsolète ? En outre, le type de soutien de la communauté est généralement axé sur la version actuelle du logiciel.
Si votre produit utilise une version antérieure, vous aurez peut-être des difficultés à trouver de l'aide pour résoudre des problèmes.

INCONVÉNIENTS

Le déploiement d'un SE sur un équipement embarqué est un engagement à long terme. La question d'un support à long terme est donc primordiale. Pour un SE libre, pouvez-vous compter sur la présence de la communauté à long terme ? Aussi, la documentation est-elle à la hauteur étant donné qu'elle peut souvent apporter des solutions ?

Le développement d'un code parallélisé est une tâche complexe et l'utilisation d'outils de débogage adéquats s'impose. Si un SE libre est prisé, un tel outil peut avoir été développé par un tiers, mais la question de l'assistance demeure.

Les logiciels libres représentent un défi psychologique. Dans la mesure où le code source est immédiatement disponible et que les développeurs le perçoivent comme n'ayant aucune

valeur monétaire, la tentation de l'« améliorer » est grande. Cette amélioration peut prendre la

forme d'ajustements mineurs visant à accroître l'efficacité du code ou d'ajout de nouvelles fonctionnalités. Dans les deux cas, il peut en résulter l'utilisation de nombreuses versions similaires mais différentes du SE dans plusieurs projets. La maintenance devient alors un véritable casse-tête.

Dans la plupart des applications embarquées dans lesquelles un SE est déployé, l'utilisation d'un composant plus élaboré qu'un simple noyau parallélisé, par exemple un système de fichiers ou une pile réseau, est nécessaire. De telles options peuvent être disponibles, mais dans le cas contraire, un effort supplémentaire doit être consenti pour localiser le composant logiciel nécessaire et le porter ou l'adapter au SE choisi.

L'un des principaux critères de sélection d'un SE (qu'il soit libre ou non) est la prise en charge du processeur qui a été choisi pour votre projet. Cependant, après avoir choisi le SE, un investissement considérable en temps et efforts est indispensable. S'il peut être mis à profit sur un grand nombre de projets, cela en vaut la peine. Si différents processeurs sont sélectionnés, comment garantir la disponibilité actuelle ou future de l'assistance ?

Tous les logiciels, qu'ils soient libres ou commerciaux, font d'une manière ou d'une autre l'objet d'une licence. La licence peut être un document très complexe qui limite le déploiement du logiciel d'une certaine façon. Les sanctions en cas de violation d'une licence peuvent être sévères. Les logiciels Open source peuvent s'avérer particulièrement problématiques, car la licence risque de compromettre l'état de votre code applicatif, vous obligeant ainsi à mettre son code source à la disposition du public. Avant d'intégrer un code Open source dans votre application, il est recommandé d'obtenir l'avis d'un conseiller juridique.

SYSTÈMES D'EXPLOITATION PERSONNALISÉS

AVANTAGES

La raison la plus couramment invoquée pour justifier le développement d'un noyau en interne est le maintien du contrôle complet du code. Cette approche semble judicieuse, mais elle présuppose que les ressources possédant les compétences nécessaires, qui sont relativement spécialisées, soient maintenues en fonctions.

Il n'existe, bien entendu, aucun coût de licence permanent pour le code développé en interne. Mais les coûts récurrents de maintenance ne peuvent pas être ignorés.

Un autre motif couramment avancé pour le développement en interne est la capacité des RTOS obtenus à répondre précisément aux exigences du projet. On pourrait penser que les fonctions du noyau sont plus susceptibles d'être définies en fonction des efforts de développement pouvant y être consacrés. Un SE commercial, entièrement configurable, sera très probablement en mesure de répondre précisément aux besoins et ce sans compromis.

INCONVÉNIENTS

Toute opération de développement de logiciel coûte de l'argent. Cependant, les coûts de développement d'un noyau en interne sont couramment absorbés dans un projet de sorte qu'ils ne sont pas visibles.

Une fois déployé, il est probable qu'un SE soit utilisé pendant un certain temps. Une assistance à long terme est par conséquent nécessaire. Cela ne pose pas de problème si les développeurs restent en fonction. En outre, le besoin d'assistance est moins problématique si le code est minutieusement documenté. Mais il s'agit là d'hypothèses dangereuses.

Comme indiqué précédemment, le code parallélisé peut présenter des erreurs subtiles, qui nécessitent un débogueur compatible avec le SE pour être détectées. Après avoir créé un noyau, il est peu probable que des efforts supplémentaires soient déployés pour développer un tel outil. Une solution consiste à adapter un débogueur commercial pour le noyau interne.

Il est probable qu'un noyau interne soit utilisé dans plusieurs projets au fil du temps. Toutefois, il est à craindre que chaque équipe de projet cherchera à améliorer le code de base. Il en résulte plusieurs versions du noyau, ce qui augmentera le problème de support et de maintenance.

Il est peu probable qu'un simple noyau suffise. Le développement de composants logiciels supplémentaires augmenterait davantage les coûts de développement et il est peu probable que les fournisseurs de composants commerciaux envisagent la possibilité d'un portage vers des RTOS propriétaires.

A mesure que la technologie évolue, il sera nécessaire d'envisager de changer de processeur. Cela est un problème si le noyau exige des caractéristiques de traitement très sophistiquées. Généralement, une partie significative du langage assembleur ainsi que la prise en charge de « l'endianisme » (organisation de l'encodage) et des interruptions doivent être examinés.

Le principal inconvénient du développement d'un SE en interne est d'ordre philosophique. Les entreprises les plus performantes se concentrent toujours sur leurs compétences de base. À moins d'être un développeur de noyaux, il semble plus logique de se focaliser sur votre cœur de métier.

CRITÈRES DE CHOIX DU SE

Une fois la décision prise d'utiliser un SE commercial ou du moins supporté commercialement, une série de questions de qualification doivent être résolues :

- *Votre application s'exécute-t-elle en temps réel ?*
  Temps réel ne veut pas forcément dire rapide, mais plutôt prévisible ou déterministe. Ces termes ne sont pas absolus. Il s'agit de savoir dans quelle mesure votre système doit réagir de manière prévisible aux événements. Quelle est l'importance du facteur temps ? Si vous nécessitez un niveau élevé de déterminisme, alors un RTOS est probablement votre meilleur choix. Dans certains cas, Linux peut convenir, en particulier si l'utilisation d'extensions en temps réel est acceptable.

La taille de la mémoire est-elle limitée ?

Contrairement à un ordinateur de bureau, la taille de la mémoire de la plupart des systèmes embarqués est limitée. La quantité de mémoire disponible est un autre critère de choix primordial. À moins que plusieurs mégaoctets soient disponibles pour le SE, il est peu probable que la mise en œuvre de Linux, soit une solution viable.

- *La puissance du processeur est-elle limitée ?*

  La puissance de processeur disponible est tout aussi importante. Si la puissance du processeur est tout juste suffisante pour exécuter l'application, vous ne disposez d'aucune marge pour absorber les coûts d'exécution introduits par le SE. Les systèmes d'exploitation en temps réel ont tendance à utiliser le temps processeur de manière très efficace mais également prévisible.

- *La consommation d'énergie de l'appareil est-elle une priorité ?*

  Sur de nombreux types de systèmes, la consommation d'énergie suscite un nombre toujours plus croissant de préoccupations, par souci de préservation de la durée de vie de la batterie sur les appareils portatifs ou pour des raisons environnementales et économiques dans le cas de systèmes fixes. Les critères précédents de taille de mémoire et de puissance processeur ont un impact à ce niveau. La taille et l'efficacité d'exécution du SE ont donc leur importance. Un certain nombre de systèmes d'exploitation incluent des fonctions de gestion de consommation d'énergie. Linux inclut quelques-unes de ces fonctionnalités et un nombre croissant de RTOS intègrent des mécanismes de gestion d'alimentation, notamment notre propre RTOS Nucleus.

- *Disposez-vous de périphériques inconnus ou personnalisés ?*

  Un système embarqué inclut toujours un certain nombre de périphériques parallèlement au processeur. Si ces périphériques sont largement utilisés, des pilotes devraient être disponibles, quel que soit le SE choisi. Il en va de même pour les protocoles de communication. Si vous possédez des appareils moins connus, vous risquez d'avoir des problèmes. Même si de nombreux RTOS supportent une gamme étendue de pilotes, ils seront presque toujours devancés par Linux. Si vous disposez d'un matériel unique et personnalisé, un pilote devra être écrit. L'un des avantages de Linux est la disponibilité d'une très large base de compétences pour le développement de pilotes. Vous avez donc la possibilité de recruter du personnel ou de demander l'aide d'un fournisseur prenant en charge un système Linux embarqué, tel que Mentor Embedded.

- *Disposez-vous d'une unité de gestion de la mémoire (ou pouvez-vous en inclure une) ?*

  Si votre conception n'inclut pas d'unité de gestion de la mémoire (MMU), il ne sera pas possible d'utiliser Linux, car une MMU est obligatoire pour tous les systèmes d'exploitation supportant un modèle de processus. La plupart des RTOS supportent des modèles d'exécution de type fil (« Thread ») et ne nécessitent pas de MMU. Cependant, certains RTOS peuvent utiliser efficacement une MMU, si celle-ci est disponible.

- *La sécurité des applications est-elle une priorité ?*

Certains appareils embarqués ont besoin d'une sécurité supplémentaire. Souvent, cela signifie que les tâches doivent être protégées les unes des autres. Une telle protection ne peut véritablement être mise en œuvre que si une MMU est disponible. Les systèmes d'exploitation haut de gamme et quelques RTOS utilisent un modèle de processus offrant un haut niveau de sécurité. D'autres RTOS, qui utilisent un modèle d'exécution de type « Thread », peuvent également offrir un niveau de protection adéquat en utilisant la MMU pour rendre simplement des parties de la mémoire inaccessibles au lieu de remapper entièrement la mémoire.

Ce mode, communément appelé mode protégé par thread, implique des d'exécution inférieurs à ceux du modèle de processus.

- *Votre application nécessite-t-elle une certification de sécurité ?*

  Dans certains secteurs, la certification est une obligation. Ce processus peut être coûteux et exige l'accès à l'intégralité du code source. Les coûts de certification étant d'une certaine façon liés au nombre de lignes de code, un SE de plus petite taille est naturellement attractif. Il est généralement nécessaire de certifier une application complète. Il n'est donc pas possible d'acheter un SE pré-certifié. Certains fournisseurs de SE peuvent vous aider en fournissant une partie de la documentation requise. Il est évidemment recommandé de choisir un SE qui a fait ses preuves dans le domaine d'application en question.

  Par exemple, même s'il n'est pas possible de vous vendre une version de Nucleus® RTOS « certifiée pour système médical », nous comptons, parmi nos clients, un grand nombre de personnes ayant utilisé le produit avec succès dans ces applications.

- *Nécessitez-vous d'une interopérabilité avec les progiciels ?*

  Si votre appareil doit assurer une interopérabilité importante avec des progiciels, dans ce cas, les produits Microsoft peuvent être un excellent choix.

- *Connaît-on le prix de vente et le volume livré pour l'application ?*
  Il est facile de se perdre dans les détails techniques lors de l'analyse des différents systèmes d'exploitation disponibles. Souvent, comme dans de nombreuses décisions d'achat, le coût est un facteur décisif. Dans ce cas, il ne s'agit pas seulement d'obtenir le meilleur prix possible. Le modèle de gestion est également déterminant. Pour certains appareils embarqués, le paiement d'une redevance sur chacun d'eux est raisonnable. Pour d'autres avec un volume de production important, un modèle libre de redevances est préférable.
  L'Open source est bien sûr un choix intéressant, mais, dans ce cas, comme pour les produits strictement commerciaux, les coûts de maintenance récurrents doivent être pris en compte.
- *Vous ou votre équipe justifiez-vous a-t-elle une expérience significative avec d'autres SE ?*
  La formation étant onéreuse et fastidieuse, il est toujours préférable de tirer parti de l'expérience existante chaque fois qu'il est possible. Si l'équipe de développement est familiarisée avec une API en particulier, cela peut avoir une influence significative sur le processus de sélection. Bien entendu, si cette expérience porte sur une norme, comme POSIX, votre marge d'action est plus grande. L'expérience auprès d'un fournisseur a également son importance.
  En particulier, les échanges antérieurs avec le support technique sont précieux. De même, tenez compte de la qualité de la documentation et du code source. Un contre-exemple, pour au moins un fournisseur, le code source est volontairement rendu illisible.

QUESTIONS RELATIVES AUX SYSTÈMES MULTI-CŒURS

ARCHITECTURES MULTI-CŒURS

D'une manière générale, on distingue deux types de systèmes multi-cœurs. Si les processeurs/cœurs sont tous identiques, ils sont qualifiés d'*homogènes*. Si les processeurs/cœurs ne présentent pas tous la même architecture, ils sont considérés comme *hétérogènes*. Un système peut également être un hybride des deux types.

Par ailleurs, on dénombre généralement deux architectures logicielles. Le *multitraitement symétrique* (SMP) désigne l'exécution d'un seul SE sur plusieurs cœurs qui répartit le travail entre ces derniers. Le SMP ne peut être mis en œuvre que sur un système multi-cœur homogène. Le *multitraitement asymétrique* (AMP) attribue à chaque processeur une instance propre d'un SE. L'AMP peut être mis en œuvre sur n'importe quelle configuration multi-cœur. Un hybride, dans lequel certaines parties du système forment un sous-système SMP et d'autres représentent un AMP, est tout à fait possible.

CHOIX D'UN SE MULTICŒUR

Avec un système SMP, le SE répartit le travail entre les cœurs disponibles. Pour cela, une variante de SE spécifique est nécessaire. Tous les systèmes d'exploitation haut de gamme disposent de cette option, puisqu'il s'agit d'une pratique courante sur les systèmes de bureau. Un nombre croissant de systèmes d'exploitation temps réel, tels que Nucleus RTOS, incluent une version SMP. De toute évidence, l'efficacité avec laquelle l'architecture multi-cœur est utilisée peut être un facteur clé de choix du SE.

Le choix du SE pour chaque cœur d'un système AMP impose de suivre la même procédure que lors de la sélection d'un SE pour un système mono-cœur. Cependant, reste également la question de la communication inter-cœurs, où l'utilisation de MCAPI peut être une solution intéressante. Par ailleurs, un hyperviseur peut être utilisé pour assurer la supervision globale d'un système AMP.

Il importe de réfléchir aux outils. Pour tout développement de logiciel embarqué, disposer des outils adéquats est essentiel. Dans le cas d'un projet multi-SE et multi-cœur, le choix de l'outil le mieux adapté est absolument crucial. Pour garantir la visibilité complète sur le système et, évaluer ses performances et déboguer les interactions complexes entre le code sur différents cœurs, l'utilisation d'outils sophistiqués est nécessaire. La disponibilité de ces outils peut avoir une influence majeure sur le choix des systèmes d'exploitation et le fournisseur que vous avez sélectionné. Mentor® Embedded Sourcery™ Analyzer est un outil idéal pour prendre en charge les conceptions multi-SE et multi-cœurs.

CONCLUSIONS

Le choix d'un SE embarqué est un processus complexe qui nécessite la prise en compte de plusieurs facteurs, à la fois techniques et commerciaux. En outre, une mise à profit de l'expérience est essentielle. Tous les ordinateurs de bureau sont en substance identiques. C'est pourquoi le choix se limite à trois systèmes d'exploitation, tous à peu près équivalents. D'autre part, les systèmes embarqués sont tous différents, ce qui explique le fait que nous ayons un aussi grand choix de produits SE sur le marché et pourquoi ce processus de sélection est nécessaire.

Pour de plus amples informations sur Mentor Embedded Sourcery Analyzer ou pour demander votre propre évaluation, rendez-vous sur http://www.mentor.com/embedded-software/sourcery-tools/sourcery-analyzer/

Pour en savoir plus sur le système d'exploitation en temps réel Nucleus, rendez-vous sur http://www.mentor.com/embedded-software/nucleus

À propos de l'auteur

Colin Walls compte plus de vingt-cinq ans d'expérience dans l'industrie électronique, se consacrant principalement aux logiciels embarqués. Intervenant fréquent dans des conférences et auteur de deux ouvrages sur les logiciels embarqués, Colin est membre du service marketing de la division Systèmes embarqués de Mentor.

Pour les toutes dernières informations produit, appelez-nous ou rendez-vous sur: www.mentor.com

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Garden
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

Mentor® A Siemens Business

MGC 05-14 TECH12110-w