ENSEIRB

Les Systèmes Embarqués
Linux pour l ’embarqué

Patrice KADIONIK

email : kadionik@enseirb.fr
http : http://www.enseirb.fr/~kadionik

HISTORIQUE

- V1.0 09/02 : création du document
- V2.0 09/03 : MAJ des offres Linux embarqué pour plus de clarté. Ajout mesures performances TR.
- V2.1 09/03 : Ajout chapitre sur la mise au point des systèmes embarqués.

CHAPITRE 0 : INTRODUCTION

INTRODUCTION

- Cette formation a pour but de présenter tous les éléments techniques pour appréhender le mode des systèmes embarqués d 'aujourd 'hui :
  - Les systèmes embarqués aujourd 'hui: systèmes embarqués, Temps Réel, Linux embarqué, Codesign…
  - Le codesign aujourd 'hui : le mariage du matériel avec le logiciel.
  - La mise au point des systèmes embarqués : conception, les outils de debug, trucs et astuces.
  - La connectivité Internet : protocoles Internet pour une connectivité IP. Positionnement par rapport à l 'offre réseau de terrain.

INTRODUCTION

- Cette formation a pour but de présenter tous les éléments techniques pour appréhender le mode des systèmes embarqués d 'aujourd 'hui :
  - Linux embarqué : Les concepts. Le panorama aujourd 'hui. Présentation de la mise en œuvre de µClinux comme exemple.
  - Le Temps Réel et Linux. Les concepts. Le panorama aujourd 'hui. Présentation de la mise en œuvre de RTLinux comme exemple.
- Des exemples d 'applications ENSEIRB mettant en œuvre les concepts développés précédemment seront donnés.

CHAPITRE 1 :
LES SYSTEMES EMBARQUES AUJOURD 'HUI.
LE BESOIN D 'EMBARQUER INTERNET

PARTIE 1 :
CARACTERISTIQUES D ’UN SYSTEME EMBARQUE

IMPORTANCE DU MARCHE DE L ’EMBARQUE

- Les systèmes (numériques) embarqués ont vu leur importance progresser au rythme de l’importance prise par les microprocesseurs.
  - 1971 : premier microprocesseur 4 bits 4004 d’Intel à 92,5 kHz vendu 200 $. Le succès a été là tout de suite.
  - Juin 1978 : premier processeur x86 8086 à 4,77 MHz (technologie 3 µm, 29000 transistors), bus d ’adresse 20 bits à 9,1 Mo/s, bus de données 16 bits.
  - Juin 1979 : 8088 intégré dans le premier IBM-PC en 1981.
  - Motorola, Zilog, TI ont emboîté le pas…
- Le marché des microprocesseurs est un marché qui croît de façon exponentielle.

IMPORTANCE DU MARCHE DE L ’EMBARQUE

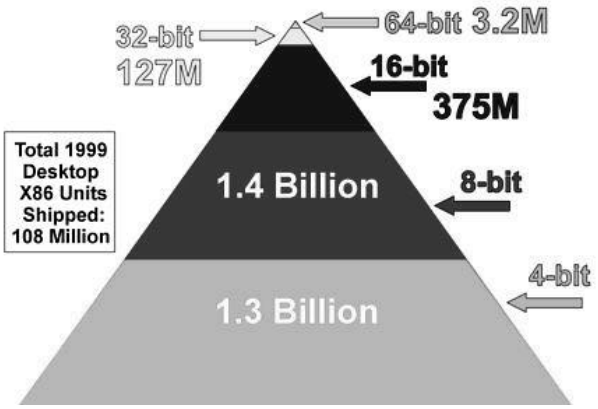
- Deux lois empiriques sont vérifiées depuis 30 ans (en plus de la loi de Moore) :
  - Loi de JOY : la puissance CPU en MIPS double tous les 2 ans.
  - Loi de RUGE : on a besoin d’une Bande Passante de 0,3 à 1 Mb/s par MIPS.
- Le marché du microprocesseur a aussi tiré le marché des systèmes embarqués (et des télécommunications !).

IMPORTANCE DU MARCHE DE L ’EMBARQUE

- Grâce aux progrès de l’intégration sur silicium, on est passé rapidement du processeur 4 bits au :
  - processeur 8 bits.
  - processeur 16 bits.
  - processeur 32 bits.
  - processeurs 64 bits.
- Il ne faut pas croire que le marché du microprocesseur se résume à celui du PC via les processeurs x86.

IMPORTANCE DU MARCHE DE L ’EMBARQUE

- La figure suivante démontre le contraire (année 1999) :



IMPORTANCE DU MARCHE DE L ’EMBARQUE

- Il a été vendu 108 millions de processeurs x86 pour le marché du PC contre 1,4 milliard de processeurs 8 bits pour le marché des systèmes embarqués (appelé aussi marché de l’embarqué) !
- On voit ainsi que 5 % des processeurs vendus sont pour le marché du PC. Dans 85 % des cas, Microsoft Windows est utilisé.
- Pour 95 % des autres processeurs vendus, on utilisera généralement un autre système d’exploitation (OS : *Operating System*).
- On trouvera ici dans 60 % des cas un OS propriétaire ; beaucoup optent pour des OS libres comme Linux pour limiter les coûts…

LE CHOIX D 'UN PROCESSEUR POUR L 'EMBARQUE

Embedded Processor	System Requirement	Feature	Benefit
Microcontroller	I/O Control	I/O Ports with bit-level control	Efficient control of external devices Direct interface to actuators, switches and digital status signals
	Peripheral Communication	Serial Ports : SPI, I²C, Microwire, UART, CAN	Hardware support for expansion & external device networking and communications
	Precision control of motors and actuators	Sophisticated timers and PWM peripherals	Low software overhead
	Quickly resolve complex software program control flow	Conditional jumps Bit test instructions Interrupt priority control	Efficiently implement control oriented algorithms
	Fast response to external events	External interrupts with multiple priority levels	Program control immediately redirected on event occurrence with minimal overhead
	Conversion of sensor data	Analog-to-Digital (A/D) Converters	Hardware support for external sensors

LE CHOIX D 'UN PROCESSEUR POUR L 'EMBARQUE

Embedded Processor	System Requirement	Feature	Benefit
DSP	Software Filters	Multiply/Accumulate Unit Zero-overhead loops	Digital filtering in few cycles
	Interface to codecs	High-speed serial ports	Hardware support for translation of analog signals
	High data Throughput from serial ports	Peripheral DMA	Less wasted cycles fetching data from serial ports
	Fast data access	Harvard architectures and variants	Fast execution of signal processing algorithms

LE CHOIX D 'UN PROCESSEUR POUR L 'EMBARQUE

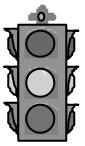
Besoin	Miniature	Petit	Moyen	Haut de gamme	PC embarqué	Embarqué haute disponibilité
Taille RAM	<0.1 Mo	0.1-4 Mo	2-8 Mo	8-32 Mo	16-64 Mo	> x Mo
Taille ROM/FLASH	0.1-0.5 Mo	0.5-2 Mo	2-4 Mo FLASH	4-16 Mo FLASH	xx Mo	Go-To
Processeurs	DragonBall 68K Mcore ColdFire ARM		MIPS Hitachi SH x86 PowerPC			Pentium PowerPC
Caractéristiques matérielles	MMU optionnelle		Ardoise Internet Carte unité centrale System on Chip (SoC)			CompactPCI
Exemples d'applications	Caméra numérique PDA Téléphone		Routeur Décodeur Stockage en réseau Imprimante en réseau			Commutateur téléphonique Routeur haute performance Serveur central

SYSTEME EMBARQUE : DEFINITION

- Un système embarqué peut être défini comme un système électronique et informatique autonome ne possédant pas des entrées/sorties standards comme un clavier ou un écran d'ordinateur (PC).

- Le système matériel et l'application sont intimement liés et noyés dans le matériel et ne sont pas aussi facilement discernables comme dans un environnement de travail classique de type PC.

SYSTEME EMBARQUE : DEFINITION



- Un système embarqué :
  - Est un système numérique.
  - Utilise généralement un processeur.
  - Exécute un logiciel dédié pour réaliser une fonctionnalité précise.
  - Remplace souvent des composants électromécaniques.
  - N 'a pas réellement de clavier standard (BP, clavier matriciel...). L 'affichage est limité (écran LCD…) ou n 'existe pas du tout.
  - N 'est pas un PC.
  - N 'exécute pas une application scientifique ou commerciale traditionnelle.

SYSTEME EMBARQUE : DEFINITION

- Différences avec un ordinateur de bureau :
  - L 'interface IHM peut être aussi simple qu 'une led qui clignote ou aussi complexe qu 'un système de vision de nuit en Temps Réel.
  - Des circuits numériques FPGA, ASIC ou des circuits analogiques sont utilisés en plus pour augmenter les performances du système ou sa fiabilité.
  - Le logiciel a une fonctionnalité fixe à exécuter et est spécifique à une application.

LES 4 TYPES DE SYSTEMES EMBARQUES

General Computing

• Application similaire à une application de bureau mais empaquetée dans un système embarqué.

• jeu vidéo, set- top box.

Control Systems

• Contrôle de systèmes en Temps Réel.

• Moteur d'automobile, process chimique, process nucléaire, système de navigation aérien.

Signal Processing

• Calcul sur de grosses quantités de données.

• Radar, Sonar, compression vidéo.

Communication & Networking

• Transmission d'information et commutation.

• Téléphone, Internet.

EXEMPLES DE SYSTEMES EMBARQUES

Office systems and mobile equipment	Building systems	Manufacturing and Process Control
Answering machines **Copiers** **Faxes** **Laptops and notebooks** **Mobile Telephones** **PDAs, Personal organisers** **Still and video cameras** **Telephone systems** **Time recording systems** **Printer** **Microwave**	**Air conditioning** **Backup lighting and generators** **Building management systems** **CTV systems** **Fire Control systems** **Heating and ventilating systems** **Lifts, elevators, escalators** **Lighting systems** **Security systems** **Security cameras** **Sprinkler systems**	**Automated factories** **Bottling plants** **Energy control systems** **Manufacturing plants** **Nuclear power stations** **Oil refineries and related storage facilities** **Power grid systems** **Power stations** **Robots** **Switching systems** **Water and sewage systems**

EXEMPLES DE SYSTEMES EMBARQUES

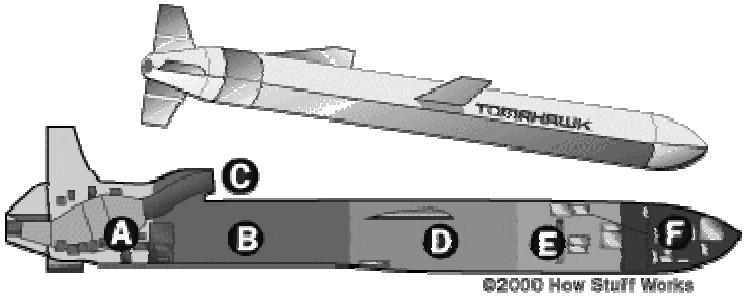
Transport	Communications	Other equipment
Aeroplanes **Trains** **Buses** **Marine craft** **Jetties** **Automobiles** **Air Traffic Control** **Signalling Systems** **Radar Systems** **Traffic Lights** **Ticketing machines** **Speed cameras, Radar speed detectors**	**Telephone systems** **Cable systems** **Telephone switches** **Satellites** **Global Positioning System**	**Automated teller systems** **Credit card systems** **Medical Imaging equipment** **Domestic Central Heating control** **VCRs**

EXEMPLE : WIRELESS

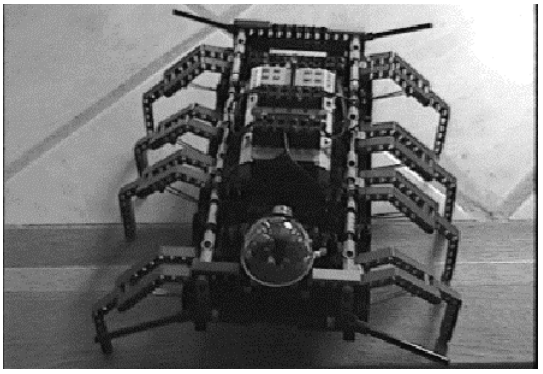
Hand-held GPS Units

Telematics System for Automobiles

EXEMPLE : GUIDAGE MISSILE

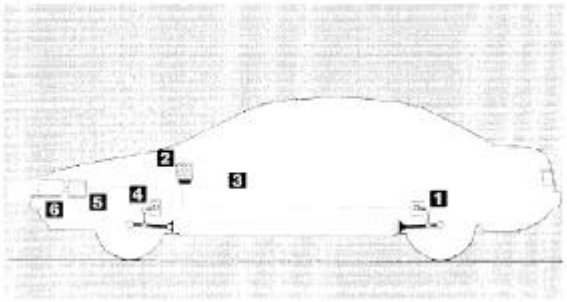


EXEMPLE : ROBOTIQUE



Spider robot – constructed with LEGO Mindstorms Components

EXEMPLE : AUTOMOBILE



Car with an automatic headlight leveling system. 1: Rear distance Sensor, 2: Control unit, 3: Speed signal, 4: Front distance sensor, 5: Motor, 6: Lamps.

CARACTERISTIQUES D ’UN SYSTEME EMBARQUE

- Fonctionnement en Temps Réel :
  - Réactivité : des opérations de calcul doivent être faites en réponse à un événement extérieur (interruption matérielle).
  - La validité d ’un résultat (et sa pertinence) dépend du moment où il est délivré.
  - Rater une échéance va causer une erreur de fonctionnement.
    - Temps Réel dur : plantage.
    - Temps Réel mou : dégradation non dramatique des performances du système.
  - Beaucoup de systèmes sont « multirate » : traitement d ’informations à différents rythmes.

CARACTERISTIQUES D ’UN SYSTEME EMBARQUE

- Faible encombrement, faible poids :
  - Electronique « *pocket PC* », applications portables où l ’on doit minimiser la consommation électrique (bioinstrumentation...).
  - Difficulté pour réaliser le packaging afin de faire cohabiter sur une faible surface électronique analogique, électronique numérique, RF sans interférences.
- Faible consommation :
  - Batterie de 8 heures et plus (PC portable : 2 heures).

CARACTERISTIQUES D ’UN SYSTEME EMBARQUE

- Environnement :
  - Température, vibrations, chocs, variations d ’alimentation, interférences RF, corrosion, eau, feu, radiations.
  - Le système n ’évolue pas dans un environnement contrôlé.
  - Prise en compte des évolutions des caractéristiques des composants en fonction de la température, des radiations...

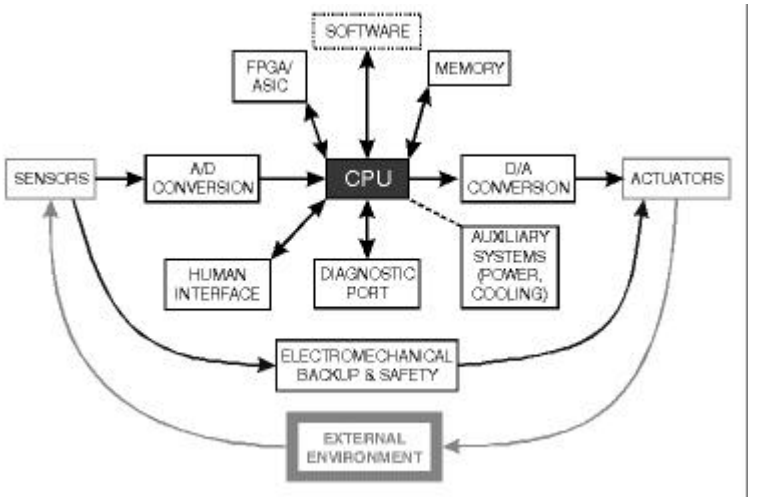
CARACTERISTIQUES D 'UN SYSTEME EMBARQUE

- Fonctionnement critique pour la sécurité des personnes. Sûreté :
  - Le système doit toujours fonctionner correctement.
  - Sûreté à faible coût avec une redondance minimale.
  - Sûreté de fonctionnement du logiciel
  - Système opérationnel même quand un composant électronique lâche.
  - Choix entre un design tout électronique ou électromécanique.

CARACTERISTIQUES D 'UN SYSTEME EMBARQUE

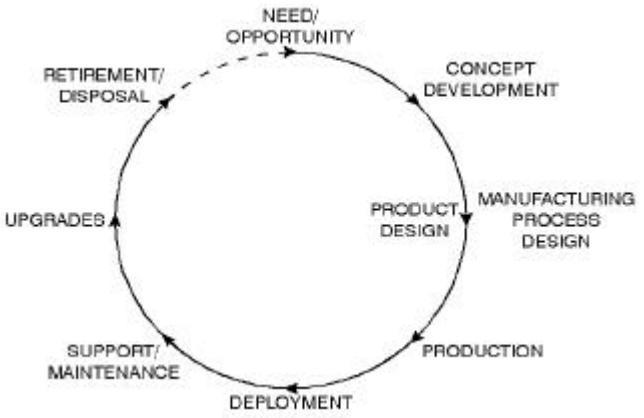
- Beaucoup de systèmes embarqués sont fabriqués en grande série et doivent avoir des prix de revient extrêmement faibles, ce qui induit :
  - Une faible capacité mémoire.
  - Un petit processeur (4 bits). Petit mais en grand nombre !
- La consommation est un point critique pour les systèmes avec autonomie.
  - Une consommation excessive augmente le prix de revient du système embarqué car il faut alors des batteries de forte capacité.
- Faible coût :
  - Optimisation du prix de revient.

SYSTEME EMBARQUE TYPIQUE



CARACTERISTIQUES D ’UN SYSTEME EMBARQUE

- Cycle de vie d’un système embarqué :



LES SYSTEMES EMBARQUES ET LE TEMPS REEL

- Généralement, un système embarqué doit respecter :
  - des contraintes temporelles fortes *(Hard Real Time*).
  - on y trouve enfoui un système d'exploitation ou un noyau Temps Réel (*Real Time Operating System*, RTOS).
- Le Temps Réel est un concept un peu vague. On pourrait le définir comme : "*Un système est dit Temps Réel lorsque l'information après acquisition et traitement reste encore pertinente*".

LES SYSTEMES EMBARQUES ET LE TEMPS REEL

- Cela veut dire que dans le cas d'une information arrivant de façon périodique (sous forme d'une interruption périodique du système), les temps d'acquisition et de traitement doivent rester inférieurs à la période de rafraîchissement de cette information.
- Pour cela, il faut que le noyau ou le système Temps Réel soit *déterministe* et *préemptif* pour toujours donner la main durant le prochain *tick* à la tâche de plus forte priorité prête.

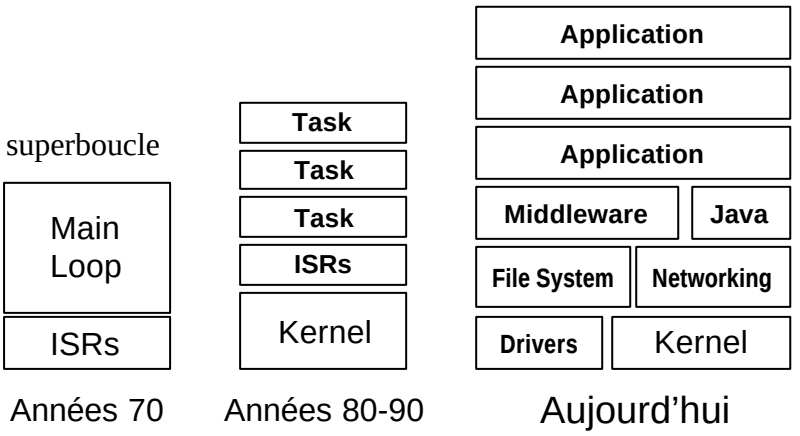
LES SYSTEMES EMBARQUES ET LE TEMPS REEL

- Une confusion classique est de mélanger Temps Réel et rapidité de calcul du système donc puissance du processeur (microprocesseur, microcontrôleur, DSP).
- On entend souvent : *" Être temps Réel, c'est avoir beaucoup de puissance : des MIPS, des MFLOPS… "*.

LES (RT)OS ET LES SYSTEMES EMBARQUES AUJOURD 'HUI

- La question d 'utiliser un système d 'exploitation Temps Réel ou non ne se pose plus aujourd 'hui pour des raisons évidentes :
  - Simplifications de l 'écriture de l 'application embarquée.
  - Portabilité.
  - Evolutivité.
  - Maîtrise des coûts.
  - …
- Le système d 'exploitation peut être même maison : encore dans 50 % des cas !

LES (RT)OS ET LES SYSTEMES EMBARQUES AUJOURD 'HUI



LES (RT)OS ET LES SYSTEMES EMBARQUES AUJOURD 'HUI

- La superboucle n 'est pas périmée aujourd 'hui mais reste réservée aux petits systèmes (8 bits).
- Le choix doit être bien sûr le bon choix pour minimiser les coûts du système. On n 'oublie pas les bonne recettes du passé !

LES SYSTEMES EMBARQUES ET LINUX

- Linux depuis presque 3 ans est en train de conquérir un domaine où on ne l'attendait pas vraiment : l'univers des systèmes embarqués.
- Pourquoi retrouve-t-on Linux dans l'embarqué ? Tout d'abord pour ses qualités qu'on lui reconnaît maintenant dans l'environnement plus standard du PC grand public :
  - Libre, disponible gratuitement au niveau source : pas de royalties à reverser.
  - Ouvert.
  - Différentes distributions proposées pour coller au mieux à un type d'application.

LES SYSTEMES EMBARQUES ET LINUX

- Pourquoi retrouve-t-on Linux dans l'embarqué ? Tout d'abord pour ses qualités qu'on lui reconnaît maintenant dans l'environnement plus standard du PC grand public :
  - Stable et efficace.
  - Aide rapide en cas de problèmes par la communauté Internet des développeurs Linux.
  - Nombre de plus en plus important de logiciels disponibles.
  - **Connectivité IP en standard**.

LES SYSTEMES EMBARQUES ET LINUX

- Linux a aussi d'autres atouts très importants pour les systèmes embarqués :
  - Portage sur processeurs autres que x86 : PowerPC, ARM, MIPS, 68K, ColdFire…
  - Taille du noyau modeste compatible avec les tailles de mémoires utilisées dans un système embarqué (<500 Ko).
  - Différentes distributions proposées suivant le domaine : routeur IP, PDA, téléphone…
  - Support du chargement dynamique de modules qui permet d'optimiser la taille du noyau.
  - Migration rapide et en douceur pour un spécialiste Linux à Linux embarqué ; ce qui réduit les temps de formation (et les coûts…).

LES SYSTEMES EMBARQUES ET LINUX

- On a en fait entendu parler pour la première fois officiellement de Linux embarqué à une exposition *Linux World* en 1999 où les sociétés Motorola, Force et Ziatech ont présenté un système CompactPCI fonctionnant sous Linux.

- En 2000 a été créé le consortium Linux embarqué (*Embedded Linux Consortium*) dont le but est de centraliser et de promouvoir les développements de solutions Linux embarqué. Ce consortium regroupe des éditeurs de distribution Linux, des éditeurs de systèmes Temps Réel propriétaires (comme WindRiver pour VxWorks) et des fabricants de composants. Il compte actuellement plus de 100 membres.

LES SYSTEMES EMBARQUES ET LINUX

- Les distributions Linux embarqué ont une part de marché grandissante face à des distributions propriétaires généralement Temps Réel comme VxWorks, pSOS, QNX… où l'on est d'abord obligé de payer pour accéder à la plateforme de développement puis de payer des royalties pour chaque système (ou cible) que l'on commercialise ensuite.

- Il est à noter que l'on observe une évolution de ce système à péage de certains face à la " menace " Linux.

LES SYSTEMES EMBARQUES ET LINUX

- Linux embarqué supporte aussi différentes extensions Temps Réel qui mettent en place une couche d'abstraction logique entre matériel, interruptions et Linux. Linux et l'ensemble des processus sont généralement considérés comme la tâche de fond exécutée quand il y a rien de Temps Réel à faire…

- On peut citer comme extensions Temps Réel :
  - La distribution RTLinux et sa distribution Mini RTLinux pour l'embarqué.
  - La distribution RTAI.

PARTIE 2 :
QUAND LE MATERIEL REJOINT LE LOGICIEL

CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

- La capacité de conception de systèmes numériques permet aujourd'hui de tout intégrer dans un même composant (concept du *single chip*).
- On travaille donc au niveau système et non plus au niveau porte élémentaire ou schématique. On parle de système sur silicium SoC (*System on Chip*) ou SoPC *(System on Programmable Chip*).
- Ceci est lié à la loi empirique de Moore qui dit que pour une surface de silicium donné, on double le nombre de transistors intégrés tous les 18 mois !

CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

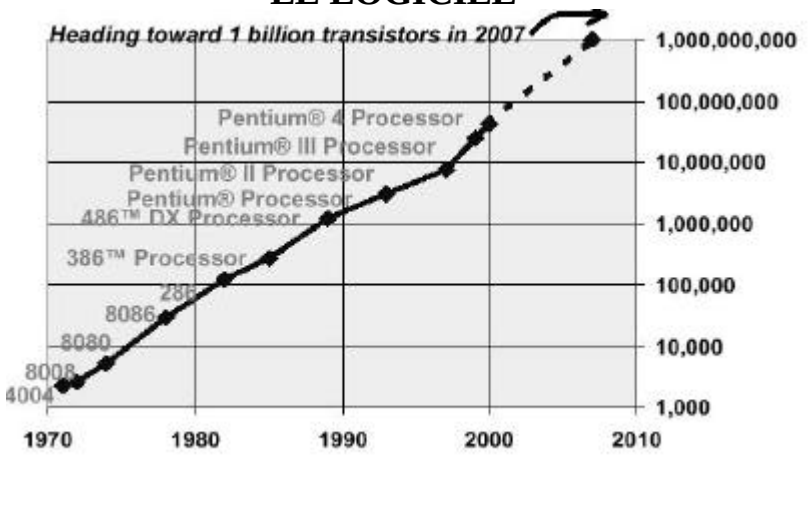
	1998	1999	2001
Technologie	0,25 µm	0,18 µm	0,15 µm
Complexité	1 M de portes	2-5 M	5-10 M

Loi de Moore

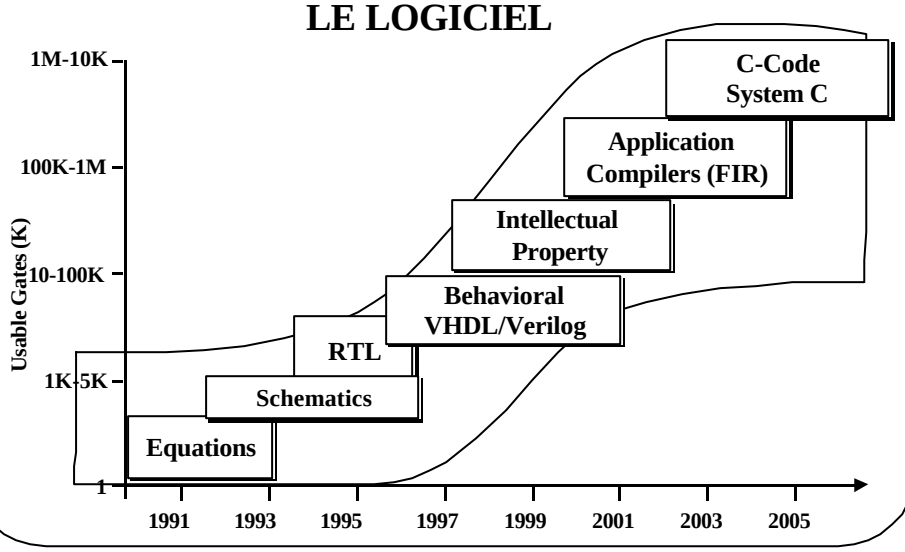
CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

	1997	1998	1999	2002
Technology	**350nm**	**250nm**	**180nm**	**130nm**
Cost	**$1.5-2.0billion**	**$2-3billion**	**$3-4billion**	**$4+ billion**
Design cycle	**18-12mo**	**12-10mo**	**10-8mo**	**8-6mo**
Complexity	**200-500k**	**1-2M**	**4-6M**	**10-25M**
Application	**Cellar phone, DVDs**	**Wireless PDAs**	**Internet appliances**	**Ubiquitous computing**

CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL



CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL



CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

- On utilise maintenant des langages de description du matériel (VHDL, Verilog) pour synthétiser et aussi tester les circuits numériques. On a ainsi une approche logicielle pour concevoir du matériel.

- Avec l'augmentation de l'intégration, les systèmes numériques se sont complexifiés alors que la mise sur le marché doit être la plus rapide possible :
  - Prise en compte du *Time To Market* (TTM).
  - Réutilisation de choses déjà réalisées (*Design Reuse*).

CODESIGN : QUAND LE MATERIEL REJOINT LE LOGICIEL

- On a ainsi vu apparaître la notion de blocs IP (*Intellectual Property*) qui est possible par l'utilisation des langages de description du matériel.

- On achète des blocs IP comme on achète un circuit intégré :
  - interface CAN.
  - DCT.
  - Interface MAC IEEE 802.3 10BaseT qui est la condition nécessaire pour assurer la connectivité IP sur réseau Ethernet.

PARTIE 3 :
CONNECTIVITE IP : LE BESOIN DE SYSTEMES COMMUNICANTS

CONNECTIVITE INTERNET : UNE INTRODUCTION

- La connectivité Internet permet de raccorder tout système électronique (système embarqué) au réseau Internet. On parle aussi de connectivité IP (*Internet Protocol*).

- Ajouter une connectivité IP à un système électronique permet de le contrôler à distance de n 'importe où dans le monde :
  - par une application réseau.
  - plus simplement par le « *web* » en utilisant un navigateur Internet (Netscape, Internet Explorer…).

CONNECTIVITE INTERNET : UNE DEFINITION

- Cette ultime (?) étape est l 'aboutissement du contrôle à distance d 'un système électronique :
  - par un terminal VT100 80x24 caractères relié par une liaison série RS.232/V.24 .
  - par une application graphique sur un PC relié par une liaison série.
  - par une application graphique X11 sur un PC ou une station de travail relié par une liaison Ethernet (ou par un bus de terrain).
  - par une application **graphique** de type navigateur *web* sur un équipement de contrôle (PC, station de travail, ordinateur de poche, téléphone portable…) relié à l 'Internet.

CONNECTIVITE INTERNET : UNE DEFINITION

- La connectivité IP demande d 'embarquer une suite de protocoles Internet sur le système électronique pour pouvoir être mise en œuvre.
- On parle alors de protocoles Internet embarqués (sur le système) ou plus simplement d 'Internet embarqué.
- La suite des protocoles IP à embarquer est moins ou moins importante en fonction du service à implanter :
  - contrôle par une application réseau spécifique.
  - contrôle par le web.
  - envoi d '*emails*.

CONNECTIVITE IP : UN PREMIER BILAN

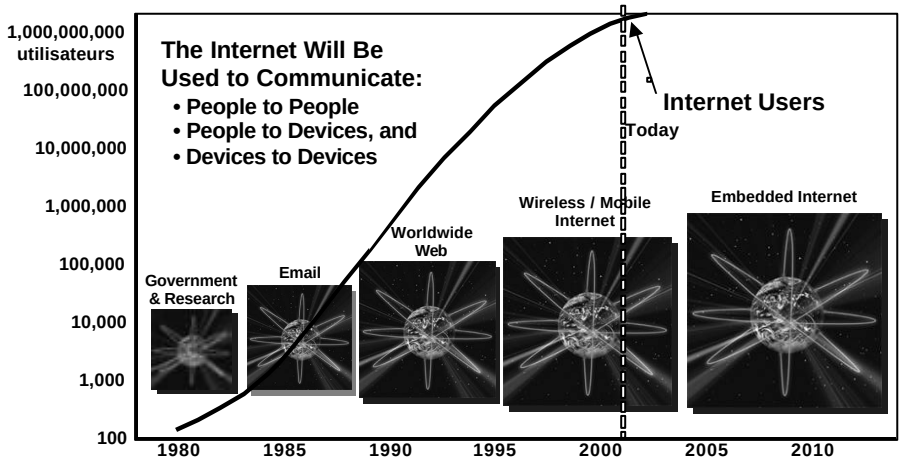
- La connectivité IP permet de raccorder tout système électronique (système embarqué) au réseau Internet. Elle met en œuvre une suite protocoles Internet que l 'on doit embarquer sur le système.
- Avec une frontière de plus en plus floue entre matériel et logiciel, on voit apparaître maintenant de véritables offres de ***codesign***. En conséquence, l 'ajout de la connectivité IP qui se faisait en grande partie en logiciel a tendance maintenant à être remplacée par son homologue matériel (utilisation d 'un bloc IP).

CONNECTIVITE IP : UN PREMIER BILAN

- La connectivité IP permet de contrôler un équipement électronique de n 'importe où dans le monde. Cet équipement peut à sont tour prévenir un opérateur n 'importe où dans le monde.
- La connectivité IP présume inconsciemment l 'utilisation d 'interfaces graphiques modernes et banalisées (navigateur web…) en adéquation avec les besoins (de confort) actuels des clients.
- C 'est en fait l 'aboutissement d 'un lent processus de modernisation du télécontrôle allant de la liaison série RS.232/V.24 déportée sur un terminal VT100 à l ' *applet Java* exécutée par un navigateur web interrogeant un serveur web embarqué !

CONNECTIVITE IP : UN PREMIER BILAN

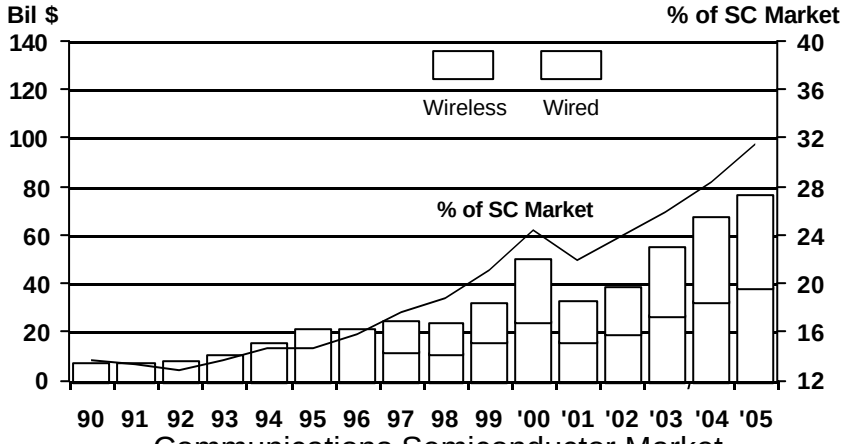
- Explosion du marché de l ' Internet embarqué



Source: Motorola, Network Wizards,Motorola, Microsoft, The State of the Net

CONNECTIVITE IP : UN PREMIER BILAN

- Importance croissante du *Wireless Embedded Internet*



Source: Motorola, WSTS, SPS & Dataquest

CHAPITRE 2 :
IMPORTANCE DU CODESIGN DANS L 'EMBARQUE

HARDWARE OU software ?

- Avec l 'intégration sur silicium de plus en plus importante, la difficulté est maintenant de savoir comment implémenter une fonctionnalité. Exemple d 'un algorithme de compression vidéo :
  - Plus rapide par hardware mais plus cher.
  - Plus flexible par logiciel mais plus lent.
- On doit être capable de jongler en plus avec les paramètres suivants :
  - Coût.
  - Rapidité.
  - Robustesse.
  - Packaging.

APPROCHE TRADITIONNELLE DE DEVELOPPEMENT D ’UN SYSTEME EMBARQUE

- 1. Choix du matériel (composants électroniques, processeur…) pour le système embarqué.
- 2. Donner le système ainsi conçu aux programmeurs.
- 3. Les programmeurs doivent réaliser un logiciel qui « colle » au matériel en n ’exploitant que les ressources offertes.

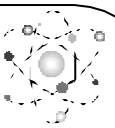
COMPLEXITE GRANDISSANTE D ’UN SYSTEME EMBARQUE

- Les systèmes embarqués sont de plus en plus complexes.
- Il est de plus en plus difficile de penser à une solution globale optimisée du premier jet.
- Il est de plus en plus difficile de corriger les « bugs ».
- Il est de plus en plus difficile de maintenir le système au cours du temps (obsolescence des composants…).
- En conséquence, l ’approche traditionnelle de développement d ’un système embarqué doit évoluer...

MOINS DE TEMPS POUR LE DEVELOPPEMENT

- Dans l ’électronique embarquée, le temps de conception devient de plus en plus réduit (pour diminuer les coûts, gagner un appel d ’offre).
  - Le temps développement de l ’électronique dans l ’industrie automobile est passée de 3 à 5 ans à 1 à 3 ans.
- Et le système doit toujours être toujours aussi sûr et robuste.

SOLUTION A LA COMPLEXITE



- Dans le processus de conception du système, on doit garder un niveau d ’abstraction important le plus longtemps possible.
- Le système doit pouvoir être décomposer en sous-systèmes suivant une hiérarchie logique (approche objet).
  - Si le design change, on doit pouvoir en réutiliser une bonne partie (*design reuse*).
  - Il ne faut pas jeter à la poubelle un précédent design et repartir «*from scratch* » !
- On doit pouvoir utiliser des outils de vérification automatique.

L ’INTERET DE L ’EMPLOI D ’UN PROCESSEUR

- L ’usage d ’un processeur facilite la réalisation du système dans la majorité des cas.
- Il se peut aussi que le processeur soit surdimensionné par rapport à la fonctionnalité logique à implémenter par logiciel. L ’usage de circuits logiques programmables FPGA (ou des ASIC) est alors intéressante dans ce cas.
- L ’usage d ’un processeur spécialisé comme un microcontrôleur ou un DSP peut aussi réduire considérablement le surdimensionnement et le nombre de composants électroniques.

LA QUADRATURE DU CERCLE

- Quand on conçoit un système embarqué, un certain nombre de contraintes apparaissent :
  - Quel hardware a-t-on besoin ?
  - Quel processeur choisir ? Quelle puissance CPU ?
  - Quelle type de mémoire et taille mémoire a-t-on besoin ?
  - Choix entre hardware rapide ou software intelligent ?
  - La consommation ? Minimiser les accès mémoire ? Choisir les instructions assembleur en fonction de leur consommation ?
  - Les contraintes de temps de conception seront-elles respectées ? Le TTM ?

LA QUADRATURE DU CERCLE

- Quand on conçoit un système embarqué, un certain nombre de contraintes apparaissent :
  - Est-ce que le système final marche correctement ?
  - Est-ce que les spécifications fonctionnelles ont été respectées ?
  - Comment doit-on tester les caractéristiques Temps Réel du système ? Doit-on le tester avec des données réelles ? Quelle plateforme de tests doit-on utiliser ?

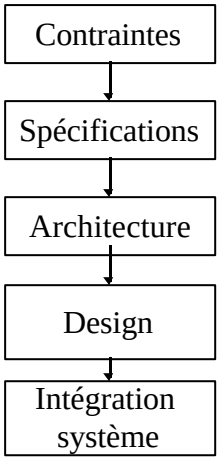
LA QUADRATURE DU CERCLE ET LE CONCEPTEUR DE SYSTEME EMBARQUE

- Concevoir un système embarqué remplissant toutes les contraintes précédemment énoncées demande au concepteur des compétences multidisciplinaires :
  - Compétences hardware et software : microprocesseur, microcontrôleur, DSP, FPGA, codesign, VDHL, assembleur, E/S, C, RTOS, réseau, UML, Linux, Java...
  - Connaissance des systèmes numériques.
  - Savoir travailler en équipe avec des ingénieurs d ’autres disciplines.
  - Comprendre le besoin du client et savoir aussi l ’identifier !

LA QUADRATURE DU CERCLE ET LA METHODOLOGIE DE CONCEPTION

- Une méthodologie dans la phase de conception du système doit être utilisée :
  - La méthodologie mise en œuvre permet d 'être sûr de ne pas oublier une étape.
  - Des outils de CAO et d 'aide à la conception peuvent être utilisés : automatisation et vérification en cours de conception.
  - Traçabilité des étapes de conception du système.
  - Maîtrise des contraintes : temps de conception, consommation, coût...

LES ETAPES DE CONCEPTION D 'UN SYSTEME



METHODOLOGIE TOP-DOWN ET BOTTOM-UP

- La méthodologie ***top-down*** part du niveau le plus abstrait (flux de données) pour aller à l 'entité matérielle ou logicielle élémentaire.
- La méthodologie ***bottom-up*** part de l 'entité matérielle ou logicielle élémentaire à implanter pour aller au système complet.
- A chaque niveau d 'abstraction, on doit analyser le design pour en déterminer les caractéristiques courantes pour ensuite les affiner au niveau d 'abstraction inférieur.

METHODOLOGIE TOP-DOWN ET BOTTOM-UP

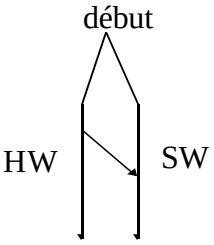
- La phase d 'intégration est primordiale car les « bugs » apparaissent à cette étape. Les tests d 'intégration doivent être les plus complets et sont donc consommateurs de temps donc d 'argent !
- Le *design reuse* est fortement conseillé et il vaut mieux de pas partir *from scratch*. C 'est en fait une des qualités intrinsèques des concepteurs de systèmes embarqués !

CODESIGN HARDWARE/SOFTWARE

- Le codesign dans la méthodologie de conception d ’un système embarqué est de plus en plus utilisé.
- Le codesign permet de concevoir en même temps à la fois le matériel et le logiciel pour une fonctionnalité à implémenter. Cela est maintenant possible avec les niveaux d ’intégration offerts dans les circuits logiques programmables.
- Le codesign permet de repousser le plus loin possible dans la conception du système les choix matériels à faire contrairement à l ’approche classique où les choix matériels sont faits en premier lieu !

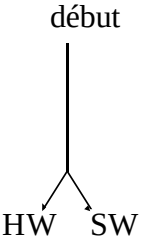
CONCEPTION ET CODESIGN

Conception traditionnelle



Réalisée par des groupes d’ingénieurs indépendants

Codesign (flot concurrent)

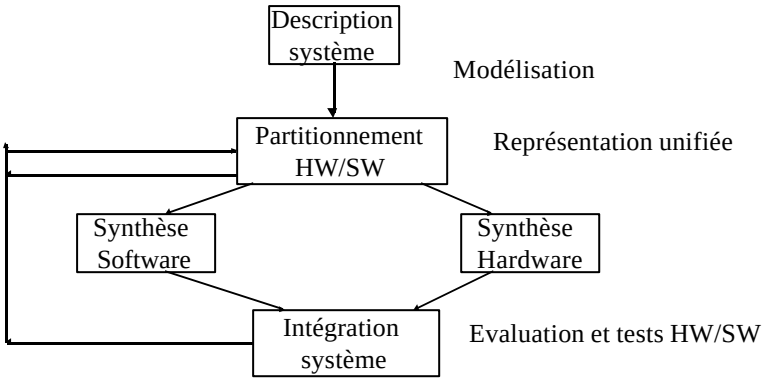


Réalisée par le même groupe d’ingénieurs en coopération

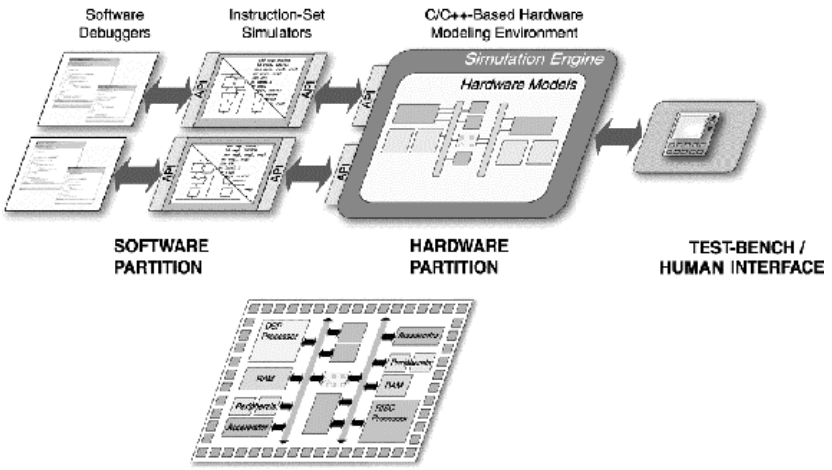
LES ETAPES DANS LE CODESIGN

- Durant le processus de codesign, on distingue les étapes suivantes :
  - Spécifications : liste des fonctionnalités du système de façon abstraite.
  - Modélisation : conceptualisation et affinement des spécifications produisant un modèle du matériel et du logiciel.
  - Partitionnement.
  - Synthèse et optimisation : synthèse matérielle et compilation logicielle.
  - Validation : cosimulation.
  - Intégration.
  - Tests d 'intégration.

LES ETAPES DANS LE CODESIGN



CODESIGN HARDWARE/SOFTWARE



AVANTAGES DU CODESIGN

- Le codesign est intéressant pour la conception de systèmes embarqués (ou de SoC) :
  - Amélioration des performances : parallélisme, algorithmes distribués, architecture spécialisée…
  - **Reconfiguration** statique ou dynamique en cours de fonctionnement.
  - Indépendance vis à vis des évolutions technologiques des circuits logiques programmables.
  - Mise à profit des améliorations des outils de conception fournis par les fabricants de circuits logiques programmables : synthèse plus efficace, performance accrue.

CODESIGN ET SYSTEMES EMBARQUES : UN BILAN

- Les systèmes embarqués étant de plus en plus complexes, un méthodologie rigoureuse de conception doit être maintenant mise en œuvre.
- Le codesign fait partie intégrante de cette méthodologie. Cela est maintenant possible grâce à des niveaux d 'intégration sur silicium très importants des circuits logiques programmables qui peuvent aujourd 'hui embarquer des processeurs.
- L 'apparition de modules IP, véritables circuits électroniques sous forme logicielle (« le *java bean* de l 'électronique ») n 'a fait qu 'accentuer l 'importance du codesign et du *design reuse*.

CHAPITRE 3 :
LA CONCEPTION ET LA MISE AU POINT DES SYSTEMES EMBARQUES.
LES OUTILS, METHODES, TRUCS ET ASTUCES

PARTIE 1 :
INTRODUCTION

INTRODUCTION

- La conception et la mise au point d ’un système embarqué est un processus difficile où le concepteur est vite en proie au doute et demande un certain talent.
- Ce chapitre se propose de donner quelques pistes et quelques règles de bon sens dans l ’art de mettre au point (« debugger ») un système embarqué :
  - Conception : mise au point du matériel.
  - Tests : mise au point du logiciel avec éventuellement modification du matériel.
- L ’expérience acquise lors de la mise au point de précédents systèmes est un capital inestimable...

INTRODUCTION

- La mise au point (debug) d ’un système est pénible, long et consommateur d ’argent.
- Dans la conception matérielle d ’un circuit SoC, cette étape de tests/debug (avec des patterns bien choisis) consomme 60 % du temps du projet.
- Du point de vue logiciel, on estime qu ’il reste encore 5 % de bugs après compilation statique d ’un code source. Sur un programme de 10000 lignes, cela représente 500 bugs avant inspection du code et tests. L ’inpection du code permet d ’en éliminer 70 à 80 %, ce qui en laisse encore 100. Après tests, on estime en avoir éliminé 50 %, ce qui en laisse 50 dans le produit fini ! (d ’après J. Ganssle)

INTRODUCTION

- Le calcul précédent est intolérable bien sûr (à moins de vouloir vendre des releases à tout coup) et l ’on a donc besoin d ’outils très performants et chers (analyseur de couverture du code…) pour faire tendre ce chiffre vers 0.
- Toutes les techniques de management et méthodologie n ’élimineront jamais le besoin de tester et de « debugger »…
- Le debug est ainsi très important pour ne pas « surinfecter » le produit fini...

PARTIE 2 :
LES OUTILS POUR LA MISE AU POINT D 'UN SYSTEME EMBARQUE

INTRODUCTION

INTRODUCTION

- Dans le processus de mise au point du système électronique, il est primordial de bien connaître la palette d ’outils à disposition.
- Mettre au point un système est d ’autant plus compliqué que l ’on travaille avec un matériel non encore testé...
- A chaque problème, il existe l ’outil approprié…il convient donc de les présenter...

L ’OSCILLOSCOPE

INTRODUCTION

- L ’oscilloscope est le premier outil de debug hard.
- Il permet de visualiser réellement un signal électrique :
  - Analogique.
  - Digitale (sans niveaux logiques comme avec l ’analyseur logique).

AVANTAGES ET INCONVENIENTS

- L ’oscilloscope permet de mettre en évidence parasites, glitchs…
- Il est limité à 4 sondes au plus soit 4 signaux.
- La logique de synchronisation (trigger) est très limitée.
- Il peut être intrusif en rajoutant une charge :
  - un circuit logique marche quand on branche une sond sur une broche puis ne marche plus quand on la débranche (charge capacitive).
- Il est important d ’avoir de bonnes sondes **calibrées** avec une grande bande passante pour ne pas déformer le signal.

L ’ANALYSEUR LOGIQUE

INTRODUCTION

- L ’analyseur logique espionne des signaux logiques : bus d ’adresses et de données, signaux de contrôle...
- On peut collecter les accès du processeur en mémoire et ensuite désassembler le code exécuté.
- On voit des adresses physiques externes donc si le cache d ’instructions ou de données est activé, on ne voit pas tout…
- Si les caches du processeur sont désactivés, il faudra faire attention au prefetch du processeur.

AVANTAGES

- L ’analyseur logique est un outil universel dans la mise au point de systèmes numériques. Il n ’est pas lié à un processeur particulier.
- L ’analyseur logique permet de mettre en place des triggers de déclenchement de l ’acquisition.
- Il est important de garder à l ’esprit (idem pour l ’émulateur ICE) que dans 90 % des cas, on choisit des conditions de trigger simples (booléen) par rapport aux triggers complexes tant vantés par les constructeurs ; ce qui en augmente le prix !

INCONVENIENTS

- Un analyseur logique est cher.
- Il y a le problème de la connectique et du nombre important de signaux à capturer.
- Pour désassembler le code exécuté, on est obligé de capturer les bus d ’adresses et de données ainsi que les signaux de contrôle soit un nombre très important de signaux logiques.
- Si l ’on a validé l ’optimisation du code (compilé ou assemblé), il peut être difficile de remonter au fichier source par désassemblage.

INCONVENIENTS

- Quand on branche l ’analyseur logique , on introduit une charge différente sur un bus. On peut ainsi masquer ou créer des problèmes.
- La charge peut réduire le bruit induit et l ’ensemble marche dans un environnement bruité. On enlève l ’analyseur et plus rien ne marche. Cela peut être aussi le contraire.
- L ’analyseur logique est donc intrusif.
- L ’analyseur logique est passif : on n ’a pas de contrôle sur le processeur.

INCONVENIENTS

- Il faudra faire attention aux caches validés.
- L ’analyseur logique fait un échantillonnage. Il convient donc d ’avoir une fréquence d ’acquisition suffisante et une taille mémoire d ’acquisition conséquente...

LE MONITEUR ROM

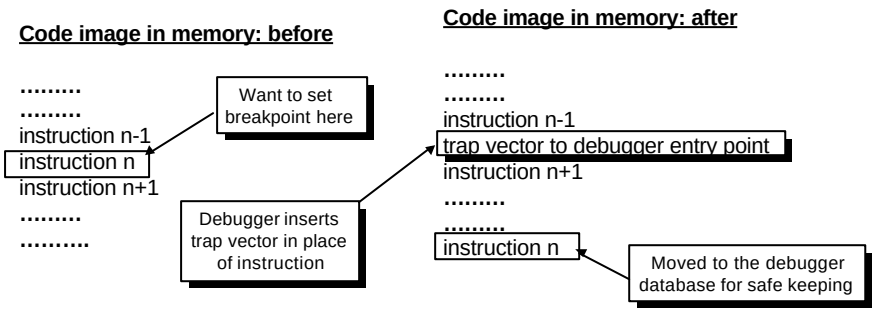
INTRODUCTION

- Le moniteur ROM est un programme de debug accessible depuis un port de communication (liaison série).
- Il dialogue avec une application debugger pour lui envoyer la valeur courante d ’un registre, d ’une adresse mémoire et placer un point d ’arrêt simple à une adresse donnée ou télécharger en mémoire du code.

POINT D ’ARRET

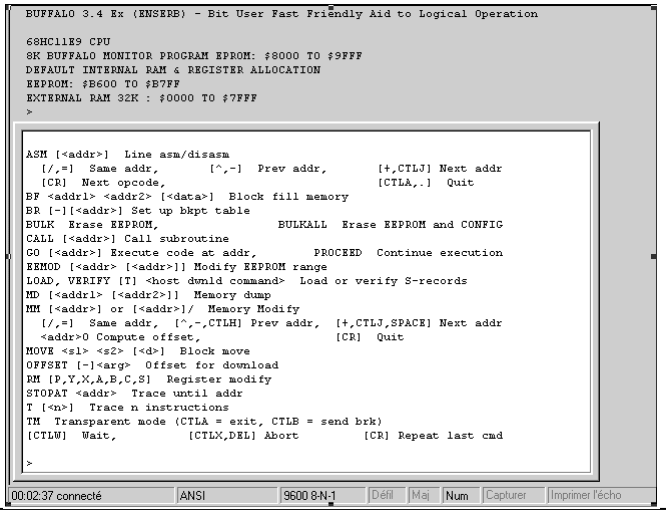
- Un point d ’arrêt (breakpoint) permet de dérouter l ’exécution normale du code de l ’application du système pour en examiner l ’état (du processeur, de la mémoire, des E/S...) ou le modifier.
- L ’instruction pointée par le point d ’arrêt (son adresse) est remplacée par une routine spécifique (ou un trap) :
  - Sauvegarde des registres du processeur.
  - Examen de l ’état du processeur.
  - Restauration des registres du processeur.

POINT D ’ARRET



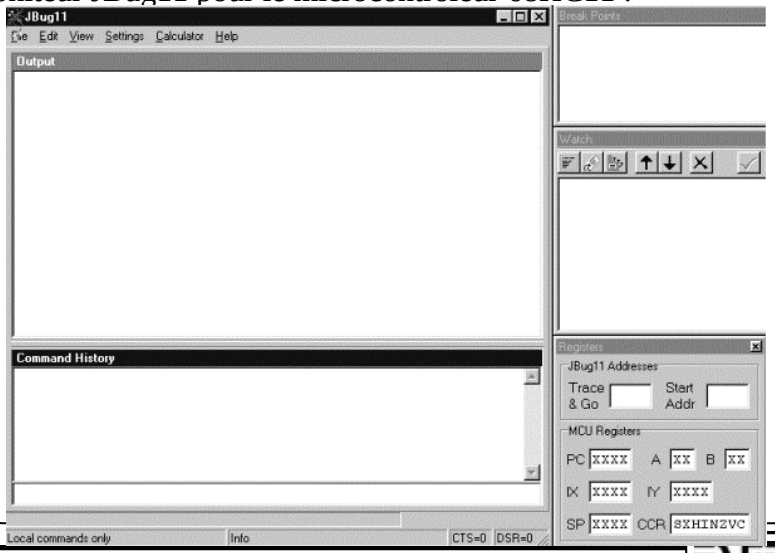
EXEMPLE DE MONITEUR

- Moniteur Buffalo pour le microcontrôleur 68HC11 :



EXEMPLE DE MONITEUR

- Moniteur JBug11 pour le microcontrôleur 68HC11 :



AVANTAGES

- Le moniteur ROM est bon marché.
- Il n ’y a pas de problèmes de connectique.
- Il n ’y a pas de problèmes de rapidité (c ’est du code exécuté).

INCONVENIENTS

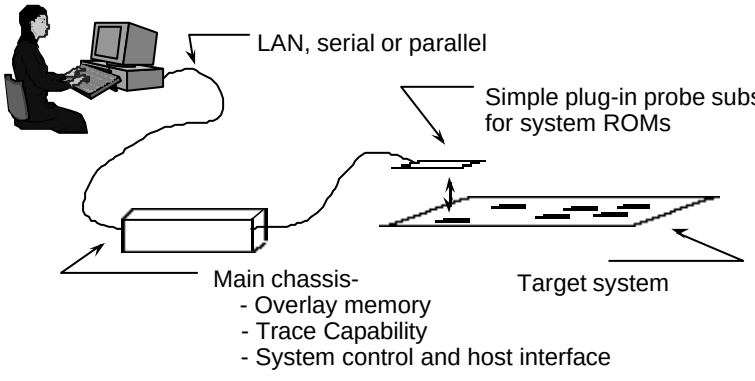
- Le moniteur ROM demande un port de communication exclusif à prévoir dans le design.
- Il consomme des ressources du système cible : RAM, ROM, interruptions ; ce qui peut être gênant pour un tout petit système.
- Il y a toujours des problèmes de configuration lors de la prise en main.
- Il ne marche pas s ’il y a des bugs matériels.
- Il ne possède pas ou peu de possibilités de debug en Temps Réel.

L ’EMULATEUR ROM

INTRODUCTION

- L ’émulateur ROM est un équipement qui se branche sur un support de mémoire ROM du système cible mais contient de la RAM au lieu de la ROM (Overlay RAM)
- Il possède :
  - Un câble de liaison en concordance avec les spécificités du support ROM.
  - De la RAM rapide.
  - Un contrôle du processeur (reset)…
  - Un port de communication avec le système hôte pour le debug.
- C ’est un moyen simple et rapide de télécharger du code dans la cible en lieu et place de la mémoire ROM initiale.

INTRODUCTION



AVANTAGES

- L ’émulateur ROM est bon marché.
- Il est compatible avec différents types de mémoires et n ’est donc pas lié à un type de processeur comme l ’émulateur ICE
- Il permet d ’ajouter un port de communication temporaire au système.
- Le téléchargement de code est rapide.
- On peut tracer l ’activité ROM en Temps Réel.

AVANTAGES

- On peut mettre un point d ’arrêt en ROM.
- Il peut être utilisé avec d ’autres outils de debug.

INCONVENIENTS

- L ’émulateur ROM a besoin d ’être de plus en plus rapide (comme l ’émulateur ICE) avec l ’évolution des processeurs.
- Il ne fonctionne qu ’avec une interface ROM : problème si l ’on a un microcontrôleur avec de la ROM interne.
- Beaucoup de systèmes font une recopie ROM vers RAM pour des soucis de performance donc l ’émulateur ROM est hors jeu.
- Il est inefficace en cas de problèmes matériels sur le système.
- Il est intrusif.

L ’EMULATEUR ICE

INTRODUCTION

- ICE est l ’acronyme de In Circuit Emulator.
- Pourquoi utiliser un émulateur ICE ?
  - Si le hardware du système n ’est pas parfait.
  - C ’est le meilleur outil pour détecter des problèmes matériels mais aussi logiciels.
  - Il n ’utilise pas de ressources du système cible.
  - Il supporte un debug en Temps Réel avec possibilités de trace, triggers…
- C ’est l ’outil parfait pour un intégration conjointe matérielle et logicielle.

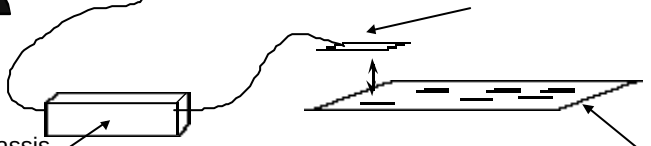
INTRODUCTION

- L ’émulateur ICE combine à la fois un debugger, un émulateur ROM et un analyseur logique.
- Il se substitue complètement au processeur qu ’il va émuler :

INTRODUCTION

Host computer runs emulator control software
- Provides run control
- Displays real time trace at source level
- Loads overlay memory with object code
- High-speed link to emulation chassis

Probe head contain emulation microprocessor
- Substitutes for, or disables target microprocessor
- Contains run control circuitry and cable buffers
- May contain memory mapping hardware



Main chassis
- Contains emulation (overlay memory)
- Trace analysis hardware and trace memory
- Performance analysis hardware (
- Power supply
- Control and communications

Target system

INTRODUCTION

- L ’émulateur ICE permet de contrôler le système :
  - Lecture/écriture en mémoire.
  - Modification des registres du processeur.
  - Pas à pas.
  - Points d ’arrêt.
  - Désassemblage de code.
  - Téléchargement de code.

INTRODUCTION

- L ’émulateur ICE se comporte comme un émulateur ROM :
  - Overlay RAM.
  - Possibilité de panacher mémoire Overlay RAM et mémoire du système cible.
  - Possibilité de fixer des attributs à des zones mémoire :
    - ROM : pas d’écriture possible.
    - Mémoire données, mémoire code.
    - Mémoire partagée : simulation d’E/S.

INTRODUCTION

- L ’émulateur ICE supporte l ’Overlay RAM : à l ’intérieur de l ’émulateur, la RAM est mappée dans l ’espace d ’adressage du processeur de la cible pour remplacer sa mémoire ROM ou FLASH. On peut ainsi télécharger rapidement du code à tester en Overlay RAM sans reprogrammer la mémoire ROM de la carte cible.
- Point d ’arrêt matériel : stoppe l ’exécution du programme sans affecter le contexte du processeur.
- Point d ’arrêt logiciel : insertion dans le code d ’un appel vers une routine de debug. Il y a un problème si le code est en mémoire externe ROM car pas d ’écriture possible.

TRACES TEMPS REEL

- L ’émulateur ICE permet de tracer l ’activité du processeur (comme l ’analyseur logique).
- Il admet des triggers complexes pour le lancement d ’une acquisition (trace pour la surveillance d ’une interruption) et désassemble le code capturé stocké dans une mémoire tampon circulaire.
- Si l ’on possède la table des symboles, on peut remonter au fichier source directement.

TRACES TEMPS REEL

Example of Real Time Trace from HP64700 emulator

```
Trace List                    Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:  Address   Data                  Opcode or Status                time count
Base:     hex     hex                       mnemonic                    absolute
_______ ________  ________ ____________ ____________________________ ____________
after    004FFA     2700     2700  supr data rd word                  ------------
+001     004FFC     0000     0000  supr data rd word                  + 520      nS
+002     004FFE     2000     2000  supr data rd word                  +   1.0    uS
+003     002000     2479  MOVEA.L 0001000,A2                         +   1.5    uS
+004     002002     0000     0000  supr prog                          +   2.0    uS
+005     002004     1000     1000  supr prog                          +   2.5    uS
+006     002006     2679  MOVEA.L 0001004,A3                         +   3.0    uS
+007     001000     0000     0000  supr data rd word                  +   3.5    uS
+008     001002     3000     3000  supr data rd word                  +   4.00   uS
+009     002008     0000     0000  supr prog                          +   4.52   uS
+010     00200A     1004     1004  supr prog                          +   5.00   uS
+011     00200C     14BC   MOVE.B  #000,[A2]                         +   5.52   uS
+012     001004     0000     0000  supr data rd word                  +   6.00   uS
+013     001006     4000     4000  supr data rd word                  +   6.52   uS
+014     00200E     0000     0000  supr prog                          +   7.00   uS
```


TRACES TEMPS REEL

- **The user interface software has access to the linker symbol table**
- **Can intersperse C or C++ source lines with assembly language execution**

set source on

display trace mnemonic

```
Trace List                    Offset=0      More data off screen (ctrl-F, ctrl-G)
Label:       Address       Data        Opcode or Status w/ Source Lines      time
Base:        symbols       hex              mnemonic w/symbols               rel
+009    sysstack:+003FC2     0738    0738  supr data wr word                  520
+010    sysstack:+003FC0     0006    0006  supr data wr word                  480
+011    main:main+00000A     01AA    01AA  supr prog                          520
       ##########main.c - line   104 ##############################################
            initialize_system();
+012    main:main+00000C     4EB9  JSR     |_initialize_sys                   480
+013    main:main+00000E     0000    0000  supr prog                          520
+014    main:main+000010     114A    114A  supr prog                          480
       ##########initSystem.c - line     1 thru    38 ############################
       void refresh_menu_window();

       void
       initialize_system()
       {
+015    |_initialize_sys     4E56  LINK    A6,#00000                          520

STATUS:   M68000--Running user program    Emulation trace complete_____.........
```

AVANTAGES

- L ’émulateur ICE est l ’outil roi pour la mise au point matérielle et logicielle.
- Il fournit toutes les fonctionnalités de debug nécessaires.
- Le contrôle du processeur est garanti quel que soit l ’état matériel du système cible.
- Il connaît les instructions en cache et il filtre les activités de prefetch du processeur.

INCONVENIENTS

- L ’émulateur ICE coûte très cher.
- Il est fabriqué pour un type de processeur donné. Si l ’on utilise un nouveau processeur, il faudra mettre à jour l ’émulateur ICE.
- Problème de la connectique, du prix et de la fragilité de la tête : adaptation à pourvoir si processeur CMS.
- Il ne supporte pas toujours toutes les variantes d ’un processeur donné (microcontrôleur).
- Les gens pensent que l ’émulateur ICE est difficile à utiliser.

INCONVENIENTS

- Le système cible doit être conçu pour supporter la tête de l ’émulateur ICE.
- Un émulateur ICE ne fonctionne pas toujours à la fréquence maximale du processeur.
- Quand on branche l ’émulateur ICE, on introduit une charge différente de celle du processeur. On peut masquer ou créer des problèmes.
- La charge introduite peut réduire le bruit induit et l ’ensemble marche dans un environnement bruité. On enlève l ’émulateur ICE et plus rien ne marche. Cela peut être le contraire.
- L ’émulateur ICE est donc intrusif.

JTAG

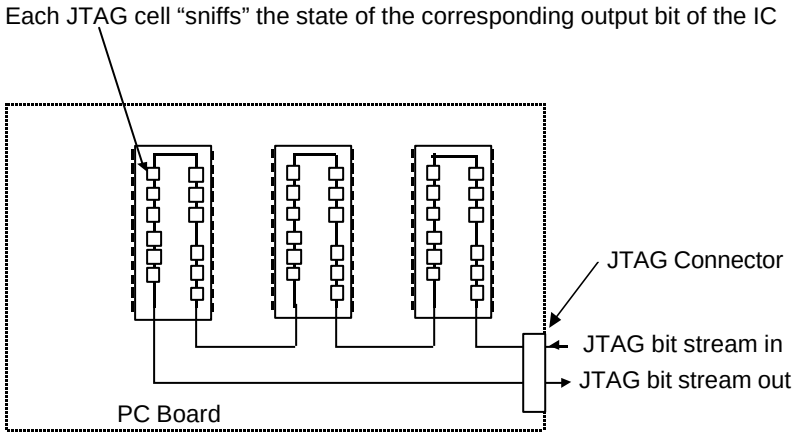
INTRODUCTION

- Le JTAG (Joint Test Action Group) a été originellement créé pour le BIST : Built In Self Test (autotests de circuits électroniques).
- C 'est un standard IEEE (1149.1) qui permet originellement de tester des circuits électroniques pour l 'industrie des cartes PC.
- Le JTAG possède 3 entités :
  - le contrôleur TAP (Test Access Port). Machine à 16 états.
  - Un registre à décalage d 'instructions.
  - Un registre à décalage de données.

INTRODUCTION

- Le JTAG forme une boucle série interconnectant les broches d 'E/S des circuits à surveiller/analyser (un point d 'entrée, un point de sortie).
- La boucle peut être active (écriture) et/ou passive (lecture). On récupère ainsi un flux de données série plus ou moins important. On va donc balayer périodiquement les composants JTAG (boundary scan).
- Chaque bit du registre à décalage de données du JTAG correspond à la valeur instantanée d 'une broche du circuit. On peut lire sa valeur courante ou lui affecter une nouvelle valeur (écriture).

INTRODUCTION

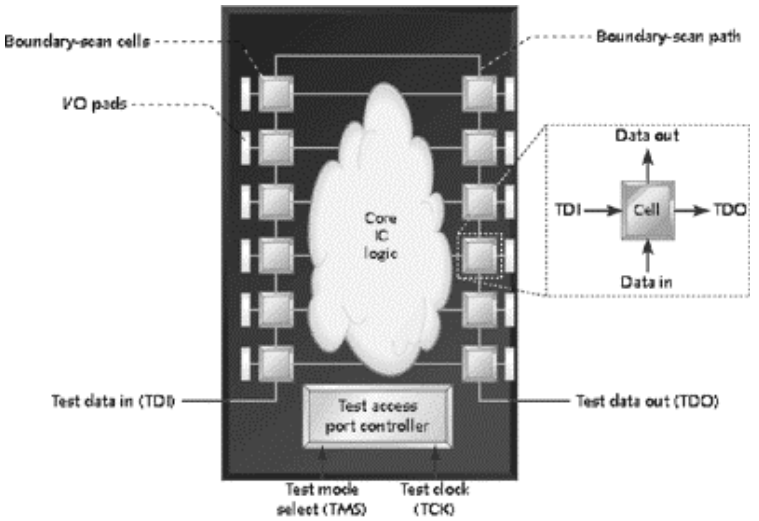


Bit stream forms one long shift-register

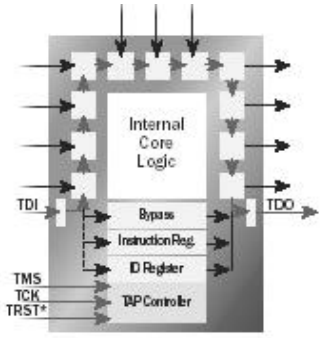
INTRODUCTION

- Le contrôleur TAP possède 4 broches :
  - TCK : horloge (Test ClocK).
  - TMS : mode test de contrôle du TAP (Test Mode Select).
  - TDI : entrée donnée série (Test Data In).
  - TDO : sortie donnée série (Test Data Out).

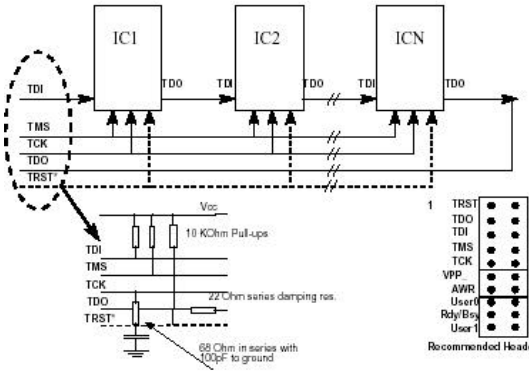
INTRODUCTION



INTRODUCTION



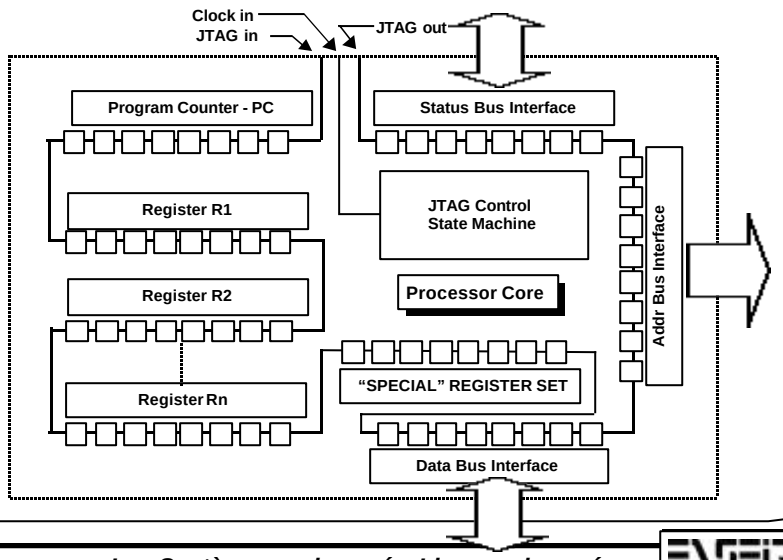
IEEE 1149.1 Device Architecture



DEBUG PAR LE JTAG

- Il est possible de mettre au point un système par le JTAG : la norme JTAG a été étendue par certains fondeurs pour cela.
- On lie tous les registres internes et les blocs fonctionnels importants par une boucle JTAG. On a donc un debugger simple sur silicium (on chip).
- On autorise la lecture comme l ’écriture dans les registres.
- On peut rajouter des registres de debug spécifiques accessibles par le JTAG.

DEBUG PAR LE JTAG



AVANTAGES

- Le debugger JTAG est un debugger on chip (OCD On Chip Debugger). On peut mettre des points d ’arrêt, faire du pas à pas, regarder la mémoire, changer une valeur en mémoire, télécharger du code…
- Le JTAG est un standard ouvert et non propriétaire comme le BDM.
- Il est adapté aux composants montés en surface.
- Il est bon marché.

AVANTAGES

- Il n ’ utilise pas beaucoup de broches car il met en œuvre un protocole série.
- Il permet d ’observer des entrées et positionner des sorties sans utiliser la logique du système.
- Il permet de réduire les points de tests sur le système.
- Il ne marche qu ’avec un processeur le supportant. Si un processeur supporte le JTAG pour le debug, rajouter toujours le connecteur JTAG sur le PCB (on ne sait jamais) ou le connecteur BDM !

AVANTAGES

- Il peut être utilisé aussi pour la programmation de mémoire flash embarquée sur le système.
- Il supporte éventuellement des point d ’arrêt matériels simples. Une broche spéciale suspend l ’activité du CPU dans l ’extension JTAG.

INCONVENIENTS

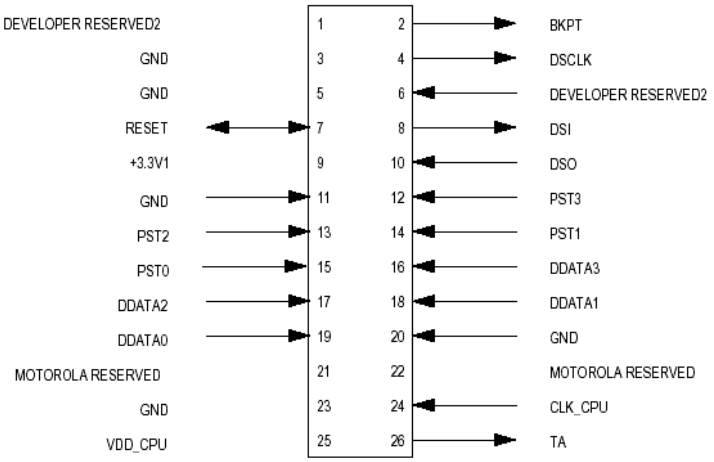
- Il peut être lent : chaque bit doit être décalé dans le registre à décalage. La boucle peut être grande (>10000 bits).
- Il n ’a pas toujours de points d ’arrêt hard.
- Il n ’y a pas de points d ’arrêt complexes comme avec l ’émulateur ICE.
- Il n ’y a pas d ’ Overlay RAM.

BDM

INTRODUCTION

- Le BDM (Background Debug Monitor) est un debugger on chip (OCD On Chip Debugger) propriétaire développé par Motorola pour ses processeurs CPU32 (683xx, ColdFire).
- Il utilise un connecteur 10 ou 26 broches sur le système cible.
- Un logiciel de debug sur l 'hôte dialogue via la liaison BDM avec le debugger OCD en envoyant des commandes de debug de (16+1) bits émis en série.

INTRODUCTION



NOTES: 1. Supplied by target
2. Pins reserved for BDM developer use. Contact developer.

PORTS DDATA ET PST

- Le BDM permet de contrôler le processeur et de le tracer en Temps Réel.
- Le debug par le BDM doit être validé (signal MTMOD=1 (Motorola Test MODe) pour le ColdFire).
- Les 4 bits de sortie (port DDATA) DDATA0 à DDATA3 correspondent aux données de debug tandis que les 4 bits de sortie PST0 à PST3 (port PST) correspondent à l ’état du processeur.
- Ces bits sont positionnés à chaque exécution d ’une instruction par le processeur.

PORTS DDATA ET PST

- Le port DDATA inclut 2 registres de 32 bits de stockage pour capturer l 'activité sur le bus interne (adresse et donnée). Ces 2 registres de 32 bits forment des FIFO lues par paquets de 4 bits.

- L 'exécution du processeur (dans le cas du ColdFire) est affectée quand il y a 2 données de 32 bits valides dans les FIFO. Il doit attendre qu 'une des FIFO soit vidée sur le port DDATA. S 'il n 'y a qu 'une FIFO valide, alors le processeur travaille à pleine vitesse. Cette attente n 'apparaît bien sûr quand on est en mode BDM (MTMOD=1).

PORTS DDATA ET PST

PST[3:0]		DEFINITION
(HEX)	(BINARY)	
$0	0000	Continue execution
$1	0001	Begin execution of an instruction
$2	0010	Reserved
$3	0011	Entry into user-mode
$4	0100	Begin execution of **PULSE** and **WDDATA** instructions
$5	0101	Begin execution of taken branch or Sync_PC[1]
$6	0110	Reserved
$7	0111	Begin execution of **RTE** instruction
$8	1000	Begin 1-byte transfer on DDATA
$9	1001	Begin 2-byte transfer on DDATA
$A	1010	Begin 3-byte transfer on DDATA
$B	1011	Begin 4-byte transfer on DDATA
$C	1100	Exception processing†
$D	1101	Emulator-mode entry exception processing†
$E	1110	Processor is stopped, waiting for interrupt†
$F	1111	Processor is halted †

NOTE: †These encodings are asserted for multiple cycles.
1 Rev. B ehnancement.

PORTS DDATA ET PST

- Exemple d 'activité sur les ports PST et DDATA lors de l 'exécution par un ColdFire (compatible avec le jeu d 'instructions du 68020) :

  JMP $1000 (adresse courte sur 16 bits)

PST	$5	$9	$0			
DDATA	$0	$0	A3-A0	A7-A4	A11-A8	A15-A12

PST $5 : début de l 'exécution d 'un branchement valide

PST $9 : début d 'un transfert de 2 octets sur le port DDATA (adresse de branchement ici soit $1000)

PST $0 : suite transfert

PORTS DDATA ET PST

- Dans le cas du processeur ColdFire (Voir le manuel utilisateur du 5206), les bits PSTx sont positionnés à 0100 ($4) lors de l 'exécution de l 'instruction PULSE ou WDDATA. Cela permet ainsi de générer un trigger depuis le code exécuté pour lancer une trace en Temps Réel.

- L 'instruction PULSE génère une marque de temps (trigger).

- L 'instruction WDDATA permet en plus de passer une donnée 32 bits quelconque sur le port DDATA.

DEBUG

- Les signaux du BDM DSCLK, DSI et DSO forment une liaison série synchrone pour passer des commandes de debug (DSI) et de récuperer des informations (DSO).

- Les commandes de debug et les données échangées forment des mots de 17 bits :
  - 16 bits : commande, paramètre, donnée.
  - 1 bit : contrôle/état. Non utilisé et laissé à 0.

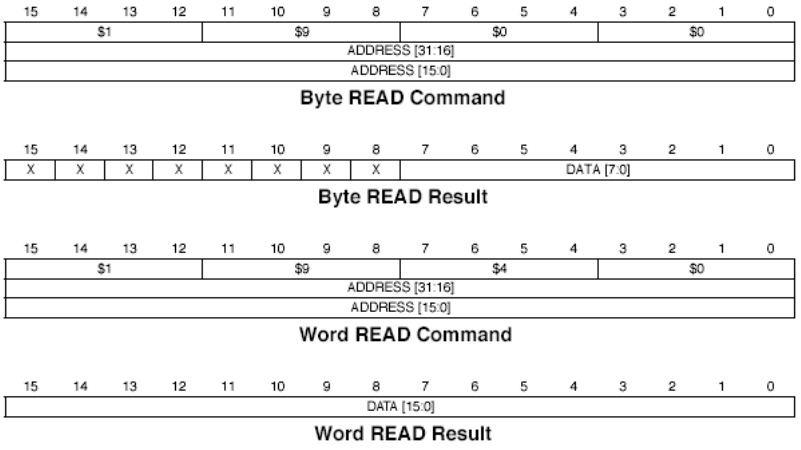
DEBUG

- Liste des commandes de debug :

COMMAND	MNEMONIC	DESCRIPTION	CPUIMPACT[1]
Read A/D Register	RAREG/RDREG	Read the selected address or data register and return the result via the serial BDM interface	Halted
Write A/D Register	WAREG/WDREG	The data operand is written to the specified address or data register via the serial BDM interface	Halted
Read Memory Location	READ	Read the sized data at the memory location specified by the longword address	Cycle Steal
Write Memory Location	WRITE	Write the operand data to the memory location specified by the longword address	Cycle Steal
Dump Memory Block	DUMP	Used in conjunction with the READ command to dump large blocks of memory. An initial READ is executed to set up the starting address of the block and to retrieve the first result. Subsequent operands are retrieved with the DUMP command.	Cycle Steal
Fill Memory Block	FILL	Used in conjunction with the WRITE command to fill large blocks of memory. An initial WRITE is executed to set up the starting address of the block and to supply the first operand. Subsequent operands are written with the FILL command.	Cycle Steal
Resume Execution	GO	The pipeline is flushed and refilled before resuming instruction execution at the current PC	Halted
No Operation	NOP	NOP performs no operation and may be used as a null command	Parallel
Read Control Register	RCREG	Read the system control register	Halted
Write Control Register	WCREG	Write the operand data to the system control register	Halted
Read Debug Module Register	RDMREG	Read the Debug module register	Parallel
Write Debug Module Register	WDMREG	Write the operand data to the Debug module register	Halted

DEBUG

- Commande READ :



AVANTAGES

- Le BDM est bon marché. C ’est un OCD (On Chip Debugger).
- Le BDM a les avantages de mise au point d ’un monitor (on Chip) et offre en plus beaucoup des fonctionnalités d ’un émulateur ICE.
- Comme la logique BDM est à l ’intérieur du processeur, on peut voir toute l ’activité CPU et les accès aux caches.

INCONVENIENTS

- Il n ’y a pas de points d ’arrêt complexes.
- Il n ’a pas toujours de points d ’arrêt hard.
- Il n ’y a pas d ’ Overlay RAM.

SIMULATEUR

SIMULATEUR

- Le simulateur (ISS Instruction Set Simulator) est un logiciel exécuté par le système hôte de développement pour simuler un processeur cible particulier.
- La principale difficulté est de simuler les périphériques réels du système cible.
- Les simulateurs sont plutôt réservés à de petits processeurs (microcontôleur) :
  - MPLAB pour le simulateur PIC Microchip.

LE CHOIX DES OUTILS

LE CHOIX DES OUTILS

- Le bon sens doit avant guider l 'ingénieur dans le choix des outils de debug à utiliser.

- On part généralement des outils les plus simples aux outils les plus sophistiqués :
  - L 'oscilloscope : vérification des horloges, signaux critiques…
  - L 'analyseur logique : reset, vérification des bus d 'adresses et de données, accès mémoire.
  - L 'émulateur ROM, ICE : couplage avec le logiciel. Debug au niveau source.
  - On peut ensuite embarquer un moniteur en ROM pour le lancement de l 'application finale ou faire des tests en cours de production.

LE CHOIX DES OUTILS

Fonctionnalité	**Emulateur ICE**	**BDM**	**ROM Moniteur**	**Analyseur logique**	**ROM Emulateur**
Event triggers	Oui	Certains	Non	Oui	Non
Overlay RAM	Oui	Non	Non	Non	Oui
Points d'arrêt hard	Oui	Certains	Non	Non	Certains
Points d'arrêt complexes	Oui	Non	Non	Oui	Non
Time stamps	Oui	Non	Non	Oui	Non
Accès non intrusif	Oui	Oui	Non	Oui	Non
Coût	Très cher	Bon marché	Bon marché	Cher	Bon marché

LE CHOIX DES OUTILS

Fonctionnalité	Emulateur ICE	BDM	ROM Moniteur	Analyseur logique	ROM Emulateur
Debug au niveau source	Oui	Oui	Oui	Certains	Oui
Téléchargement code	Oui	Oui	Oui	Non	Oui
Pas à pas	Oui	Oui	Oui	Non	Oui
Points d'arrêt	Oui	Oui	Oui	Non	Oui
Display/modifs registres	Oui	Oui	Oui	Oui	Oui
Regarder variables	Oui	Oui	Oui	Oui	Oui
Traces TR	Oui	Certains	Non	Oui	Non

PARTIE 3 :
LA MISE AU POINT MATERIELLE D 'UN SYSTEME EMBARQUE

ENVIRONNEMENT DE DEVELOPPEMENT

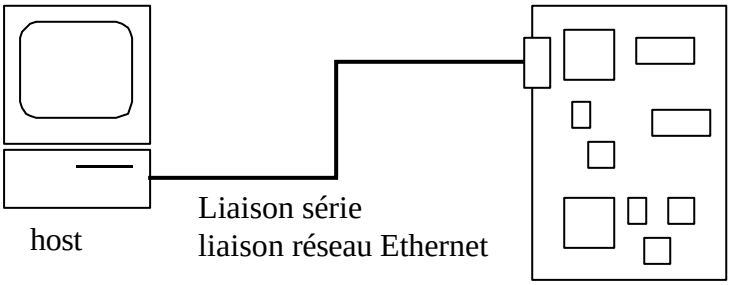
INTRODUCTION

- Développer et mettre au point une application pour un système embarqué est un art difficile à maîtriser pour une application allant de la simple superboucle (boucle plus interruptions) jusqu ’à un ensemble de processus coopératifs exécutés par un système d ’exploitation embarqué.

- On a généralement un environnement de développement croisé avec :
  - Une machine hôte (host) pour le développement et la mise au point.
  - Un système cible ou target dans lequel on va télécharger l ’application puis l ’exécuter dans la phase de mise au point.
- La mise au point fera appel aux outils précédents : émulateurs ICE ou ROM, BDM, JTAG, moniteur.

ENVIRONNEMENT DE DEVELOPPEMENT

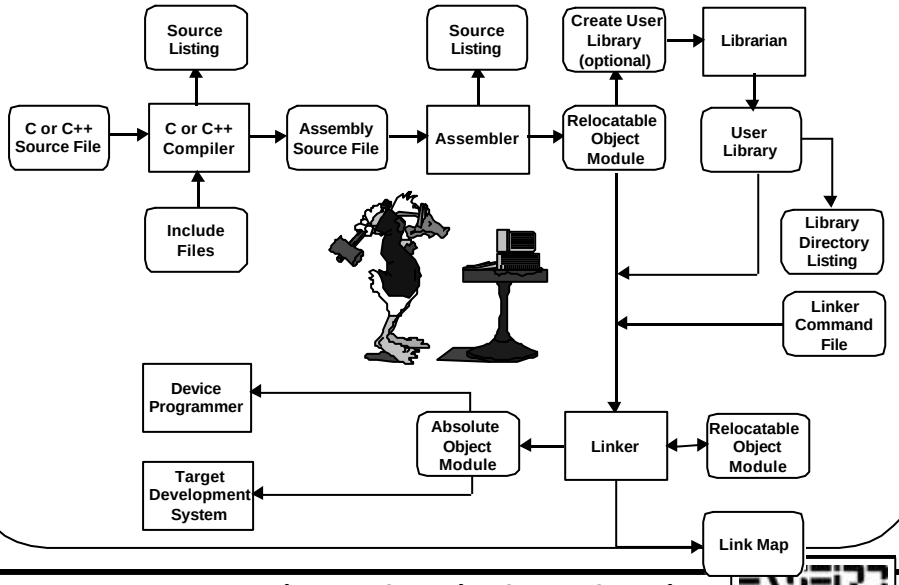
- On a un environnement de développement croisé :



ENVIRONNEMENT DE DEVELOPPEMENT

- On va développer et tester l ’application sur une plateforme standard (PC) avec des outils logiciels conviviaux afin de faciliter le debug.
- On aura un compilateur croisé sur l ’hôte pour avoir un code objet exécutable par le processeur de la cible.
- On aura un debugger croisé sur l ’hôte pour mettre au point l ’application exécutée par le processeur de la cible.

ENVIRONNEMENT DE DEVELOPPEMENT



ENVIRONNEMENT DE DEVELOPPEMENT

- Si l ’on travaille en langage évolué (langage C), on teste au niveau du source en langage évolué (source level debugging).
- Le debugger croisé (au niveau source) sera couplé avec un équipement de debug (ICE, émulateur ROM, BDM, JTAG...).
- Il doit supporter le mode trace en Temps Réel pour capturer le traitement d ’une interruption (ISR) sans ralentir le système.

TRUCS ET ASTUCES POUR UNE BONNE MISE AU POINT. AVANT DE COMMENCER

INTRODUCTION

- Avant de commencer quoi que ce soit, il convient d ’abord de préparer le terrain quand on doit développer le firmware.
- Il est important de savoir de quels outils on a besoin et s ’assurer qu ’il sont compatible avec le système à debugger.
- Les points à ne pas négliger :
  - Etre impliqué dans la conception du système.
  - Comprendre et maîtriser le hardware du système et être sympa avec le designer.
  - Avoir une copie de tous les documents.
  - Etre sûr que le matériel marche.
  - Commencer doucement (mais sûrement).
  - Regarder ce que l ’on vient de concevoir.

ETRE IMPLIQUE DANS LA CONCEPTION

- Vérifier que le composant sur lequel démarre (boot) le système est reprogrammable : pas de désoudage à faire, interface JTAG pour le reprogrammer…
- Le système doit inclure un mécanisme de communication entre le logiciel de boot (firmware) et l 'humain : leds, port série, connecteur spécial non équipé dans la version finale pour limiter le coût du produit fini, JTAG, BDM...

MAITRISER LE HARDWARE

- Etablir une bonne relation entre les gens du matériel et du logiciel, ne pas se jeter la balle :
  - « c 'est pas le soft qui plante, c 'est le hard…non c 'est pas le hard qui plante, c 'est le soft »…
- Passer un peu de temps avec le gens du hardware, poser des questions.
- Ne pas passer son temps à les « espionner ». Rester humble.
- Avoir une copie des schémas.

AVOIR UNE COPIE DES DOCUMENTS

- On doit connaître plus que les schémas.
- Avoir les datasheets de tous les circuits utilisés dans le système (travailler sur papier).
- Vérifier s ’il existe pas d ’errata pour chaque datasheet surtout quand un circuit récent est utilisé. Il est bien sûr possible alors de découvrir des bugs.

ETRE SUR QUE LE MATERIEL MARCHE

- Si le système est neuf, s ’assurer que les tests électriques élémentaires ont été faits :
  - Court-circuits.
  - Tests de continuité.
  - Pas de faux contacts.
  - Soudures froides.
- Savoir comment brancher l ’alimentation.

COMMENCER DOUCEMENT

- Faire de petits pas. Ne pas se lancer dans un énorme programme mais procéder par de petits programmes de tests pour tester un fonctionnalité bien particulière.

- Points à vérifier :
  - Est-ce que le programme est bien « mappé » en mémoire ?
  - Est-ce que la procédure de reset du CPU est bien comprise ?
  - Est-ce que le fichier objet est dans le bon format ?
  - Est-ce que le programmateur est bien configuré ?
  - Est-ce que le partitionnement de la mémoire de boot adresses paires/impaires est bien réalisé ?
  - Est-ce que le matériel marche réellement ?

COMMENCER DOUCEMENT

- Rechercher des exemples de code sur le site du fondeur du processeur.

- S ’abonner aux user groups adéquats.

- Rechercher des informations sur Internet.

REGARDER CE QUE L ’ON A CONCU

- Faire des dumps hexadécimal des fichiers objets.
- Consulter le fichier de mapping mémoire.
- Vérifier le mapping mémoire :
  - Vérifier les sections de code : .text
  - Vérifier les sections de données initialisées : .data
  - Vérifier les sections de données non initialisées : .bss (Bloc Starting by Symbol)

TRUCS ET ASTUCES POUR UNE BONNE MISE AU POINT. DANS L ’ACTION

INTRODUCTION

- Après le préambule précédent, il est possible de « toucher » au matériel.
- En cours d ’écriture du logiciel (firmware), il y a néanmoins des tests matériels à faire quelquefois quand on découvre des bugs matériels ou un firmware qui ne fait pas ce que l ’on veut.
- Il est important de faire alors appel à l ’équipe hardware pour confirmation et correction.
- Les procédures suivantes sont importantes à valider.

POINTS A VERIFIER

- Vérifier les tensions d ’alimentation :
  - Utiliser un voltmètre.
  - Utiliser un oscilloscope. Présence de bruit sur les alimentations ?
- Vérifier la présence d ’une horloge valide :
  - Utiliser un oscilloscope : présence d ’une horloge sur le CPU, bonne fréquence ?
- Vérifier la bonne assertion des « Chip Select » au boot (reset).

POINTS A VERIFIER

- Utiliser une loupe pour vérifier la bonne soudure des broches : pas de soudures froides, faux contacts.
- Faire attention aux décharges ESD. Utiliser un bracelet anti statique.
- Ecrire une simple boucle installée au vecteur de reset pour vérifier la bonne assertion du Chip Select de la mémoire ROM (68000, à installer au vecteur 0).
- S 'aider au besoin d 'un analyseur logique :

```
LOOP
        JUMP        LOOP
```


POINTS A VERIFIER

- Utiliser une led comme test (avec le programme à tester) si l 'on n 'a pas d 'analyseur logique ou d 'oscilloscope (ce qui serait surprenant !). Jouer sur la rapidité du clignotement, led allumée ou éteinte. Une fois ce point acquis, écrire 3 routines :
  - led_on()
  - led_blink_slow()
  - led_blink_fast()

  et s 'en servir comme routines de debug.
- Si l 'on a plus de leds, on peut indiquer un état de progression dans la phase de boot (3 leds soit 8 états).

POINTS A VERIFIER

- Tester la SRAM externe (patterns $55 et $AA comme tests grossiers).
- Tester la liaison série (programme d 'écho).
- Initialiser le contrôleur de DRAM.
- Et si tout va bien alors : programmer en langage évolué de type C.
- Utiliser un compilateur croisé C qui offre les routines C ANSI d 'E/S et modifier le getchar() et putchar() pour coller à son hardware. On a accès alors à son printf() préféré pour un debug simple...

POINTS A VERIFIER

- Faire attention aux données C initialisées au reset du système. Elles doivent donc être mise en mémoire non volatile. On ne peut donc pas les modifier en cours d 'exécution d 'un programme.
- Regarder la documentation du compilateur croisé et notamment la section chargeur (loader) en analysant plus particulièrement le fichier ASM crts.s.

PARTIE 4 :
LA MISE AU POINT LOGICIELLE D 'UN SYSTEME EMBARQUE

INTRODUCTION

- Il ne suffit pas savoir programmer, il faut savoir bien programmer !
- Un système embarqué doit être robuste et son code bien écrit.
- Le langage de prédilection pour le développement logiciel est le langage C.
- Le langage C est un langage de haut niveau mais proche du matériel.
- Cette partie met le doigt sur certains points de la programmation en langage C auxquels il faudra faire très **attention**.

GENERALITES

- Se méfier des options d ’optimisation du compilateur.
- Bien traiter les interruptions utilisées par le système en renseignant les vecteurs concernés dans la table des vecteur du processeur. Mise en place de la routine d ’interruption ISR (Interrupt SubRoutine).
- Renseigner toute la table des vecteurs même pour les interruptions non utilisées (ISR dummy…).

CLASSES DE STOCKAGE

- Pour un système embarqué, on est concerné par la façon dont le compilateur (C) va stocker les variables.
- On doit garder le contrôle du placement/mapping mémoire.
- La classe de stockage d ’une variable peut être :
  - auto : auto int a;
    - Classe de stockage par défaut pour une fonction.
    - Stockage sur la pile.
    - Variable détruite (désallouée) en fin d ’exécution de la fonction.

CLASSES DE STOCKAGE

- La classe de stockage d ’une variable peut être :
  - register : register int a;
    - Stockage de la variable dans un registre du processeur.
    - Accès à la variable très rapide.
    - Limitation par le nombre de registres de données.
  - static : static int a;
    - Contraire de auto.
    - Dans le cas d ’une telle variable de fonction, elle existe durant toute la durée d ’existence de la fonction.
    - Si c ’est une variable globale, elle existe durant toute la durée d ’exécution du programme.

CLASSES DE STOCKAGE

- La classe de stockage d ’une variable peut être :
  - extern : extern int a;
    - Variable définie dans un autre fichier source.
    - Variable importée.
    - Utilisé durant la phase d ’édition de liens pour résoudre les références croisées.

MODIFICATEURS D 'ACCES

- Il existe 2 modificateurs d 'accès à une variable :
  - const : const int a=10;
    - La valeur de la variable ne peut pas être changée par le programme.
    - A placer en mémoire ROM.
  - volatile : volatile char a;
    - La valeur de la variable peut changer d 'elle-même en cours d 'exécution du programme.
    - Registre d 'E/S mappé en mémoire.
    - Zone mémoire servant de buffer par un périphérique externe (contrôleur DMA).

MODIFICATEUR D 'ACCES VOLATILE

- Le modificateur d 'accès volatile est à utiliser **obligatoirement** :
  - Pour l 'accès à des registres de périphériques mappés en mémoire.
  - Variables globales modifiées par une ISR.
  - Variables globales dans une application multi-tâche.

- Source : Introduction to the Volatile Keyword.N.Jones, Embedded Systems Programming. http://www.embedded.com/story/OEG20010615S0107

VOLATILE : MAUVAISE PROGRAMMATION

- Exemple : scrutation active d ’un registre d ’état d ’un périphériques 8 bits.
- Le registre d ’état 8 bits est à l ’adresse 0x1234.
- Scrutation active du registre d ’état jusqu ’à ce qu ’il soit non nul.

VOLATILE : MAUVAISE PROGRAMMATION

```
unsigned char  * ptr = (unsigned char *) 0x1234;

// Wait for register to become non-zero
while (*ptr == 0)
   ;
// Do something else
```

- Cela risque de ne pas marcher dès que le compilateur optimise le code. Le code généré pourrait être :

```
        mov   ptr, #0x1234
        mov   a, @ptr
loop    bz    loop
```

VOLATILE : MAUVAISE PROGRAMMATION

- Le compilateur C voit qu ’il a déjà la valeur courante de la variable.
- Il n ’y a aucun besoin d ’aller la relire puisque c ’est la toujours même valeur (la valeur d ’une case mémoire ne change pas d ’elle même !).
- Il génère alors une boucle infinie.
- On ne sort donc pas de le boucle !
- D ’où un BUG en run time !

VOLATILE : BONNE PROGRAMMATION

```
volatile unsigned char * ptr = (volatile unsigned char *)
   0x1234;
```

- Le code assembleur généré est alors :

```
   mov      ptr, #0x1234
loop        mov   a, @ptr
            bz    loop
```

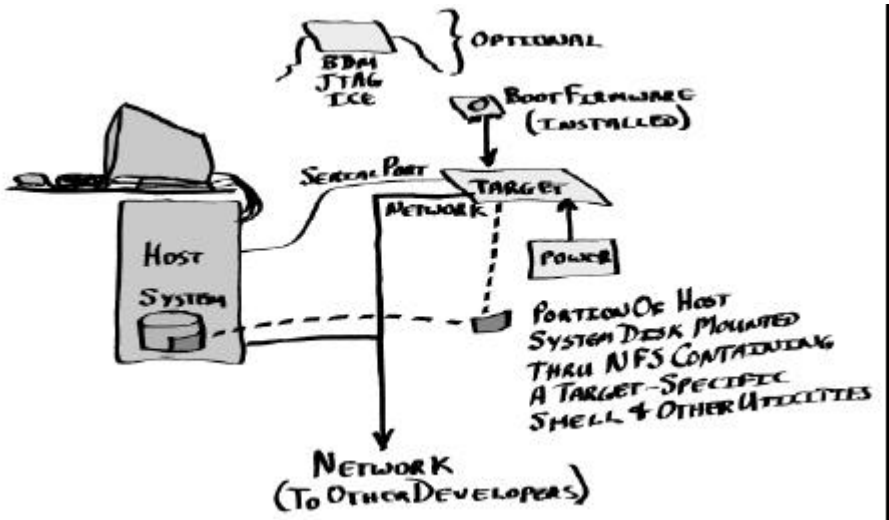
On a alors le comportement attendu !

PARTIE 5 :
EXEMPLE ILLUSTRE : MISE AU POINT LOGICIELLE D ’UN SYSTEME LINUX EMBARQUE

LINUX EMBARQUE : LES ETAPES

- 1. Développement du bootloader.
- 2. Portage du noyau Linux.
- 3. Développement des drivers spécifiques.
- 4. Développement de l ’application finale.
- 5. Intégration du tout.

L ’ENVIRONNEMENT



LE SYSTEME HOTE OU HOST

- Utilisation d ’un PC ou une station de travail comme host sous Linux en général (à défaut sous Windows avec Cygwin).
- Connexion par liaison série (ou par réseau Ethernet maintenant) avec le système embarqué cible ou target : minicom, hyperterminal
- Partage de fichiers avec la cible par montage NFS si la cible a une interface réseau Ethernet.
- Développement croisé sur l ’hôte avec tous les composants logiciels nécessaires.

LE SYSTEME CIBLE OU TARGET

- Utilisation d ’une carte CPU qui exécute Linux.
- On a besoin d ’y intégrer différents composants logiciels :
  - une boot ROM contenant un bootloader chargeant le noyau Linux en mémoire et puis l ’exécutant.
  - Un noyau Linux (compressé).
  - Une image disque d ’un système de fichiers Linux contenant les programmes applicatifs, bibliothèques partagées, modules Linux, fichiers de script...

TELECHARGEMENT DANS LA CIBLE

- Utilisation d ’un équipement particulier pour le téléchargement dans la cible en phase de développement (en final, tout sera « romé ») :
  - ICE, JTAG, BDM…
- On flashera en premier en mémoire FLASH le bootloader.
- Le bootloader pourra ensuite télécharger le noyau et l ’image du système de fichiers par la liaison série ou par le réseau (TFTP) (et le mettre en mémoire FLASH au final…).
- Le système de fichiers (root File System) peut être éventuellement monté par NFS au lancement du noyau Linux de la cible...

ENVIRONNEMENT DE DEVELOPPEMENT CROISE

- Environnement de développement croisé sous Linux. On utilisera les outils GNU :
  - gcc, binutils.
  - ld, as, nm, obcopy, strip, ar…
- Ces outils permettent de créer une image binaire du noyau et du système de fichiers (root file System) sur l 'hôte.
- On pourra utiliser un atelier de génie logiciel (IDE : Integrated Design Environment) : Kdevelop, Metrowerks Code Warrior...

ENVIRONNEMENT DE DEVELOPPEMENT CROISE

- Utilisation du debugger GNU ou GDB.
- Debug par la liaison série ou par le réseau. On a sur la cible un programme serveur gdbserver qui attend les ordres d 'une application cliente exécutée par l 'hôte. On bénéfie alors d 'un interface graphique conviviale sur l 'hôte qui permet de debugger au niveau source :
  - utilisation d 'un « front end » à gdb comme DDD (Data Display Debugger).
- On pourra aussi mettre à profit les outils de base Linux sur l 'hôte pour la gestion du projet :
  - make, CVS (Current Version System)...

DEVELOPPEMENT VS PRODUCTION

- En phase de développement, on a à disposition des outils :
  - Qui permettent un téléchargement rapide.
  - Qui permettent de télécharger le noyau Linux.
  - Qui permettent de télécharger l 'application par NFS.
  - Qui permettent de développer, de tester sur l 'hôte puis de porter sur la cible.
- En phase de production, le bootloader, le noyau et son root File System sont romés en mémoire FLASH :
  - Taille totale ?
  - Comment mettre à jour (correction de bugs…) ?

QUOI DEVELOPPER ?

- On doit développer :
  - Le système électronique (procédure classique).
  - Le bootloader.
  - Le portage du noyau.
  - Les pilotes de périphériques.
  - L 'application.
  - L 'intégration du tout.

DEVELOPPEMENT DU BOOTLOADER

- Développement du bootloader :
  - En utilisant les sources éventuels fournis avec la carte si on l 'a achetée.
  - En utilisant un bootloader open source : U-boot, RedBoot, Colilo…
  - Se l 'écrire soi-même.
- Le bootloader est en mémoire FLASH. Il prend en charge l 'initialisation de base du système, les autotests, la décompression de l 'image+root FS stockés en mémoire FLASH ou téléchargé (liaison série ou réseau).
- Il peut avoir des fonctions de debugger (moniteur).

PORTAGE DU NOYAU

- Il est important de comprendre comment marche le noyau et notamment la procédure de boot.
- Il faut choisir le portage correspondant au type de processeur de la cible.
- Il faut modifier éventuellement les sources du noyau (fichiers ASM) pour coller au hardware de la carte cible.

DEVELOPPEMENT DES DRIVERS

- Il est souhaitable de choisir des composants qui sont supportés par Linux lors de la phase de conception du système.
- Lors de la phase de configuration du noyau, on choisira les drivers appropriés. On pourra éventuellement les modifier pour coller au mapping mémoire de la cible.
- Si l 'on doit écrire un nouveau driver pour un matériel spécifique, il est préférable de partir d 'un driver existant similaire.
- Il faudra faire le choix entre développer un driver statique lié au noyau ou un driver dynamique (module Linux) chargé en mémoire par le noyau à la demande.

DEVELOPPEMENT DE L 'APPLICATION

- Il est important de voir s 'il n 'existe pas déjà une application existante collant à son besoin. Dans le cas éventuel, la porter sur sa cible. Dans le meilleur des cas, cela se traduira par une simple recompilation croisée.
- Choisir de préférence des applications open source.
- Considérer le développement d 'une interface graphique (GUI Graphical User Interface) si besoin :
  - système de type Xwindow.
  - Frame buffer.
  - Microwindows, Qt/embedded...

INTEGRATION SYSTEME

- Il faudra veiller à :
  - Evaluer la taille mémoire consommée (RAM, FLASH)
  - Prendre en compte la faible empreinte mémoire :
    - Chargement du noyau en RAM ou XIP (eXecute In Place).
    - Root File System en RAM ou FLASH.
    - Type du système de fichiers : ext3, JFFS2 pour un système embarqué.
    - Bibliothèques : statiques ou dynamiques.
    - Bibliothèques libc light (µClibc…).

CHAPITRE 4 :
LA CONNECTIVITE IP

PARTIE 1 :
RAPPELS SUR LA NOTION DE RESEAU

INTRODUCTION

- La connectivité IP met en œuvre différents protocoles Internet qu ’il convient de présenter (de façon générale). Qui fait quoi ?
- Une introduction réseau sera donnée pour mieux comprendre l ’imbrication des différents protocoles IP.
- Internet est un réseau de transmission de données et est basé en partie sur le modèle OSI des systèmes ouverts qu ’il faut introduire.

RAPPELS RESEAU

- Le modèle OSI est le modèle d'interconnexion des systèmes ouverts (OSI) de l'Organisation de Standardisation Internationale (ISO) (norme ISO 7498 en 1983).
- Le modèle OSI est une base de référence pour identifier et séparer les différentes fonctions d'un système de communication (vue de l'esprit, modèle logique).
- Un réseau de communication est basé sur une structure en couches.

OSI : Open System Interconnexion
ISO : International Standardisation Organism

RAPPELS RESEAU

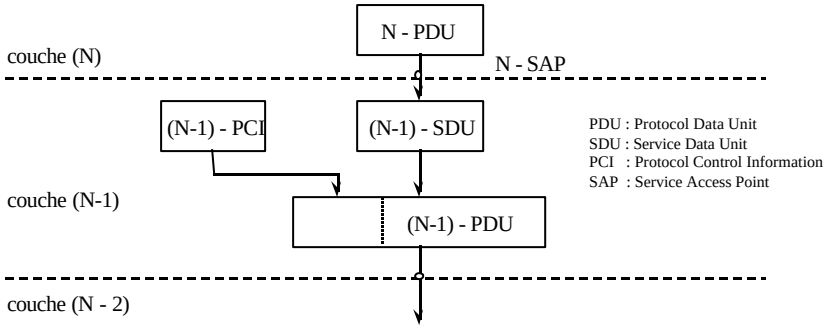
- Le modèle OSI est un modèle hiérarchique à plusieurs couches ou niveaux :
  - Une couche est créée quand un niveau d'abstraction est nécessaire.
  - Chaque couche exerce une ou plusieurs fonctions précises.
  - Le choix des frontières entre chaque couche doit limiter le flux de données échangées.
  - Le nombre de couches doit être suffisant pour éviter de faire cohabiter dans une même couche des fonctions trop différentes.

MODELE OSI A 7 COUCHES

RAPPELS RESEAU

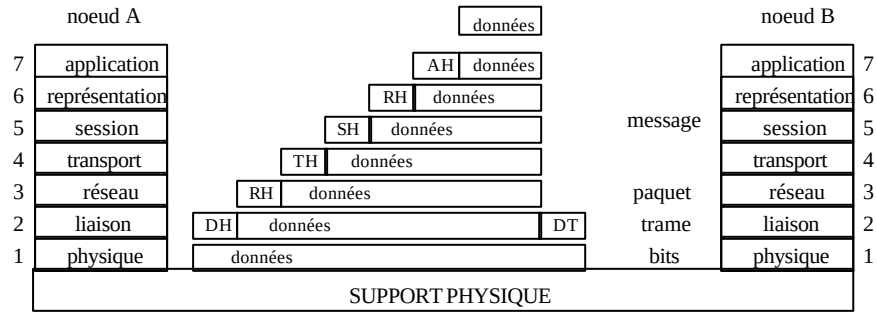
- Le modèle OSI ne propose pas une architecture de réseau universelle.
- Le modèle OSI décrit seulement ce que chaque couche doit réaliser.
- L'ISO a quand même proposé des normes (protocoles) pour ces couches (HDLC, LAP-B...) reprises par l 'UIT-T.
- Chaque couche assure un ensemble de fonctions spécifiques :
  - Chaque couche utilise les services de la couche immédiatement inférieure pour rendre à son tour un service à la couche immédiatement supérieure.
  - Une entité est l'élément actif d'une couche (matériel, logiciel).
  - Les entités d'une même couche sur 2 noeuds différents sont des entités paires ou homologues.

RAPPELS RESEAU



- Application du principe d'encapsulage des données passées d'un niveau à un autre («poupées russes») :
  - Encapsulage des données vers les niveaux inférieurs (**émission**).
  - Désencapsulage des données vers les niveaux supérieurs (**réception**).
  - Diminution du débit utile.

RAPPELS RESEAU



- Le modèle OSI possède 7 couches :
  - ☞ Couches 1 à 4 : **couches basses** chargées d'assurer un transport optimal des données.
  - ☞ Couches 5 à 7 : **couches hautes** chargées du traitement des données (représentation, cryptage...).

RAPPELS RESEAU

- Couche 1 ou couche physique :
  - ☞ Mode de représentation des données (bits) ou codage.
  - ☞ Spécifications mécaniques et électriques.
  - ☞ Synchronisation, détection erreur bit.

- Couche 2 ou couche liaison de données :
  - ☞ Assure un premier niveau de contrôle de la transmission en offrant un service de transmission sécurisé.
  - ☞ Structuration des données sous forme de trames.
  - ☞ Détection et correction (par retransmission) des erreurs. et non corrigées par le niveau 1 (utilisation de codes détecteur/ correcteur d'erreur CRC pour erreur trame).

CRC : Code de Redondance Cyclique

RAPPELS RESEAU

- Couche 3 ou couche réseau :
  - ☞ Routage et acheminement des données formatées en paquets à travers les différents noeuds du réseau (notion d'adresse).
  - ☞ Gestion de la congestion dans le réseau.

- Couche 4 ou couche transport :
  - ☞ Gestion du dialogue entre les 2 noeuds actifs.
  - ☞ Formatage des données sous forme de messages adaptés au niveau 3.
  - ☞ Deux modes de connexion :
    - mode connecté : connexion de bout en bout sécurisé avec multiplexage de voies possible (ex : TCP) .
    - mode non connecté : service datagramme non fiable (ex : UDP).

RAPPELS RESEAU

- Couche 5 ou couche session :
  - ☞ Structuration du dialogue entre la session établie (break...).

- Couche 6 ou couche représentation :
  - ☞ Représentation des données manipulées par les 2 applications communicantes (format, compression, cryptage...).

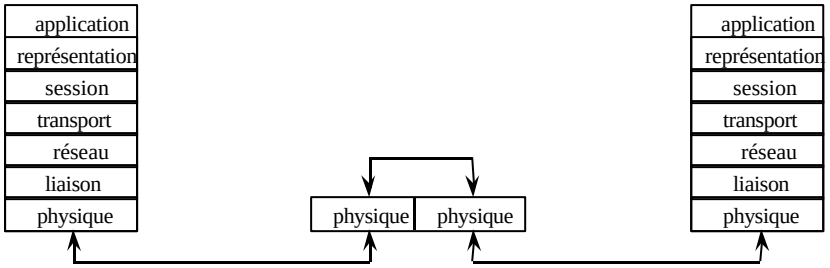
- Couche 7 ou couche application :
  - ☞ Interface entre l'application de l'utilisateur et le service de communication.
  - ☞ Définition d'applications normalisées (messagerie...).

RAPPELS RESEAU

- Nécessité d'ajouter des éléments dans un réseau de communication :
  - ☞ Extension du réseau (plus de noeuds, plus long).
  - ☞ Connexion vers un autre type de réseau.

- Différents types d'équipements mis en œuvre suivant le niveau du modèle OSI considéré.

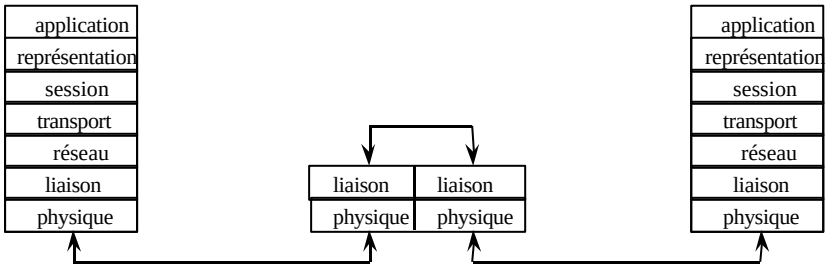
RAPPELS RESEAU



- Répéteur ou amplificateur («repeater») :
  - ☞ Amplification du signal pour augmenter la distance.
  - ☞ Conversion de signaux (RS-485 vers fibre optique).

RAPPELS RESEAU

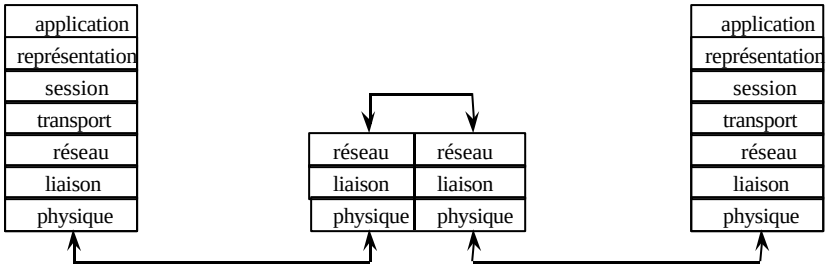
Interconnexion



- Pont (*Bridge*) :
  - ☞ Conversion de signaux (couche 1) et de format des trames du niveau liaison (couche 2).

RAPPELS RESEAU

Interconnexion



- Routeur (*router*) :
  - ☞ Conversion de format des paquets et notamment des adresses.
  - ☞ Routage des paquets suivant adresse entrante vers des liens prédéfinis (sous-réseau ou *subnetwork*) (routeur IP).
  - ☞ Système intelligent (diminution du débit).

RAPPELS RESEAU

Critères de classification

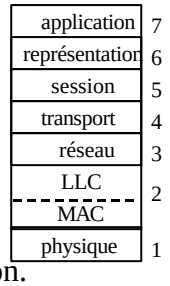
- On peut classer un réseau suivant différents critères :
  - ☞ Distance entre les éléments les plus éloignés.
  - ☞ Débit maximum.
  - ☞ Nombre maximum de nœuds.
  - ☞ Protocoles mis en œuvre (méthode d'accès au médium).
  - ☞ Topologie.

- Les différentes topologies possibles sont :
  - ☞ Anneau (*ring*).
  - ☞ Etoile (*star*).
  - ☞ Bus.
  - ☞ Arbre (*tree*).
  - ☞ Quelconque.

RAPPELS RESEAU

Méthodes d'accès au médium

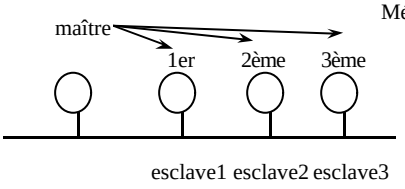
- Importance fondamentale de la couche liaison de données.



- Division en 2 sous-couches :
  - ☞ Sous-couche LLC (*Logical Link Control*) :
    - * Filtrage des messages.
    - * Recouvrement des erreurs bit / trame.
    - * Notification de surcharge (*overrun*).
  - ☞ Sous-couche MAC (*Medium ACcess*) :
    - * Mise en trame, gestion émission / réception.
    - * Détection / signalisation erreur bit.
    - * Arbitrage : gestion des accès simultanés sur le médium car collisions possibles (temps de latence).
    - * Importance de la topologie.

RAPPELS RESEAU

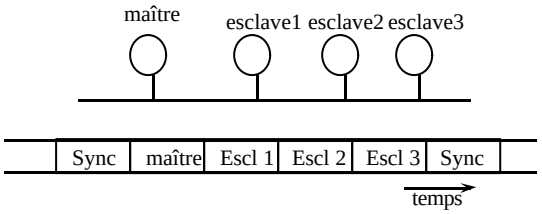
Méthodes d'accès au médium



- Accès par «polling» :
  - ☞ Un noeud maître consulte périodiquement les noeuds esclaves par un message de polling leur donnant le droit d'émettre.
  - ☞ Système centralisé (Maître/Esclave).
  - ☞ Point faible : maître.
  - ☞ Peu efficace.
  - ☞ Communication entre esclaves possible via le maître.

RAPPELS RESEAU

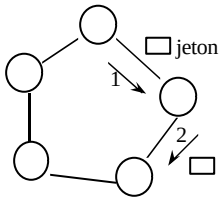
Méthodes d'accès au médium



- Accès par multiplexage temporel (*Time Division Multiple Access*) :
  - ☞ Emission d'un mot de synchronisation par le noeud maître.
  - ☞ Emission des données par tous les noeuds à un Intervalle de Temps précis (IT).
  - ☞ Taille des données fixe.
  - ☞ Meilleur efficacité que le polling .

RAPPELS RESEAU

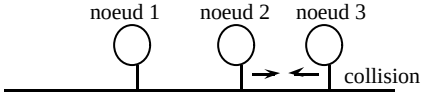
Méthodes d'accès au médium



- Accès jeton sur anneau (*Token Ring*) :
  - ☞ Topologie en anneau.
  - ☞ Circulation d'une trame particulière (jeton) de noeud en noeud quand pas d'émission.
  - ☞ Le noeud désirant émettre sur le médium garde le jeton, émet sa trame puis rend le jeton.
  - ☞ Connexion point à point, déterminisme.
  - ☞ Problèmes si médium rompu, perte ou duplication de jeton.
  - ☞ Variante : bus à jeton.

RAPPELS RESEAU

Méthodes d'accès au médium



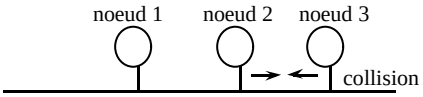
(ex : bus de terrain)

- Accès CSMA/CA (*Carrier Sense Multiple Access/Collision Avoidance*)
  - ☞ Les noeuds attendent un blanc avant d'émettre.
  - ☞ Chaque trame possède un identificateur (peut être l'identificateur du noeud).
  - ☞ On distingue le bit dominant du bit récessif.
  - ☞ Accès multiples possibles.
  - ☞ En cas de contention, l'arbitrage se fait sur les bits de l'identificateur («Bitwise Contention») (OU câblé).
  - ☞ Introduction de priorités.
  - ☞ Efficacité importante.
  - ☞ L'arbitrage introduit une longueur max du réseau :

$$time\_bit > 2\ t\_prop\_bus = 2\ l\_bus/v$$

RAPPELS RESEAU

Méthodes d'accès au médium



(ex : Ethernet)

- Accès CSMA/CD (*Carrier Sense Multiple Access/Collision Detection*) :
  - ☞ Les nœuds attendent un blanc avant d'émettre.
  - ☞ Si plusieurs émissions simultanées, détection de la collision.
  - ☞ Accès multiples possibles.
  - ☞ En cas de contention, l'arbitrage se fait par durée d'attente aléatoire pour chaque nœud en collision.
  - ☞ Problème si charge élevée.
  - ☞ Peu déterministe (risque de blocage).

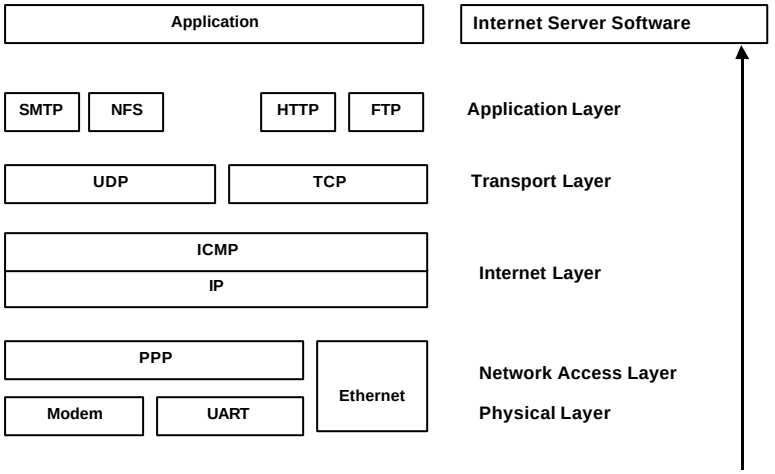
PARTIE 2 :
RAPPEL SUR LES PROTOCOLES INTERNET

ARCHITECTURE DU RESEAU INTERNET

- Architecture en 4 couches :
  - médium (1 et 2 du modèle OSI).
  - réseau IP (sans connexion) (3 du modèle OSI).
  - transport TCP (avec connexion) ou UDP (sans connexion) (4 du modèle OSI).
  - application (5, 6 et 7 du modèle OSI).
- Fonctionnalité majeure :
  - Interconnexion de réseaux hétérogènes.

TCP : Transmission Control Protocol
UDP : User Datagram Protocol
IP : Internet Protocol

ARCHITECTURE DU RESEAU INTERNET



ARCHITECTURE DU RESEAU INTERNET

- Les protocoles Internet sont indépendants du support de transmission ou médium choisi :
  - Ethernet : le plus commun.
  - Liaison série (pour accès à Internet par le RTC).
  - Liaison radio : GSM, BLR : on parle de *Wireless Internet.* Cette technologie est jeune et en cours de développement…
  - Courant porteur.

- Cette indépendance par rapport au médium en fait son intérêt et son universalité. On peut donc interconnecter des réseaux hétérogènes par Internet. On dit que l ’on met IP sur tout (IP over ATM, IP over FR…).

C ’est LE standard de fait.

ADRESSAGE

- Chaque ordinateur (ou système embarqué) est repéré de façon unique par une adresse Internet ou adresse IP. C ’est une adresse sur 32 bits (4 octets). L ’adresse est indispensable dans tout réseau de communication !

- Notation décimale pointée de 4 nombres entiers :
  - (1 par octet : nombre entre 0 et 255).
  - Ex : 147.210.18.138

- Une adresse IP comprend 2 champs :
  <id. réseau> <id. machine>

ADRESSAGE

- Le premier octet code la classe de réseau. Ce découpage en classe permet de cataloguer les différents réseaux.
- Classe A pour les très grands réseaux de plus de 2^{16}=65536 ordinateurs.
- Classe B pour les réseaux de 65536 ordinateurs maximum.
- Classe C pour les réseaux de 256 ordinateurs maximum.
- On trouve principalement en France des réseaux de classe B et C.
- La gestion des adresses est faite par INTERNIC. En France, c 'est l 'INRIA.

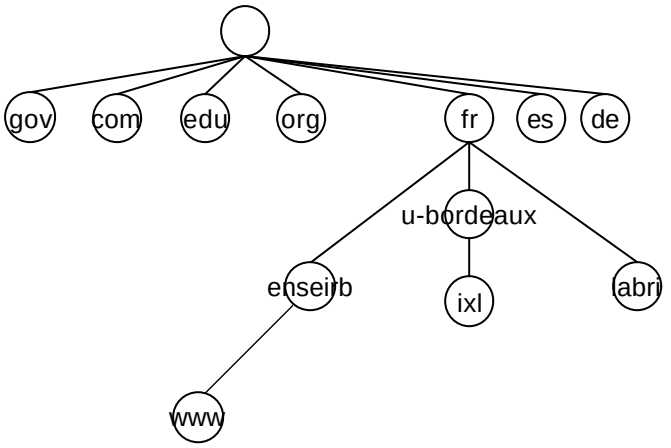
ADRESSAGE

- Certaines combinaisons sont réservées :
  - ex : <id. réseau> <id. machine 0> : sert à identifier le réseau lui-même.
  - ex : 147.210.18.0
  - ex : <id. réseau> <id. machine 255> : *broadcast*.
  - ex : 147.210.18.255
- Certaines plages d 'adresses sont réservées :
  - intranet : réseau d'ordinateurs proches non reliés à l 'Internet mondial.
  - ...

DNS (*Domain Name System* RFC 1034)

- Les humains préfèrent les noms symboliques (chaîne de caractères) aux adresses IP : rôle du DNS.
- DNS : correspondance entre un nom symbolique et une adresse IP.
- Le DNS est un espace hiérarchisé de noms symboliques.
- Chaque nœud a un nom d'au plus 63 caractères (la racine a un nom nul).

DNS



COUCHE LIAISON INTERNET

- C ’est l ’équivalent des couches 1 et 2 du modèle OSI. Les protocoles Internet au dessus ne voient pas les spécificités propres à chaque médium. Les protocoles IP sont indépendants du support de transmission.

- Le but est :
  - envoyer/recevoir des datagrammes IP.
  - envoyer/recevoir des requêtes ARP/RARP.

COUCHE LIAISON INTERNET

réseau	**IP, ARP, RARP**
liaison	**MAC(CSMA/CD), SLIP, PPP**
physique	**Ethernet, V.24, RTC, xDSL RNIS, radio, courant porteur**

(R)ARP : (Reverse) Address Resolution Protocol
CSMA/CD : Carrier Sense Medium Access/ Collision Detect
MAC : Medium ACCess
PPP : Point to Point Protocol (RFC 1548)
SLIP : Serial Link IP (RFC 1055)

RTC : Réseau téléphonique Commuté
RNIS : Réseau Numérique à Intégration de Services
xDSL : x Digital Subscriber Line

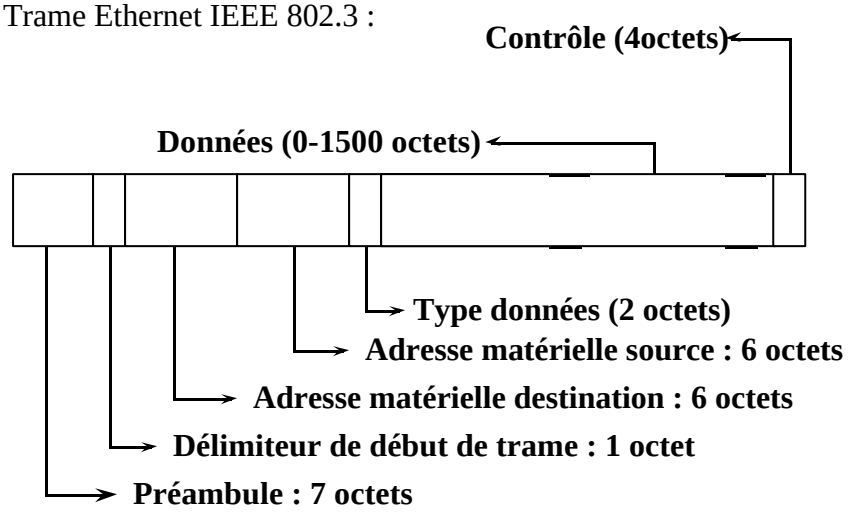
COUCHE LIAISON INTERNET

- On retrouve différents médiums :
  - Ethernet, le plus répandu. La méthode d 'accès (MAC) est de type CSMA/CD.
  - Liaison série (et couplage à un modem RTC). On utilise dans ce cas des protocoles spécifiques : SLIP, PPP...
  - Liaison radio : (*Wireless Internet).* Exemple : GSM 2G : on dispose d 'un canal de transmission de données à 9600 b/s (V.24). Le débit est faible !
  - Courant porteur.

- Pour chaque médium, on trouve définie au niveau liaison une trame : trame Ethernet, trame SLIP, trame PPP… Cette trame possède une taille **maximale.**

COUCHE LIAISON ETHERNET

- Trame Ethernet IEEE 802.3 :



SLIP (*Serial Line IP* RFC 1055)

- Protocole très simple.
- Permet d'émettre des datagrammes IP entre 2 ordinateurs reliés par une liaison série.
- Les datagrammes IP sont émis sur la ligne avec un octet de séparation END :
  - END code 0xC0.
  - si END dans les données : ESC ESC_END (0x0D, 0xDC).
  - si ESC dans les données : ESC ESC_ESC (0x0D, 0xDD).
- Pas de détection d'erreurs.
- Pas de négociation (adresse IP, taille des paquets, protocole transporté).

0xC0	Datagramme IP	0xC0

CSLIP (RFC 1144)

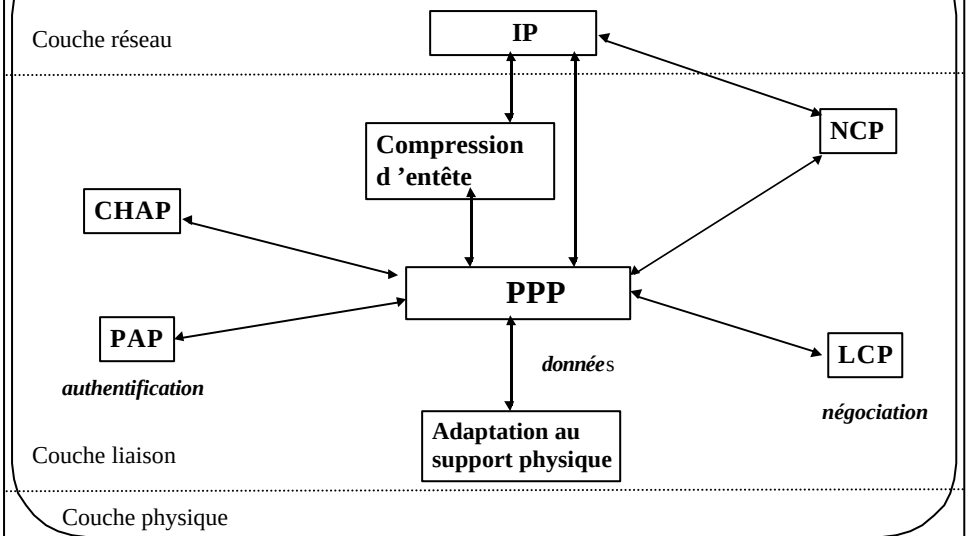
- Similaire à SLIP avec la compression des entêtes IP +TCP dite Van Jacobson (RFC 1144).
- L 'entête TCP/IP fait 40 octets sans compression !

PPP (*Point to Point Protocol* RFC 1661)

- Méthode standard pour transporter des datagrammes de protocoles différents sur des liaisons point à point (V.24, RNIS, X.25...).

- Les composants de PPP sont :
  - Une méthode pour encapsuler les datagrammes (paquet PPP).
  - Encapsulation du paquet PPP dans une trame HDLC simplifiée.
  - Un protocole de contrôle de liaison de données.
  - Établir, configurer, et tester la liaison de données.
  - Une famille de protocoles de contrôle du niveau réseau : établir, configurer les différents protocoles de la couche réseau (adresse IP dynamique).

PPP



PPP : ENCAPSULATION HDLC

7E	FF	03	paquet PPP	FCS	7E

- flag : $7E.
- address : $FF (adresse de diffusion).
- control : $03 = *Unumbered Info.*
- paquet PPP : données.
- FCS : *Frame Check Sequence (CRC).*
- $7E : flag de fin de trame HDLC.

PROTOCOLE IP (*Internet Protocol* RFC 791)

- C ’est le cœur du fonctionnement des protocoles Internet.
- Assure un mode sans connexion et un service non fiable (sans garantie) de délivrance des datagrammes IP.
- Les datagrammes IP peuvent être perdus, dupliqués, retardés, altérés ou désordonnés (analogie avec le service postal).
- Les principales fonctionnalités du protocole IP :
  - définition du datagramme IP.
  - service de transport non fiable des datagrammes IP.
  - routage dans le réseau Internet des datagrammes IP.

PROTOCOLES ARP ET RARP
(*Address Resolution Protocol* RFC 826 ET 903)

- Comme le protocole IP (avec ses adresses IP) peut être utilisé sur des supports de transmission différents (Ethernet, liaison série, radio…) ayant leur propre adresse physique, il faut établir des correspondances biunivoques entre les adresses IP et les adresses matérielles.
- Exemple : réseau Ethernet : adresse Ethernet sur 48 bits. Pour envoyer un datagramme IP sur réseau Ethernet, on a besoin des adresses IP et Ethernet.

PROTOCOLES ARP ET RARP

- ARP (*Address Resolution Protocol*) fournit une correspondance dynamique entre une adresse IP et une adresse matérielle.
- RARP (*Reverse Address Resolution Protocol*) réalise le contraire.
- Lors de la demande de transmission du premier datagramme IP :
  - envoi d'une requête ARP avec l'adresse IP du destinataire sur le réseau Internet pour récupérer son adresse matérielle.
  - la requête n'est reconnue que par le bon destinataire qui renverra alors son adresse matérielle dans un datagramme ARP en réponse.

PROTOCOLES ARP ET RARP

- Datagramme ARP :

0	8	16	24	31
Type de matériel		Type de protocole		
LGR-MAT	LGR-PROT	Opération		
Adresse matériel émetteur (octets 0-3)				
Adresse Mat émetteur (octets 4,5)		Adresse IP émetteur (octets 0,1)		
Adresse IP émetteur (octets 4,5)		Adresse Mat cible (octets 0,1)		
Adresse Matériel cible (octets 2,5)				
Adresse IP cible (octets 0-3)				

PROTOCOLE IP (*Internet Protocol* RFC 791)

- Entête de 20 octets minimum.
- Données (< 65536 octets).

0	4	8	16	19	24	31
VERS	HLEN	Type de service	Longueur totale			
Identification			Flags	Offset fragment		
Durée de vie		Protocole	Somme de contrôle Header			
Adresse IP Source						
Adresse IP Destination						
Options IP (éventuellement)					Padding	
Données						
						. . .

PROTOCOLE IP

- version : 4 bits : version 4 IPv4 (version 6 IPng en cours…)
- longueur entête : 4 bits (combien de mots de 32 bits)
- ToS : 8 bits (priorité + préférences sur la qualité)
- longueur totale : 2 octets (taille en octets donc taille < 65536 octets)
- identification, drapeaux et déplacement de fragment : 4 octets
- durée de vie : 1 octet (en nombre de routeurs traversés)
- protocole : 1 octet (protocole de plus haut niveau utilisé)
  - **6 : TCP**
  - **17 : UDP**
  - **1 : ICMP**
- header checksum : 2 octets
- adresses IP source/destination : 4 octets
- options (+ bourrage à 4 octets)

FRAGMENTATION DES DATAGRAMMES IP

- Les datagrammes sont encapsulés dans des trames de niveau 2 (liaison) qui ont leur propre taille maximale.
- Comme il y a interconnexion de réseaux hétérogènes par les protocoles Internet, il impossible de connaître la taille maximale d'une trame.
- Il y a donc nécessité de fragmenter les datagrammes IP (<65536 octets) en cours de transmission dans le réseau Internet.

FRAGMENTATION DES DATAGRAMMES IP

- Pour chaque réseau traversé par un datagramme IP, il existe ainsi un MTU (*Maximum Transfert Unit*).
- Si taille_data_IP < MTU,
  - encapsulation immédiate dans une trame correspondant au type de réseau traversé.
  - sinon, fragmentation (en multiple de 8 octets).
- S 'il y a fragmentation, c'est le destinataire final qui réassemble (même si l 'on passe par des réseaux à MTU plus grand). On code dans le champ *offset fragment*, la position du fragment par rapport au datagramme IP initial.

ROUTAGE IP

- Le routage est l 'opération d 'aiguiller chaque datagramme IP vers son destinataire.
- Cette opération est réalisée par un routeur IP.
- Si l ' échange se fait entre 2 machines connectées sur le même réseau, il suffit d'encapsuler (voire de fragmenter) le datagramme IP dans la trame de niveau liaison.
- Quand on passe par plusieurs réseaux, il faut savoir comment envoyer le datagramme vers sa destination finale. On utilise des tables de routage dans chaque routeur IP qui possède aussi une route par défaut. Des algorithmes de routage sont alors mis en œuvre.

ICMP (*Internet Control Message Protocol* RFC 792)

- Le protocole ICMP permet d'envoyer des messages de contrôle ou d'erreur vers d'autres machines ou routeurs.
- ICMP rapporte les messages d'erreur à l'émetteur initial.
- Beaucoup d'erreurs sont causées par l'émetteur, mais d'autres sont dues à des problèmes d'interconnexion rencontrés sur Internet :
  - machine destination déconnectée.
  - durée de vie du datagramme expirée (TTL=0).
  - congestion de routeurs intermédiaires.

ICMP

- Si une passerelle détecte un problème sur un datagramme IP, elle le détruit et émet un message ICMP pour informer l'émetteur initial.
- Les messages ICMP sont véhiculés à l'intérieur de datagrammes IP et sont routés comme n'importe quel datagramme IP sur Internet.
- Une erreur engendrée par un message ICMP ne peut donner naissance à un autre message ICMP (évite l'effet cumulatif).

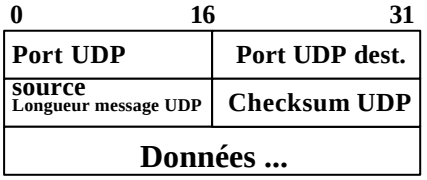
PROTOCOLES TCP ET UDP

- Les protocoles de niveau transport TCP et UDP utilisent IP comme service réseau.
- TCP procure un service de transport de données en mode connecté fiable (alors que IP ne l'est pas).
- UDP procure un service de transport de données en mode non connecté ou datagramme non fiable (comme IP).

PROTOCOLE UDP
(*User Datagram Transport* RFC 768)

- UDP achemine les données de l 'utilisateur en utilisant le service IP en mode datagramme non fiable.
- Pas d'accusé de réception (pas de vérification possible de la bonne réception).
- Pas de réordonnancement des messages.
- Pas de contrôle de flux.
- C'est à l'application de gérer les pertes, duplications, retards, déséquencement…
- UDP permet cependant de distinguer plusieurs applications destinatrices des données reçues sur la même machine par l'intermédiaire d 'un mot de 16 bits appelé numéro de port.

PROTOCOLE UDP



• Les ports source et destination contiennent les numéros de port utilisés par UDP pour démultiplexer les données destinées aux applications en attente de les recevoir. Le port source est facultatif (égal à zéro si non utilisé).

• On définit ainsi des applications clientes et des applications serveurs (programmation client/serveur).

PROTOCOLE UDP

- Pour accéder à un service Internet on est ainsi obligé de préciser :
  - L ’adresse matérielle de la machine appelée.
  - L ’adresse IP de la machine appelée.
  - Le numéro de port du service contacté.
- Il en va de même pour la machine appelante.
- Ceci est valable pour TCP.

PROTOCOLE UDP

- UDP multiplexe et démultiplexe les datagrammes IP en fonction du numéros de port.
- Lorsque UDP reçoit un datagramme, il vérifie que celui-ci est un des ports actuellement actifs (associé à une application) et le délivre à l'application responsable (mise en queue).
- Si ce n'est pas le cas, il émet un message ICMP *port unreachable*, et détruit le datagramme IP.
- Ceci est valable pour TCP.

PROTOCOLE UDP

- Certains ports sont réservés (*well-kown port assignements*) :

No port	Mot-clé	Description
7	ECHO	Echo
11	USERS	Active Users
13	DAYTIME	Daytime
37	TIME	Time
42	NAMESERVER	Host Name Server
53	DOMAIN	Domain Name Server
67	BOOTPS	Boot protocol server
68	BOOTPC	Boot protocol client
69	TFTP	Trivial File transfert protocol
123	NTP	Network Time Protocol
161	SNMP	Simple Network Management prot.

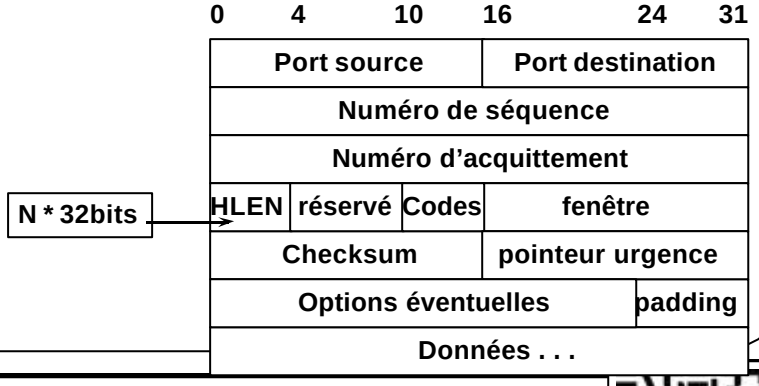
- D'autres numéros de port (non réservés) peuvent être assignés dynamiquement aux applications (>1024).

PROTOCOLE TCP
(*Transport Control Protocol* RFC 793)

- TCP procure un service en mode connecté et fiable : garantie de non perte de données ainsi que de l'ordre.
- Il permet de transférer un flux d'octets non structuré.
- Établissement préalable d'une connexion (mode connecté).
- L'unité d'information transmise est le segment. Le segment résulte de la fragmentation ou de la concaténation de données transmises par l'application.

PROTOCOLE TCP

- Segment : unité de transfert du protocole TCP :
  - établir une connexion TCP. *Handshake* par émission de 3 segments TCP.
  - transférer les données.
  - émettre des acquittements.
  - fermer les connexion TCP.



PROTOCOLE TCP

- Certains ports sont réservés (*well-kown port assignements*) :

No port	Mot-clé	Description
21	FTP	File Transfer [Control]
23	TELNET	Telnet
25	SMTP	Simple Mail Transfer
37	TIME	Time
42	NAMESERVER	Host Name Server
43	NICNAME	Who Is
53	DOMAIN	Domain Name Server
79	FINGER	Finger
80	HTTP	WWW
110	POP3	Post Office Protocol - Version 3
111	SUNRPC	SUN Remote Procedure Call

- D'autres numéros de port (non réservés) peuvent être assignés dynamiquement aux applications (>1024).

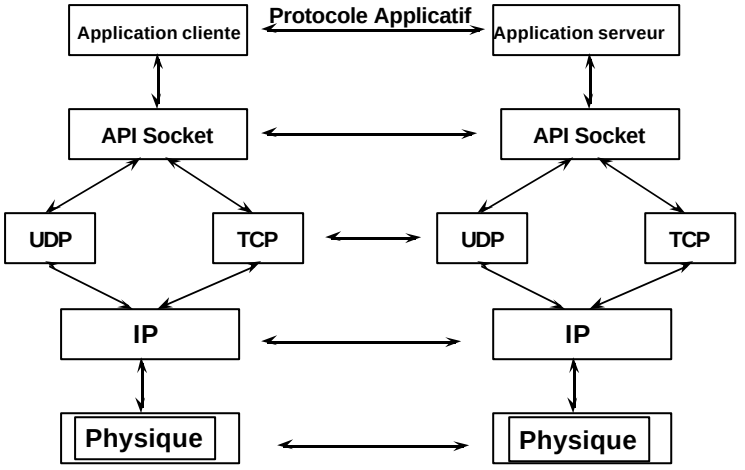
L 'API sockets

- Les sockets : interface client/serveur (API) utilisée à l'origine dans le monde UNIX et TCP/IP. Existe aujourd'hui du micro (*winsock*) au *Mainframe.*
- L' API sockets est le standard de fait pour la programmation réseau Internet.
- Il existe d 'autres APIs de programmation réseau : *Streams*, TLI, RPC, XDR, propriétaires…
- Les applications client/serveur ne voient les couches de communication qu'à travers l'API sockets (abstraction) .

L ’API sockets

- Dans l ’environnement UNIX, les sockets sont traitées de la même manière que les fichiers :
  - on a donc des appels systèmes d ’ouverture (qui permet d ’avoir un descripteur de référence),
  - de lecture,
  - d ’écriture,
  - de contrôle et de fermeture.

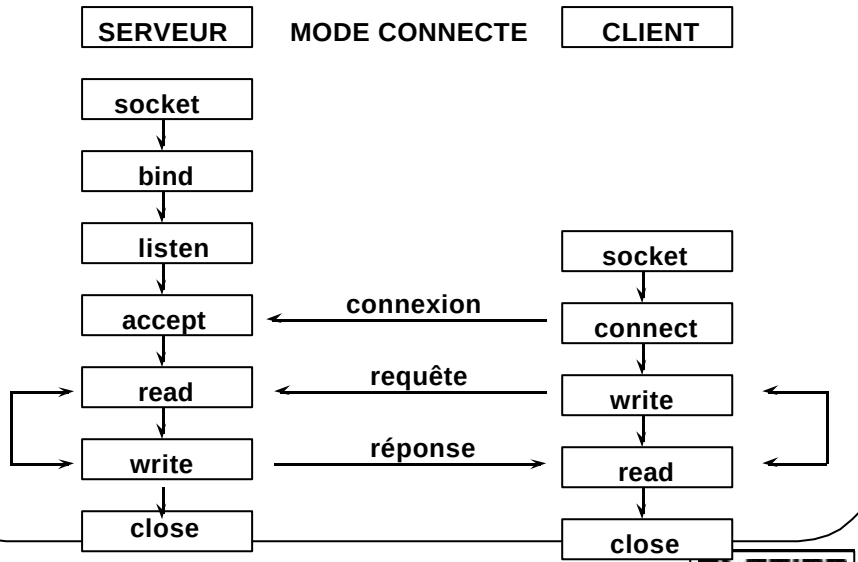
L ’API sockets



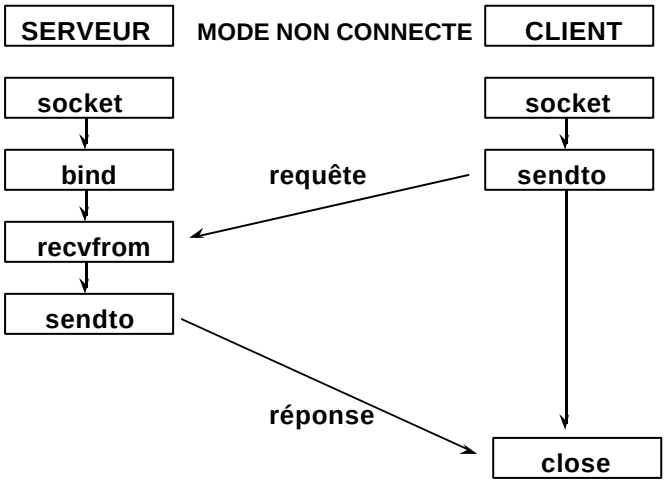
L 'API sockets

- Les sockets permettent d'établir un lien de communication en mode connecté ou non connecté sur un réseau Internet.
- Les sockets structurent une application :
  - soit en mode client.
  - soit en mode serveur.
- Les sockets permettent d'échanger des données entre ces applications.

L 'API sockets



L ’API sockets



L ’API sockets

- L ’API sockets permet d ’échanger des octets entre une application cliente et une application serveur. En mode connecté, on établit un flux octets (*stream*) non structuré.
- Les données échangées « passent en clair » sur le réseau. Elles ne sont pas cryptées. L ’aspect confidentialité des données échangées est apparue très tard dans le monde des télécoms.
- Pour échanger des données cryptées (mode sécurisé), on a développé une nouvelle API : SSL (*Secure Socket Layer*).
- Toutes les applications réseau développées utilisent l ’API sockets (volontairement ou non)...

PARTIE 3 :
LES APPLICATIONS INTERNET

TELNET ET RLOGIN (RFC 854)

- Ces commandes permettent à un utilisateur de se connecter à un ordinateur distant. Les deux utilisent TCP.
- telnet est aussi un client pour se connecter à tout serveur en mode connecté (TCP).
- rlogin ne fonctionne qu'entre 2 machines UNIX.
- rlogin fait partie de la famille des commandes UNIX r… (rsh, rcp…).
- **On a à chaque fois un client et un serveur pour les applications Internet**.

NFS (*Network File System* RFC 3010)

- NFS permet de rendre transparente l'utilisation de fichiers de systèmes de fichiers répartis sur différentes machines.
- NFS utilise UDP mais les nouvelles versions utilisent TCP.
- NFS est utile quand le système ne dispose pas de système de fichiers local (station *diskless*, système embarqué).

FTP (*File Transfer Protocol* RFC 959)

- FTP permet le transfert de fichiers d'une machine à une autre.
- FTP nécessite la connexion de l'utilisateur avec un nom et un mot de passe.
- Si l'utilisateur n'est pas reconnu, pas de connexion.
- Il existe des serveur FTP anonymes : nom d 'utilisateur *anonymous* avec son email comme mot de passe.

TFTP (*Trivial FTP* RFC 1350)

- Transfert de fichiers d'une machine à une autre.
- TFTP est plus sommaire (UDP) que FTP (TCP).
- TFTP permet de télécharger le noyau d 'un OS d 'une machine *diskless* par exemple.

SMTP (*Simple Mail Transfer Protocol* RFC 821)

- SMTP permet d'échanger des courriers électroniques entre un expéditeur et un ou plusieurs destinataires.
- SMTP utilise TCP.
- L'adresse est de la forme : nom@domaine.
- SMTP effectue une remise différée du courrier (en cas de non disponibilité temporaire du destinataire).

World Wide Web : HTTP
(*HyperText Transfer Protocol* RFC 1945)

- HTTP est le protocole de communication et d 'échange de documents multimédia du « web ».
- HTTP permet d'échanger des documents hypertextes contenant des données sous la forme de texte, d'images fixes ou animées et de sons.
- Un serveur web est écrit en utilisant l 'API sockets pour lequel on structure le flux d 'octets non structuré au départ sous forme de lignes de commandes ASCII : c 'est le protocole HTTP !
- On utilise toujours le concept d 'application client/serveur : navigateur (Netscape…)/serveur web (Apache, boa, thttpd…).

SNMP
(*Simple Network Management Protocol* RFC 1157)

- SNMP est le standard de fait dans l 'administration de réseau. Il a supplanté le standard international de l 'IUT-T dans ce domaine.
- SNMP permet aussi de contrôler à distance des matériels.
- Il est bâti autour du concept client/serveur : agent/manager SNMP.
- SNMP utilise UDP et les transferts de données entre agent et manager sont non sécurisés !

PLUS D ’INFORMATIONS

- Les RFC (*Request For Comment*), normes des protocoles Internet (gratuit) :
  - http://www.rfc-editor.org/
- Quelques RFC traduites en français :
  - http://www.guill.net/reseaux/Rfc.html

PARTIE 4 :
LES PROTOCOLES INTERNET POUR LA CONNECTIVITE IP

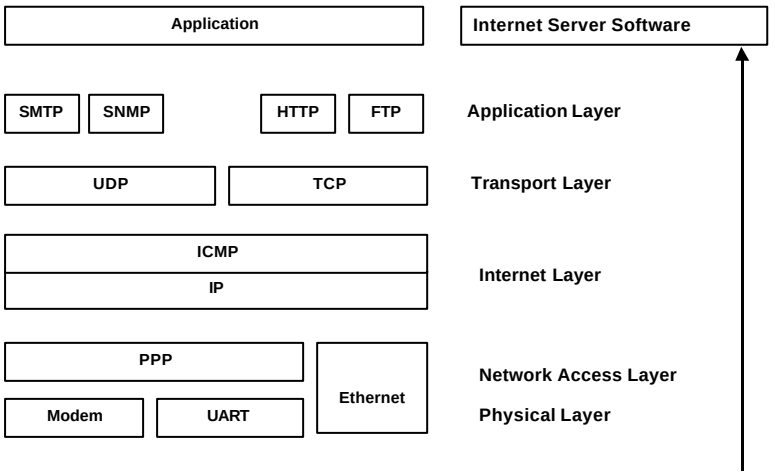
BILAN

- Les protocoles Internet sont indépendants des supports de transmission utilisés.
- Les supports de transmission préférentiels sont :
  - Ethernet.
  - Liaison série.
- Pour chaque support de transmission est définie une trame au niveau liaison :
  - Ethernet : trame Ethernet.
  - Liaison série : trame SLIP, PPP...

BILAN

- Il convient d ’implanter le protocole de niveau liaison sous forme matérielle ou logicielle :
  - Ethernet : CSMA/CD (par matériel).
  - Liaison série : SLIP, PPP… (par logiciel). Cette configuration est choisie pour un accès à Internet par le RTC.
- Il convient ensuite d ’implanter les protocoles IP en fonction des besoins du système électronique pour assurer la connectivité IP...

BILAN



CONNECTIVITE IP MINIMALE

- En fonction des particularités du système électronique, on choisira :
  - IP en mode « raw » : simple, pour un petit système . Pas de multiplexage (pas de numéro de port), pas d 'interactivité, efficace. Développement du protocole simple, bas niveau, sans état.
  - UDP/IP : simple, pour un petit à gros système. Multiplexage possible (par numéro de port), pas d 'interactivité, efficace. Développement du protocole simple, bas niveau, sans état.
  - TCP/IP : compliqué, pour un moyen à gros système. Multiplexage possible (par numéro de port), interactivité, peu efficace. Développement du protocole complexe, bas niveau, avec états.
- Dans tous les cas, les données échangées sont non structurées (octets).

CONNECTIVITE IP MINIMALE

- En marge de ces possibilités de **connectivité IP bas niveau**, il est fortement conseillé d ’embarquer les protocoles de contrôle et de supervision suivants :
  - ICMP : permet de voir si le système électronique est actif par un « *ping* ». Développement du protocole simple, bas niveau, sans état.
  - ARP (RARP) : pour que le système électronique puisse récupérer une adresse matérielle. Développement du protocole simple, bas niveau, sans état.

- IP ou UDP/IP ou TCP/IP couplés avec les « outils » ICMP/ARP suffisent pour mettre en place une connectivité IP dans un équipement. UDP/IP est plus performant que TCP/IP si l ’on a des contraintes Temps Réel à respecter.

CONNECTIVITE IP MINIMALE

- **IP ou UDP/IP ou TCP/IP couplés avec les « outils » ICMP/ARP suffisent pour mettre en place une connectivité IP dans un équipement.**

- **UDP/IP est plus performant que TCP/IP si l ’on a des contraintes Temps Réel à respecter.**

CONNECTIVITE IP : PREMIERE AMELIORATION

- Pour faciliter le développement des applications réseau, il est préférable d 'avoir disponible l 'API sockets.
- L 'API sockets assure une portabilité au niveau source des applications et une réduction du temps de développement.
- On travaille toujours sur des octets ou un flux d 'octets non structuré.
- Il convient de développer des applications UDP ou TCP s 'exécutant sur le système traitant ces octets.
- L 'utilisation d 'un OS ou un RTOS embarqué sur le système est préférable.

CONNECTIVITE IP : AUTRES AMELIORATIONS

- On préférera au dessus de UDP ou TCP utiliser des protocoles qui vont structurer les données si les performances du système le permettent pour assurer une **connectivité IP haut niveau**.
- Le flux d 'octets non structuré est généralement structuré sous forme de chaînes de caractères ASCII. Cela va permettre d 'accélérer le développement et la mise au point de l 'application serveur à embarquer dans le système.

CONNECTIVITE IP : AUTRES AMELIORATIONS

- Les protocoles couramment utilisés pour cela sont :
  - HTTP : mise en place d 'une application serveur web embarquée sur le système.
  - SMTP : mise en place d 'une application serveur de emails embarquée sur le système pour l 'envoi de courriers électroniques
  - SNMP : mise en place d 'une application serveur/agent SNMP embarquée sur le système.

CONNECTIVITE IP : SERVEUR WEB

- L 'utilisation d 'un serveur web embarqué est très employée pour le télécontrôle du système électronique.
- Le contrôle du système se fait avec n 'importe quel navigateur web.
- L 'interactivité est possible en utilisant l 'interface CGI (*Common Gateway Interface*) qui permet de faire exécuter une fonction/application par le système sur une requête du navigateur web.
- L 'action est à l 'initiative de l 'opérateur.

CONNECTIVITE IP : SMTP

- L ’utilisation d ’un serveur SMTP embarqué est aussi employée pour le télécontrôle du système électronique.
- Le système peut envoyer un mail pour alerter un opérateur (qui peut être relayé vers son portable GSM).
- L ’action est à l ’initiative du système.

CONNECTIVITE IP : AGENT SNMP

- L ’utilisation d ’un agent SNMP embarqué est moins courante pour le télécontrôle du système électronique.
- Le contrôle du système se fait avec un manager SNMP (Openview de HP…). Le manager SNMP est moins standard qu ’un navigateur web pour le grand public...
- L ’action est à l ’initiative du système (Trap SNMP) ou de l ’opérateur (Get, Set SNMP).

CONNECTIVITE IP : AUTRES AMELIORATIONS

- Les autres protocoles/services de l ’Internet peuvent être vus comme des services de confort :
  - NFS : montage de partitions NFS sur la machine de développement pour faciliter la mise au point.
  - telnet : connexion à distance sur le système pour mise au point *in situ*.
  - ftp : serveur ftp embarqué pour télécharger des mises à jour, configurations dans le système.
  - ...

CONNECTIVITE IP : LES QUESTIONS AVANT LE (BON) CHOIX

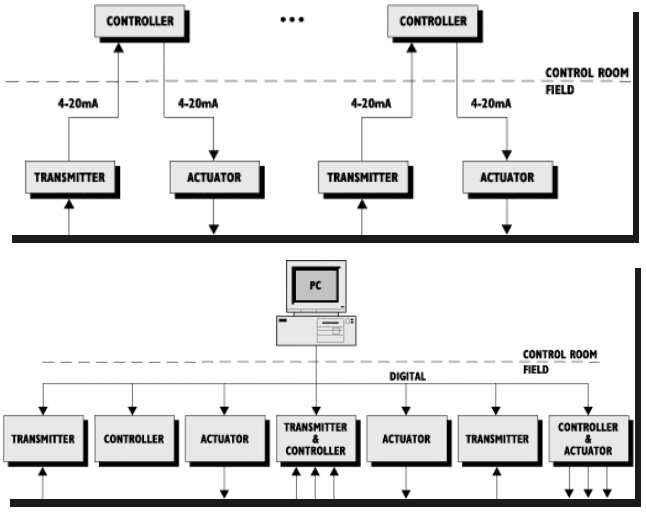
- Les choix à opérer vont dépendre de différents critères :
  - système électronique simple ou performant ?
  - accès réseau filaire, radio (mobilité) ?
  - profil métier : concepteur de cartes électroniques, intégrateur de système, utilisateur final ?
  - solution clé en main, développement *from scratch* ?
  - coûts ?
  - solution propriétaire, logiciels/matériels libres ?
  - délais (TTM) ?
  - compétence en interne ?

PARTIE 5 :
LES BUS DE TERRAIN ET LA CONNECTIVITE IP

RESEAU DE TERRAIN ET CONNECTIVITE IP

- Un BUS / RESEAU DE TERRAIN est :
  - le terme générique d'un nouveau réseau de communication *numérique* dédié à l 'automatisme et au contrôle de process.
  - un réseau bidirectionnel, multibranche (*multidrop*), série reliant

  différents types d'équipements d'automatisme :
    - E/S déportées.
    - Capteur / Actionneur.
    - Automate programmable.
    - Calculateur.
- Un réseau de terrain peut être vu comme un réseau de communication entre équipements déportés dans un zone géographique limitée. La connectivité IP n 'a pas été prise en compte au départ !



source : SMAR

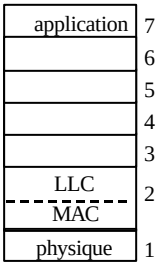
Passage de la boucle de courant analogique 4-20 mA au bus de terrain

RESEAU DE TERRAIN ET CONNECTIVITE IP

- Un bus de terrain est basé sur la restriction du modèle OSI à 3 couches :
  - Couche physique.
  - Couche liaison de données.
  - Couche application.
- Cette modélisation est respectée par les standards de fait et internationaux.

RESEAU DE TERRAIN ET CONNECTIVITE IP

- Couches 3 à 6 vides :
  - Pas de besoin d'interconnexion avec un autre réseau (pas de connectivité IP possible à priori !).
  - Gain en performance car on a besoin d 'être dans la majorité des cas déterministe.



Bus de terrain et modèle OSI

RESEAU DE TERRAIN ET CONNECTIVITE IP

- Pour mémoire, les réseaux de terrain les plus utilisés sont :
  - CAN, SDS, Devicenet.
  - Profibus.
  - WorldFIP.
  - Interbus.
  - Lonworks.
  - …

RESEAU DE TERRAIN ET CONNECTIVITE IP

- Deux solutions techniques sont envisagées pour assurer la connectivité IP :
  - Solutions qui encapsulent les trames du bus de terrain dans une trame Ethernet (sur Ethernet) ou paquet TCP/UDP.
  - Solutions qui utilisent des machines passerelles entre les 2 réseaux.

RESEAU DE TERRAIN ET CONNECTIVITE IP

- Un exemple : PROFINET de PROFIBUS.

- PROFINET est :
  - ☞ Basé sur l 'utilisation de technologies standards établies et répandues (TCP/IP…).
  - ☞ Basé sur une approche objet : objet COM/DCOM de Microsoft, manipulation d 'objets à l 'aide de Microsoft OLE et ActiveX.
  - ☞ Vendeur indépendant.
  - ☞ Intégrable à PROFIBUS sans modification.
  - ☞ Ouvert pour l 'intégration d 'autres systèmes.

RESEAU DE TERRAIN ET CONNECTIVITE IP

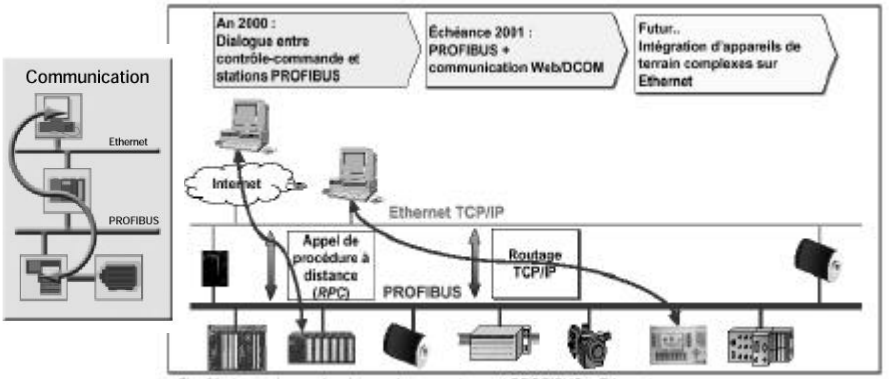
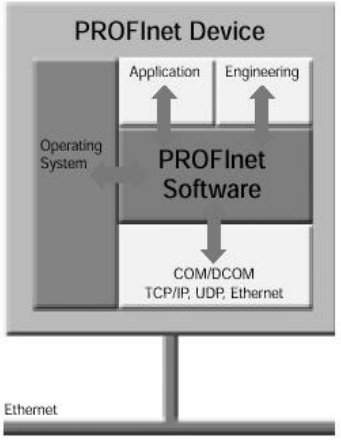


Fig. 23 : Les trois grandes étapes du rapprochement PROFIBUS – Ethernet

RESEAU DE TERRAIN ET CONNECTIVITE IP



Offre PROFINET

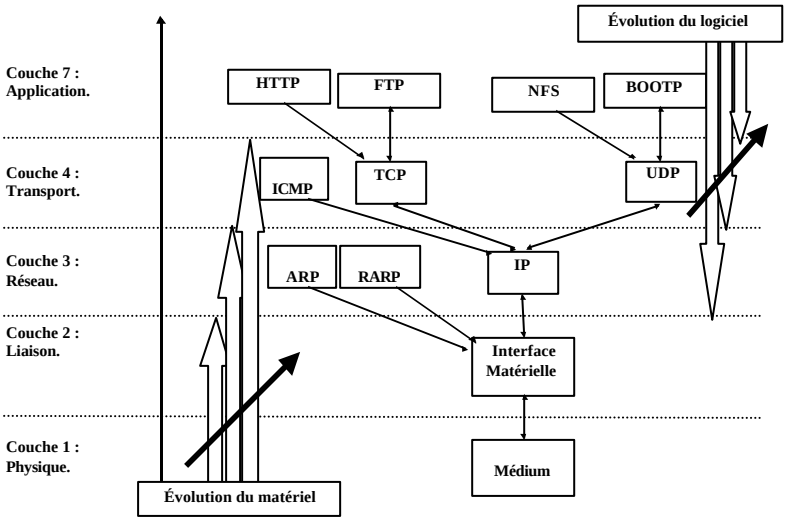
PARTIE 6 :
CONNECTIVITE IP : SOLUTION MATERIELLES ET LOGICIELLES

INTRODUCTION

- Avec une intégration sur silicium de plus en plus importante, les solutions logicielles d'hier deviennent des solutions matérielles aujourd'hui avec le gain en rapidité d'exécution et de décharge pour le processeur qui en découle.

- Il semble que l'évolution des solutions matérielles de connectivité IP se fasse au détriment des solutions logicielles pour le grand bien du concepteur !

INTRODUCTION



INTRODUCTION

- Au niveau médium, on utilise pour la mise en place de la connectivité IP principalement :
  - liaison Ethernet IEEE 802.3 10/100BaseT avec implémentation matérielle de la sous couche MAC CSMA/CD.
  - liaison série RS.232/V.24 avec encapsulage des datagrammes IP dans des paquets PPP ou plus simplement en utilisant le protocole SLIP.
  - liaison GSM : utilisation d ’un module électronique GSM qui permet d ’envoyer des *emails*.
  - autres liaisons radioélectriques : développement important du *Wireless Internet*.
  - courant porteur.

INTRODUCTION

- La solution idéale reste la liaison Ethernet IEEE 802.3 pour des questions de coûts, performances et d ’infrastructure.
- Le système avec sa connectivité IP est directement connecté au réseau local de l ’installation. L ’accès à l ’Internet est réalisé par un routeur IP. Le routeur peut se résumer à un modem RTC intégré à un PC pour se connecter à un fournisseur d ’accès.
- L ’autre solution couramment utilisée est d ’utiliser une liaison RS.232/V.24 avec PPP et modem RTC .

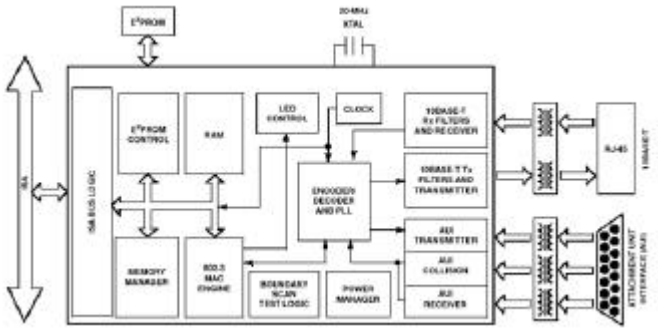
POINT 1 : SOLUTIONS MATERIELLES POUR LA CONNECTIVITE IP

SOLUTIONS MATERIELLES

- Les solutions utilisant une liaison Ethernet IEEE 802.3 10/100BaseT sont présentées ici.
- Il s 'agit de circuits électroniques d 'accès qu 'il faut intégrer dans son design.
- On utilisera ensuite les drivers (suivant l 'OS) pour servir de base à l 'implémentation des protocoles IP.

SOLUTIONS MATERIELLES : CS8900A

- CIRRUS LOGIC propose un circuit d'interface IEEE 802.3 : le CS8900A. C 'est le circuit le plus utilisé !



SOLUTIONS MATERIELLES : CS8900A

Adresse web	www.cirrus.com
Solution	Matérielle Circuit CS8900A TQFP 100 broches
Interfaces Ethernet	802.3 10BaseT, 10Base2, 10Base5 full duplex
Implémentation niveau MAC	Oui (CSMA/CD)
Implémentation niveau IP	Non
Implémentation niveaux TCP, UDP	Non
Fonctionnalités	Interface ISA Modes I/O et MEM DMA
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Excellente
Facilité de programmation	Oui
Drivers fournis	Oui Microsoft Windows Linux PSOS, VxWorks SCO
Prix des drivers	Gratuit
Prix du composant	61,10 F HT (par 10) revendeur : MEMEC
Support après vente	Oui Hotline, SOS par mail Cirrus propose gratuitement de qualifier tout design à base du circuit CS8900A

SOLUTIONS MATERIELLES : RTL8019AS

- REALTEK propose un circuit d'interface 802.3 : le RTL8019AS.

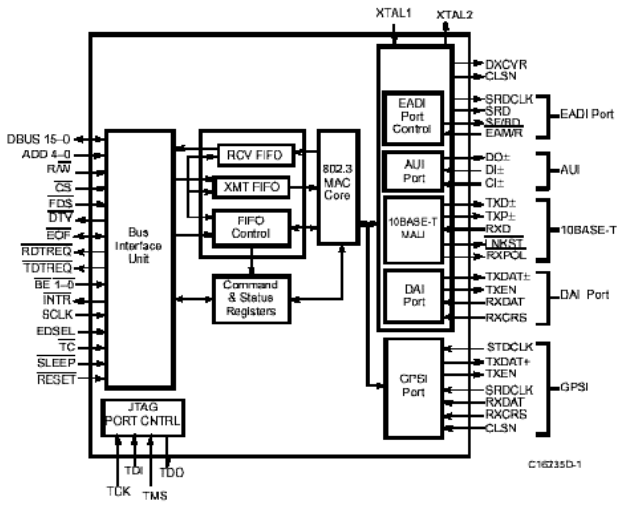
Adresse web	www.realtek.com.tw
Solution	Matérielle Circuit RTL8019AS PQFP 100 broches
Interfaces Ethernet	802.3 10BaseT, 10Base2, 10Base5 full duplex
Implémentation niveau MAC	Oui
Implémentation niveau IP	Non
Implémentation niveaux TCP, UDP	Non
Fonctionnalités	Interface ISA Modes I/O
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Bonne
Facilité de programmation	Oui
Drivers fournis	Oui Microsoft Windows Linux SCO
Prix des drivers	Gratuit
Prix du composant	?
Support après vente	Non

SOLUTIONS MATERIELLES : LANCE

- AMD, leader dans les circuits d'interface réseau propose lui aussi des circuits d'interface IEEE 802.3 basé sur son célèbre circuit LANCE 7990 : les circuits de la série 79C9xx.

- Le circuit intéressant dans cette série est le 79C940 ou circuit MACE (Media Access Controller for Ethernet). Il a été spécialement conçu pour les applications embarquées 16 bits.

SOLUTIONS MATERIELLES : LANCE

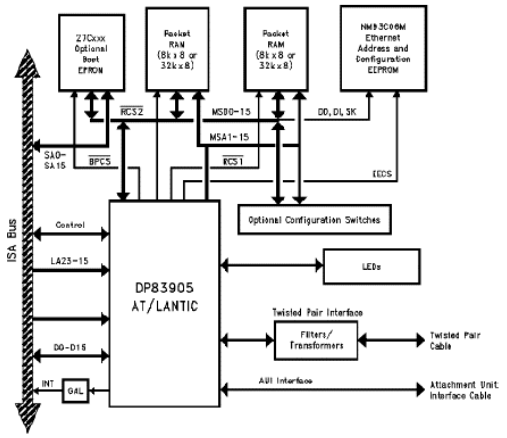


SOLUTIONS MATERIELLES : LANCE

Adresse web	www.amd.com
Solution	Matérielle Circuit Am79C940 TQFP 80 broches et PLCC 84 broches
Interfaces Ethernet	802.3 10BaseT, 10Base2, 10Base5 full duplex
Implémentation niveau MAC	Oui
Implémentation niveau IP	Non
Implémentation niveaux TCP, UDP	Non
Fonctionnalités	Modes I/O DMA FIFOs en Rx et Tx Mode sleep
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Très bonne
Facilité de programmation	Oui
Drivers fournis	Oui Microsoft Linux VxWorks SCO
Prix des drivers	Gratuit
Prix du composant	? revendeurs : Arrow, Avnet, Tekelec
Support après vente	Oui par mail

SOLUTIONS MATERIELLES : DP83905

- National Semiconductor a aussi un circuit d'interface IEEE 802.3 : le circuit DP83905.



SOLUTIONS MATERIELLES : DP83905

Adresse web	www.national.com
Solution	Matérielle Circuit DP83905 TQFP 160 broches
Interfaces Ethernet	802.3 10BaseT, 10Base2, 10Base5 full duplex
Implémentation niveau MAC	Oui
Implémentation niveau IP	Non
Implémentation niveaux TCP, UDP	Non
Fonctionnalités	Modes I/O RAM en plus en externe
Schémas de principe d'utilisation	Non
Qualité de la documentation	Moyenne
Facilité de programmation	Moyenne
Drivers fournis	Non Compatible NE2000
Prix des drivers	-
Prix du composant	9,5 $ (par 1000) revendeurs : Arrow, Avnet
Support après vente	Non

POINT 2 :
SOLUTIONS LOGICIELLES POUR LA CONNECTIVITE IP

SOLUTIONS LOGICIELLES

- Contrairement à ce que l'on pourrait croire, il existe peu de briques logicielles implémentant les protocoles et services Internet (IP, UDP, TCP…) disponibles au niveau source, gratuits ou sous licence GPL.
- En fait, l'implémentation de ces protocoles est toujours liée à un système d'exploitation lourd qui est à priori non facilement embarquable.
- On trouve ainsi au niveau source (en langage C) les protocoles IP que l'on nommera TCP/IP globalement pour les OS (*Operating System*) UNIX BSD, FreeBSD, NetBSD et Linux sous licence GPL.

SOLUTIONS LOGICIELLES

- Avec l'apparition de projets Linux embarqué, on peut avoir Linux sur une plateforme matérielle et une connectivité naturelle à Internet…
- Il ne semble pas envisageable de modifier les sources TCP/IP pour s'affranchir de l'OS sous-jacent car les deux sont intiment liés (on y fait appel à des appels systèmes propres à l'OS). Il faut dans cette optique prendre les protocoles IP et l ’OS…
- Il existe donc peu d'implémentations de protocoles IP non liées à un OS (généralement de type UNIX).

SOLUTIONS LOGICIELLES

- On peut citer 2 piles TCP/IP en libre possédant beaucoup de restrictions d'usage :
  - Projet WATTCP (www.wattcp.com) : pile TCP/IP écrite en langage C disponible gratuitement au niveau source tournant sous DOS avec le driver PKTDRVR pour un accès Internet par PPP (Cf annexe).
  - KA9QNOS (http://people.qualcomm.com/karn/code/ka9qnos/) (d'un radioamateur à la base du packet radio) : pile TCP/IP écrite en langage C disponible gratuitement au niveau source tournant sous DOS pour un accès Internet par PPP. Le contrôleur de liaison série doit être de la famille Zilog Z8530.

SOLUTIONS LOGICIELLES : TRECK

- La société TRECK propose une pile TCP/IP et ses sources. Ses produits sont optimisés et produisent des codes rapides, petits, réentrants et ROMables. A travers ses produits, on a accès à TCP/IP, UDP, PPP, ARP, ICMP, DHCP, SMTP et les services ftp, telnet tftp et serveur Web.

- Les codes ont été testés avec les processeurs PowerPC, 68K, ARM, 320C32 et x86.

SOLUTIONS LOGICIELLES : TRECK

- Il n'y a pas obligation d'utiliser un noyau temps réel *(Real Time Operating System RTOS*) mais l'intégration des produits Treck avec un RTOS est toujours possible (par exemple µC/OS II). Les drivers pour piloter les contrôleurs Ethernet sont aussi disponibles et notamment pour les 3 solutions matérielles retenues suivantes : CS8900, Am79C940 et DP83905. Treck peut aussi développer le driver pour d'autres contrôleurs Ethernet.

Les produits Treck intéressants sont :

- ♦ Treck Real-Time TCP/IP.
- ♦ Treck RomPager Embedded Web Server (de la société Allegro).
- ♦ Treck RomPager Light Embedded Web Server.

SOLUTIONS LOGICIELLES : TRECK

- Produit Treck Real-Time TCP/IP :

Adresse web	www.treck.com
Solution	Logicielle
Interfaces Ethernet	Oui par driver d'interface aux principaux contrôleurs (ex : CS8900, Am79C940 et DP83905)
Besoin d'un RTOS	Non, mise en œuvre d'un RTOS possible UC/OS II supporté
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui
Autres protocoles	ARP, ICMP
Interface de programmation	Oui Sockets BSD
Processeurs testés	PowerPC, 68K, ARM, x86, 320C32
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne, programmation sockets
Drivers fournis	oui
Prix	10000 $ si < 2000 unités 20000 $ si pas de royalty

SOLUTIONS LOGICIELLES : TRECK

- Produit Treck RomPager Embedded Web Server :

Adresse web	www.treck.com
Solution	Logicielle Produit de la société Allegro
Interfaces	Oui par Treck Real-Time TCP/IP
Besoin d'un RTOS	Non, mise en œuvre d'un RTOS possible
Implémentation niveau HTTP	Oui version 1.1 Support de HTML version 2.0 à 4.0
Qualité de la documentation	Très bonne
Prix	5000 $ si < 2000 unités 10000 $ si pas de royalty

SOLUTIONS LOGICIELLES : RTIP

- La société EBSnet propose des produits logiciels assurant une connectivité Internet et notamment une pile TCP/IP et ses sources : produit RTIP. A travers RTIP, on a accès à TCP/IP, UDP, PPP, ARP, ICMP, RARP, BOOTP. Un serveur Web embarqué est aussi proposé.

- Les sources ont été testés avec les processeurs PowerPC, 68K, ARM, x86…

- Les drivers pour piloter les contrôleurs Ethernet sont aussi disponibles et notamment pour les 2 solutions matérielles suivantes : CS8900, Am79C96x.

SOLUTIONS LOGICIELLES : RTIP

Adresse web	www.etcbin.com
Solution	Logicielle
Interfaces Ethernet	Oui par driver d'interface aux principaux contrôleurs (ex : CS8900, Am79C96x)
Besoin d'un RTOS	Oui mise en œuvre d'un RTOS possible UC/OS II supporté
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui
Autres protocoles	ARP, ICMP…
Interface de programmation	Oui Sockets BSD
Processeurs testés	PowerPC, 68K, ARM, x86
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne, programmation sockets
Drivers fournis	Oui
Prix	8250 $ pour RTIP 2200 $ pour le serveur web pas de royalty

SOLUTIONS LOGICIELLES : INTERNICHE

- La société INTERNICHE TECHNOLOGIES propose une pile TCP/IP et ses sources. A travers ses produits, on a accès à TCP/IP, UDP, PPP, ARP, ICMP, DHCP, SMTP et serveur Web.
- Il n'y a pas obligation d'utiliser un noyau temps réel mais l'intégration des produits INTERNICHE avec un RTOS est toujours possible (par exemple uC/OS II)…
- Les produits INTERNICHE intéressants sont :
  - ♦ Portable TCP/IP.
  - ♦ WebPort.

SOLUTIONS LOGICIELLES : INTERNICHE

- Produit INTERNICHE TCP/IP :

Adresse web	www.iniche.com
Solution	Logicielle
Interfaces Ethernet	Oui par driver d'interface
Besoin d'un RTOS	Non, mise en œuvre d'un RTOS possible UC/OS II supporté
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui
Autres protocoles	ARP, ICMP, BOTP…
Interface de programmation	Oui Sockets BSD
Processeurs testés	ARM
Qualité de la documentation	?
Facilité de programmation	Très bonne, programmation sockets
Drivers fournis	?
Prix	18000 $ avec WebPort revendeur : Emulations www.emulations.fr

SOLUTIONS LOGICIELLES : INTERNICHE

- Produit INTERNICHE WebPort :

Adresse web	www.iniche.com
Solution	Logicielle
Interfaces	Oui par Treck Real-Time TCP/IP
Besoin d'un RTOS	Non, mise en œuvre d'un RTOS possible
Implémentation niveau HTTP	Oui version 1.1 Support de HTML version 2.0
Qualité de la documentation	?
Prix	18000 $ avec TCP/IP revendeur : Emulations www.emulations.fr

POINT 3 :
SOLUTIONS MIXTES POUR LA CONNECTIVITE IP

SOLUTIONS MIXTES

- D'autres solutions mixtes (matériel et logiciel) existent et permettent toutes une connectivité IP immédiate à Internet généralement par liaison série.
- Ces solutions utilisent un processeur (microcontrôleur) dont des broches d ’E/S sont dédiées à la mise en place de la connectivité IP (liaison série, contrôle d ’une interface Ethernet).
- La connectivité IP est intégrée en dur dans le processeur ou apparaît comme une bibliothèque de services (fonctions) à lier avec son application.

SOLUTIONS MIXTES : ICHIP

- La société CONNECT ONE propose pour assurer une connectivité IP des produits basés sur un circuit spécifique qu'ils ont créé et commercialisent : circuit iChip.
- Ce circuit assure une connectivité Internet par l'intermédiaire d'un modem connecté à RTC via le protocole PPP.
- L'interface hôte est du type UART en utilisant des commandes ASCII AT conformes à la norme HAYES.
- Leur produit permet en fait de recevoir, émettre des emails et des pages HTML.
- Le circuit iChip implémente ainsi les protocole PPP, IP, UDP, TCP et SMTP.

SOLUTIONS MIXTES : ICHIP

Adresse web	www.connectone.com
Solution	Matérielle Circuit iChip PLCC 68 broches
Interfaces Ethernet	Non Accès par modem jusqu'à V.90
Implémentation niveau MAC	PPP
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui SMTP supporté
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Bonne
Facilité de programmation	Bonne
Drivers fournis	Non
Prix des drivers	-
Prix du composant	50$ (< 100) revendeurs : Impact Memec

SOLUTIONS MIXTES : S7600A

- La société SEIKO propose un circuit pour assurer une connectivité Internet : circuit S-7600A appelé aussi iChip.
- Ce circuit assure une connectivité Internet par l'intermédiaire d'un modem connecté à RTC via le protocole PPP.
- L'interface hôte est compatible avec la famille 68K de Motorola et x86 d'Intel. Un kit de développement est proposé pour développer des applications ainsi qu'une carte d'évaluation.

SOLUTIONS MIXTES : S7600A

Adresse web	www.seiko-usa-ecd.com
Solution	Matérielle Circuit iChip S-7600A QFP 48 broches
Interfaces Ethernet	Non Accès par modem
Implémentation niveau MAC	PPP
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui
Schémas de principe d'utilisation	Oui Kit de développement disponible
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne
Drivers fournis	-
Prix des drivers	-
Prix du composant	199$ (< 100)

SOLUTIONS MIXTES : SCENIX

- La société SCENIX propose des microcontrôleurs RISC de la famille SX permettant une connectivité Internet en utilisant des broches d'E/S.
- SCENIX introduit le concept de périphériques virtuels (*Virtual Peripheral*) qui se présentent sous forme de bibliothèques logicielles utilisant une ou plusieurs broches d'E/S du microcontrôleur.
- Comme précédemment, on assure une connectivité Internet par l'intermédiaire d'un modem connecté à RTC via le protocole PPP.

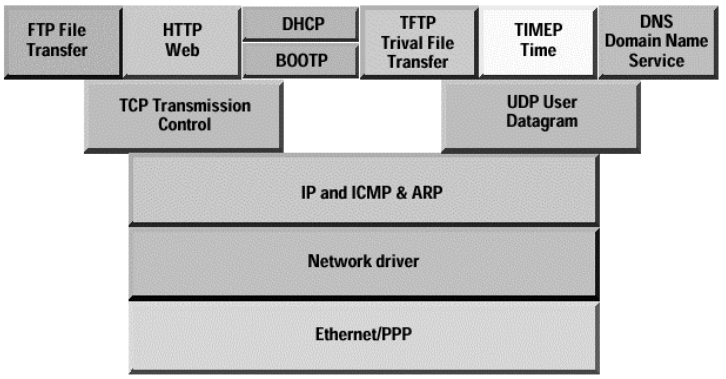
SOLUTIONS MIXTES : SCENIX

Adresse web	www.scenix.com
Solution	Matérielle et logicielle Microcontrôleurs SX
Interfaces Ethernet	Non Accès par modem
Implémentation niveau MAC	PPP Ajout d'une interface Ethernet possible
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui SMTP, HTTP supportés suivant le choix du microcontrôleur
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne
Drivers fournis	-
Prix des drivers	- revendeur : A2M
Prix du composant	-

SOLUTIONS MIXTES : EZ80

- Le produit eZ80 de Zilog est une évolution du célèbre Z80 auquel on a rajouté une connectivité IP. Le CPU n'implémente pas d'interface Ethernet IEEE 802.3. Elle est réalisée par l'ajout d'un circuit externe. La carte d'évaluation eZ80 utilise le circuit CS8900A présenté précédemment.
- L'intérêt réside dans les protocoles Internet fournis pour le eZ80
  - IP, TCP, UDP, ARP, RARP, ICMP, PPP, HTTP, DHCP/BOOTP, SLIP, SMTP, SNMP, Telnet, TFTP.
  - utilitaire de configuration.
  - convertisseur HTML to C.
  - driver Ethernet (CS8900).
  - compilateur C

SOLUTIONS MIXTES : EZ80



SOLUTIONS MIXTES : EZ80

Adresse web	www.zilog.com/ez80/
Solution	Mixte
Description du matériel	microprocesseur Z80 (version eZ80190) 50 MHz processor, multiply and accumulate engine, 16 Mbyte, linear addressing, 3.3V operation, 2 DMA channels, Universal ZiLOG Interface (selectable UART, I2C, SPI), 6 PRTs with prescalers 8KB SRAM, 32-bit GPIO with interrupt support, On-chip oscillator, Optimized pipeline architecture, ZiLOG Debug Interface (ZDI)
Interfaces Ethernet	non la carte d'évaluation utilise le circuit CS8900A
Implémentation niveau MAC	Non
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui ARP, RARP, ICMP, PPP, HTTP, DHCP/BOOTP, IGMP, SLIP, SMTP, SNMP, Telnet, TFTP
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne
Besoin d'un RTOS	Non
Interface de programmation	non
Support	oui
Prix	- Revendeur en France : Futur Electronics carte d'évaluation disponible

POINT 4 :
SOLUTIONS CLE EN MAIN POUR LA CONNECTIVITE IP

SOLUTIONS CLE EN MAIN

- Cette partie présente des solutions clé en main alliant à la fois matériel et logiciel. Il n'y a pas de développement matériel, l'essentiel se résume au développement de son application logicielle...

- On trouvera en fait deux sortes de produits :
  - serveur web embarqué permettant de contrôler des E/S. La connectivité Internet assure un contrôle à distance de ces E/S via un navigateur client.
  - système d'exploitation Linux " allégé " embarqué sur une plateforme matérielle utilisant généralement un microcontrôleur. La connectivité IP est assurée pleinement par Linux où les piles de protocoles Internet sont intiment liées au noyau.

SOLUTIONS CLE EN MAIN

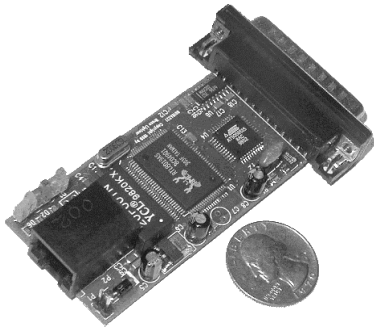
- Linux embarqué :
  - Il est clair que cet OS, fiable, disponible au niveau source sous licence GPL se prête plus qu'aucun autre à des portages sur des plateformes autres que des PC.
  - Cette solution est une voie d'avenir dans l'embarqué avec en plus une extension Temps Réel possible (RTlinux, RTAI).
  - L'adresse web collectant les projets linux embarqué est www.linuxembedded.com .

SOLUTIONS CLE EN MAIN : PICOWEB

- La société LIGHTNER ENGINEERING propose un serveur web embarqué appelé PICOWEB sur une carte possédant un accès IEEE 802.3 10BaseT.
- C'est un système autonome autorisant une connectivité Internet via TCP/IP et HTTP.
- La partie matérielle est construite autour d'un microcontrôleur ATMEL AT90S8515 possédant 8 Ko de mémoire flash, 512 octets d'EEPROM et 512 octets de RAM aussi que 32 E/S.

SOLUTIONS CLE EN MAIN : PICOWEB

- Sur la carte est intégré un contrôleur Ethernet REALTEK ainsi qu'un circuit UART. Le dialogue entre un système hôte et PICOWEB se fait d'ailleurs par la liaison série de l'UART, ce qui permet ainsi un dialogue entre un navigateur web et l'hôte. On peut aussi contrôler à distance les E/S restantes du microcontrôleur non utilisées par PICOWEB.



SOLUTIONS CLE EN MAIN : PICOWEB

- LIGHTNER ENGINEERING propose les schémas de son produit PICOWEB (sous licence) ainsi qu'une version allégée avec le logiciel au niveau objet (version *breadboard*) gratuite pour des utilisations non commerciales.

- Un kit de développement complet comprend une carte PICOWEB et est disponible pour 149 $.

- Une licence est à acquérir (9 $ à l'unité) pour chaque produit basé sur PICOWEB vendu. Une licence grand volume est possible.

SOLUTIONS CLE EN MAIN : PICOWEB

SOLUTIONS CLE EN MAIN : PICOWEB

Adresse web	www.picoweb.net
Solution	Mixte
Description du matériel	Atmel AT90S8515, 8 Ko flash, 512 o EEPROM 512 o RAM, 32 E/S, contrôleur Ethernet Realtek, UART
Interfaces Ethernet	10BaseT
Implémentation niveau MAC	Oui
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui HTTP
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne
Besoin d'un RTOS	Non
Interface de programmation	Non, dialogue par la liaison série
Support	Web support@lightner.net
Prix	145 $ pour le kit de développement royalty pour le firmware : 9 $ par PICOWEB

SOLUTIONS CLE EN MAIN : IPC@CHIP

- Le produit IPC@CHIP (versions SC01, SC02, SC11, SC12) est une solution mixte se présentant sous forme d'un boîtier DIL 32 broches incorporant hardware et software.
- Ce boîtier englobe en fait un microcontrôleur Intel 80186-80188 à 20 MHz selon la version avec au plus 512Ko de RAM et 512 Ko de Flash ainsi qu'un interface Ethernet IEEE 802.3 10BaseT.
- Un kit d'évaluation est disponible (kit DK40) permettant de tester rapidement l'IPC@CHIP.

SOLUTIONS CLE EN MAIN : IPC@CHIP

- Du point de vue logicielle, l'offre est des plus complètes :
  - noyau Temps Réel embarqué autorisant l'exécution des application DOS : on développe donc son application sous DOS à partir de son PC que l'on télécharge ensuite dans le composant. Cet environnement est bien ciblé car c'est généralement celui des PME !
  - un interpréteur de commandes DOS like.
  - une pile TCP/IP complète implémentant l'interface sockets TCP et UDP.
  - un client DHCP.
  - un serveur Web capable d'exécuter des scripts CGI.

SOLUTIONS CLE EN MAIN : IPC@CHIP

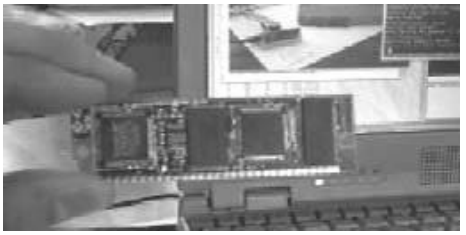
Adresse web	www.bcl-online.de
Solution	Mixte
Description du matériel	Microcontrôleur Intel 80186-80188 à 20 MHz 512Ko de RAM, 512 Ko de Flash 14 I/O programmables, 7 sorties Chip Select, entrées d'interruption INT, PWM, entrée Timer, sortie Timer, 2 UARTs, bus I2C, 2 canaux DMA, watchdog
Interfaces Ethernet	10BaseT
Implémentation niveau MAC	Oui
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui HTTP, ftp, DHCP
Schémas de principe d'utilisation	Oui
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne
Besoin d'un RTOS	Non
Interface de programmation	Oui Sockets BSD
Support	oui
Prix	SC12 : 76,56 euros Licence Run Time pour 1 SC12 : 29 euros kit d'évaluation DK40 : 58 euros

SOLUTIONS CLE EN MAIN : Linux embarqué

- Le projet uClinux fait partie des solutions mixtes où l'on retrouve Linux embarqué. La plateforme matérielle est originellement une carte SIMM 30 broches (kit µCsimm) mettant en œuvre un microcontrôleur MOTOROLA de la famille 68K, le 68EZ328. La carte SIMM possède 8 Mo de DRAM, 2 Mo de flash ROM, un port série RS.232 et une interface Ethernet IEEE 802.3 10BaseT via la circuit CRYSTAL CS8900A.

- Concernant le logiciel, les noyaux linux 2.0.38 et 2.4.x ont été portés sur cette plateforme, ce qui permet de bénéficier naturellement de la connectivité IP !

SOLUTIONS CLE EN MAIN : µClinux



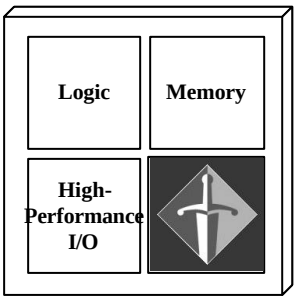
SOLUTIONS CLE EN MAIN : µClinux

Adresse web	www.uclinux.org
Solution	Mixte
Description du matériel	MOTOROLA 68EZ328 16 MHz, 2 Mo flash, 8 Mo RAM, contrôleur Ethernet CRYSTAL CS8900A, UART
Interfaces Ethernet	10BaseT
Implémentation niveau MAC	Oui
Implémentation niveau IP	Oui
Implémentation niveaux TCP, UDP	Oui HTTP et autres
Schémas de principe d'utilisation	Non
Qualité de la documentation	Très bonne
Facilité de programmation	Très bonne
Interface de programmation	Oui Sockets BSD
Support	Web uclinux@uclinux.org
Prix	495 $ pour le kit de développement complet

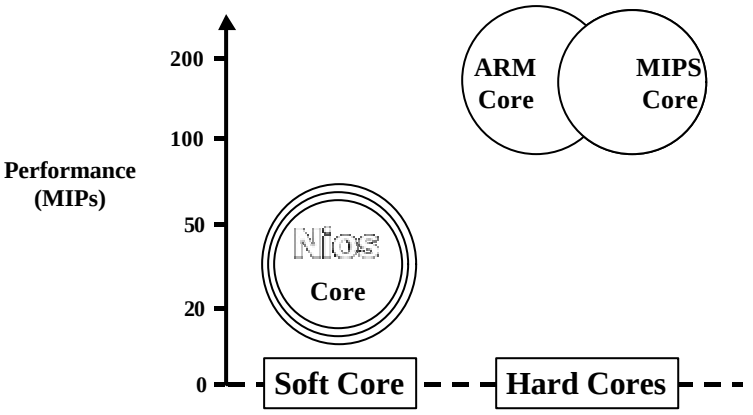
POINT 5 :
NIOS D 'ALTERA
SOLUTION DE CODESIGN
AVEC CONNECTIVITE IP

NIOS D 'ALTERA

- L'offre SoPC Excalibur d'Altera permet la flexibilité de programmation des PLD (*Programmable Logic Device*) avec les performances de temps de traitement d'un processeur embarqué sur silicium pour répondre au besoin d'un court TTM.

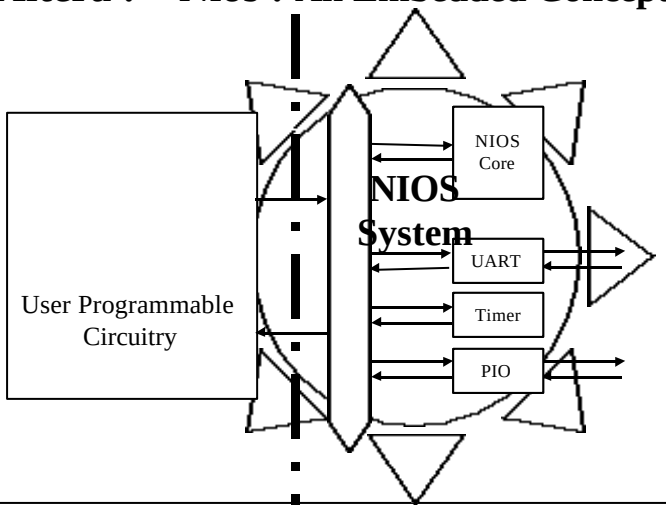


NIOS D 'ALTERA



NIOS D 'ALTERA

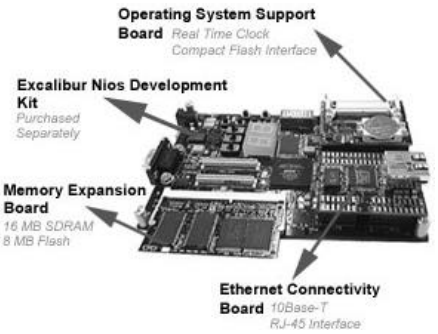
Pour Altéra : « Nios : An Embedded Concept »



NIOS D 'ALTERA

• Linux Development Kit (depuis 09/2001)

• Open-Source µClinux Operating System
• Development Kit Contents
  – µClinux Source Code
  – **Ethernet Development Board**
  – SDRAM / Flash Memory Module
  – SDRAM Controller Core
  – IDE Interface
  – Compact Flash Interface
  – Real Time Clock
  – Reference Design
• Quartus Project
• **Web Server Application**
• Price $2495 (www.microtronix.com)



NIOS D 'ALTERA

- Software Development Tools
  - RedHat GNUPro Toolkit (Compiler, Debugger)
  - Nios Ethernet Development Kit (TCP/IP Stack)

- Operating System Support
  - Linux Development Kit
  - ATI Nucleus
  - µC OS II

NIOS D 'ALTERA

- L'offre SoPC Excalibur/NIOS d'Altera complétée du portage Linux (µClinux) sur NIOS de Microtronix permet d'avoir une véritable plateforme de Codesign.

- Une interface Ethernet IEEE 802.3 10BaseT (utilisant le composant CS8900A) permet d'avoir naturellement une connectivité IP sous µClinux.

PLUS D ’INFORMATIONS

- Plus d ’informations :
  - http://www.enseirb.fr/~kadionik/embedded/embedded.html
  - http://www.enseirb.fr/~kadionik/embedded/connectivite_ip/connectivite_ip.html
  - http://www.enseirb.fr/~kadionik/embedded/uclinux/uclinux.html

PARTIE 7 :
CONNECTIVITE IP : QUELQUES EXEMPLES

INTRODUCTION

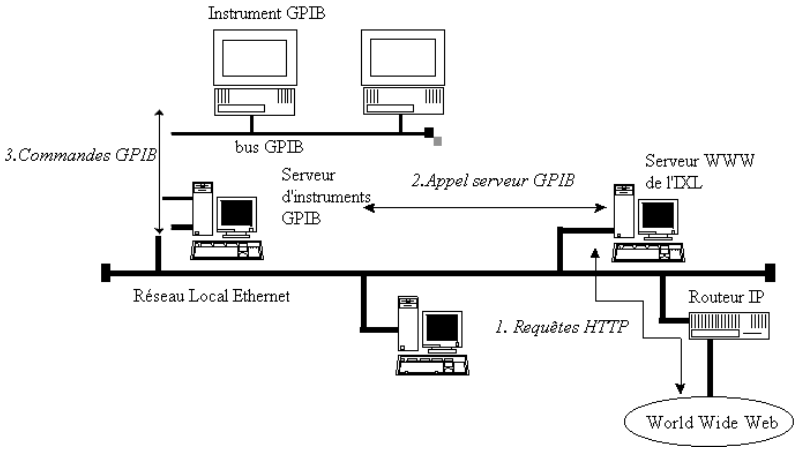
- Quelques exemples de mises en œuvre de la connectivité IP à l ’ENSEIRB sont donnés maintenant à travers 4 projets :
  - téléinstrumentation : projet européen RETWINE (REmoTe Worldwide Instrumentation NEtwork).
  - télémesure : MEDICIS (Mesure à DIstance de CIrcuitS).
  - carte 68HC11ETHER : carte à microcontrôleur 68HC11 avec Internet embarqué.
- Ces 3 projets mettent en œuvre la connectivite IP à travers une liaison Ethernet.

CONNECTIVITE IP : PROJET RETWINE

- Mise en place d ’un parc d’instrumentation depuis Internet pour effectuer des mesures à distance pour :
  - Un partage d’instruments onéreux.
  - Un accès aux instruments facile et offrant des possibilités nouvelles.
  - Une exploitation maximale des décalages horaires.

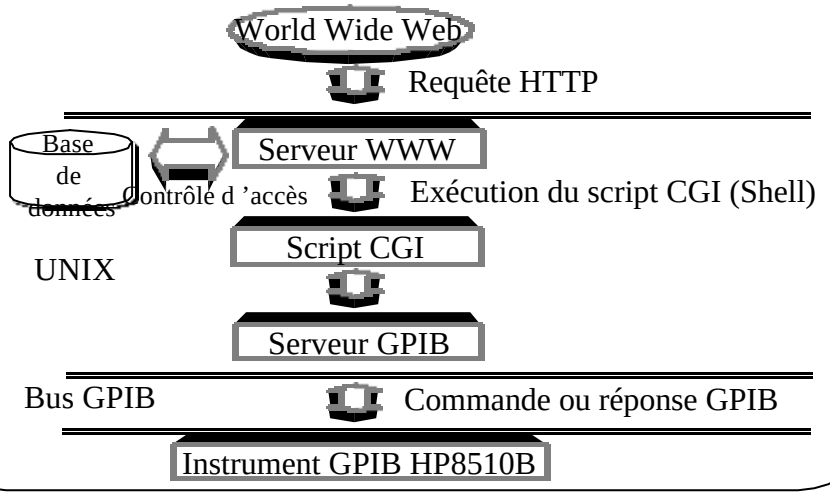
CONNECTIVITE IP : PROJET RETWINE

- Implémentation matérielle :



CONNECTIVITE IP : PROJET RETWINE

- Implémentation logicielle :



CONNECTIVITE IP : PROJET RETWINE

- Analyseur de réseau HP8510B :
- Mesure de paramètres S :
  - coefficients de réflexion et de transmission.
  - taux d'onde stationnaire TOS.
  - impédance.
  - ...

CONNECTIVITE IP : PROJET RETWINE

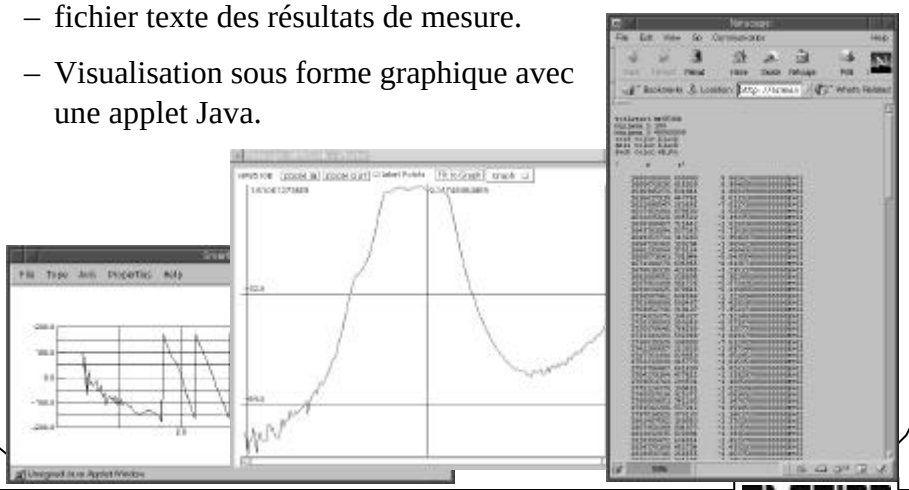
- Driver du HP8510B :
  - Développement en langage C.
  - Contrôle le HP8510B via le bus GPIB.
  - Surcouche NI-488.2M Driver pour des stations de travail SUN.
  - Génération des fichiers de résultats de mesure.

CONNECTIVITE IP : PROJET RETWINE

- Interface graphique (GUI) :
  - Applet Java téléchargée puis exécutée par le navigateur web de celui qui contrôle l 'appareil.
  - Dialogue entre l 'applet Java et le serveur web RETWINE pour le pilotage de l 'instrument.

CONNECTIVITE IP : PROJET RETWINE

- Mesures :
  - fichier texte des résultats de mesure.
  - Visualisation sous forme graphique avec une applet Java.



CONNECTIVITE IP : PROJET RETWINE

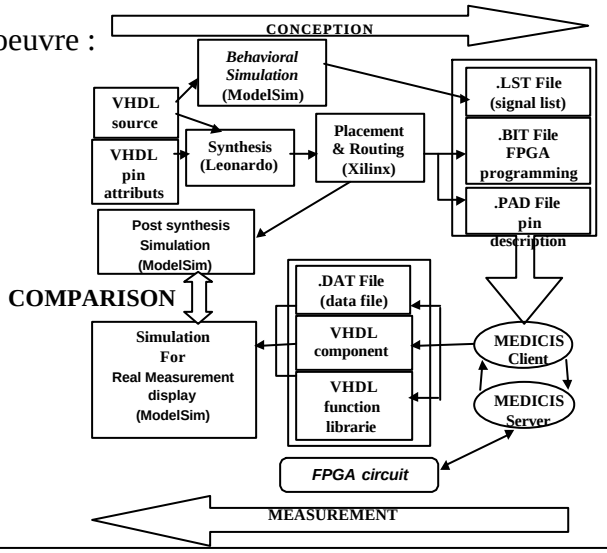
- Plus d 'informations :
  - http://retwine.net
  - http://retwine.ixl.u-bordeaux.fr:8080

CONNECTIVITE IP : PROJET MEDICIS

- Mise en place à l 'ENSEIRB à des fins d 'enseignement d 'un outil qui permet de tester automatiquement un circuit programmable FPGA de XILINX.
- MEDICIS est couplé à la CAO Mentor Graphics :
  - Écriture en VHDL.
  - Simulation avec ModelSim.
  - Synthèse logique avec Leonardo.
  - Programmation, vecteurs de tests issus de la simulation, récupération de la mesure avec MEDICIS.
  - Visualisation des résultats de mesure post synthèse sous ModelSim.

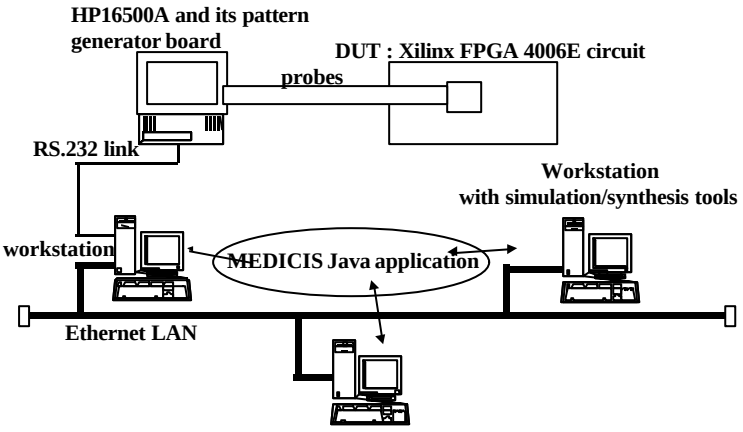
CONNECTIVITE IP : PROJET MEDICIS

- Mise en oeuvre :



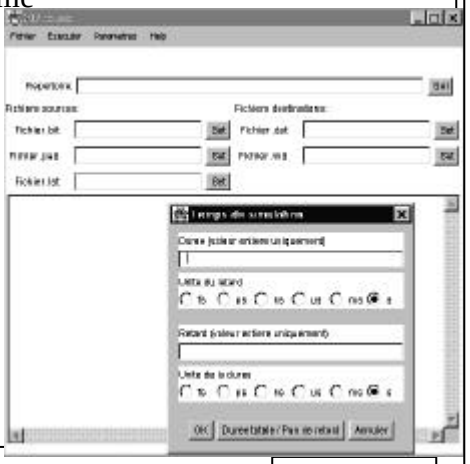
CONNECTIVITE IP : PROJET MEDICIS

- Implémentation matérielle :



CONNECTIVITE IP : PROJET MEDICIS

- Interface graphique (GUI) :
  - Application Java cliente autonome qui dialogue avec une application serveur qui contrôle l 'appareil HP16500.
  - Utilisation ici de l 'API de programmation réseau socket sous Java.



CONNECTIVITE IP : PROJET MEDICIS

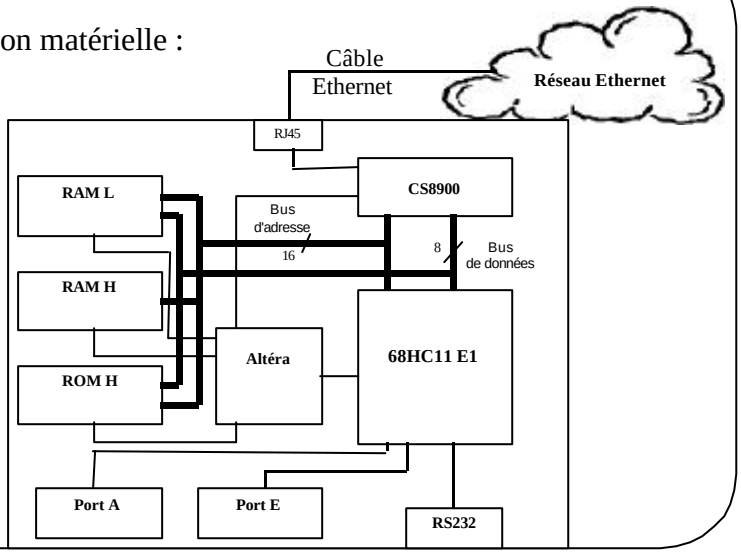
- Plus d 'informations :
  - http://www.enseirb.fr/~nouel/medicis

CONNECTIVITE IP : PROJET 68HC11ETHER

- Développement à des fins d ’enseignement d ’une carte à base de 68HC11 avec une interface réseau IEEE 802.3 10BaseT (circuit CS8900A).
- Utilisation d ’un noyau TR (RTOS) : µC/OS II.
- Écriture en langage C de la suite des protocoles Internet afin d ’assurer la connectivité IP :
  - ARP, ICMP
  - IP, UDP, TCP connexion entrante.
  - Telnet, miniserveur web (page d ’accueil).
- Une des motivations est de voir dans quelle mesure on peut embarquer Internet dans un environnement (très) contraint !

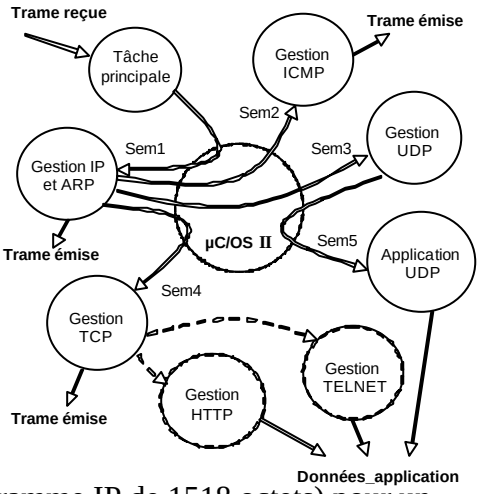
CONNECTIVITE IP : PROJET 68HC11ETHER

- Implémentation matérielle :



CONNECTIVITE IP : PROJET 68HC11ETHER

- Implémentation logicielle :



- Performance : 100 kb/s (datagramme IP de 1518 octets) pour un processeur à 8 MHz (/4) !

CONNECTIVITE IP : PROJET 68HC11ETHER

- Plus d ’informations :
  - http://www.enseirb.fr/~kadionik/68hc11/carteether_enserb/carte_68hc11_ether.html
  - les sources en langage C sont libres d ’accès (GPL).
  - pile TCP/UDP/IP/PPP en libre sous µC/OS II (µC/IP) : http://ucip.sourceforge.net/

PARTIE 8 :
BILAN SUR LA CONNECTIVITE IP

CONNECTIVITE IP : BILAN FINAL

- La connectivité IP permet de raccorder tout système électronique (système embarqué) au réseau Internet. Elle met en œuvre une suite protocoles Internet que l 'on doit embarquer dans le matériel.
- La connectivité IP permet de contrôler un équipement électronique de n 'importe où dans le monde. Cet équipement peut à aussi prévenir un opérateur n 'importe où dans le monde.
- C 'est en fait l 'aboutissement d 'un lent processus de modernisation du télécontrôle allant de la liaison série RS.323/V.24 déportée sur un terminal VT100 à l ' *applet Java* exécutée par un navigateur web interrogeant un serveur web embarqué !

CONNECTIVITE IP : BILAN FINAL

- La connectivité IP présume inconsciemment l 'utilisation d 'interfaces graphiques modernes et banalisées (navigateur web…) en adéquation avec les besoins (de confort) actuels des clients.
- Avec une frontière de plus en plus floue entre matériel et logiciel, on voit apparaître maintenant de véritables offres de *codesign*. En conséquence, l 'ajout de la connectivité IP qui se faisait en grande partie en logiciel a tendance maintenant à être remplacée par son homologue matériel (utilisation d 'un bloc IP).

CONNECTIVITE IP : BILAN FINAL

- Les protocoles Internet sont indépendants des supports de transmission utilisés.
- Les supports de transmission préférentiels sont :
  - Ethernet.
  - Liaison série.
- Des solutions de connectivité IP utilisant des liaisons radio ou sur courant porteur commencent à apparaître...

CONNECTIVITE IP : PETIT SYSTEME

- Solution « maison » :

Taille système	Petit
Liaison	Série Ethernet
Interface réseau	UART Interface Ethernet CS8900A
Composant	• PIC, 68HC11, 68HC12 • Avec prise en compte de la connectivité IP : iCHIP, SEIKO S-7600A, SCENIX, eZ80
Connectivité IP minimale	PPP – IP – UDP ICMP/ARP
Connectivité IP de confort	TCP et plus
Interaction minimale	Par une application spécifique
Interaction de confort	Miniserveur web spécifique
Besoin d'un OS/RTOS	Non On peut utiliser un RTOS (µC/OS II avec µC/IP)

CONNECTIVITE IP : PETIT SYSTEME

- Solution « clé en main » :

Taille système	**Petit**
Liaison	**Série** **Ethernet** **GSM, courant porteur**
Composant matériel	**Suivant le module**
Connectivité IP minimale	**PPP – IP – UDP ICMP/ARP**
Connectivité IP de confort	**Suivant le module**
Interaction	**Suivant le module : serveur web, email, SNMP…**
Solutions	**Picoweb, IPC@CHIP** **Produits eDevice, Webdyn**

CONNECTIVITE IP : MOYEN ET GROS SYSTEME

- Solution « maison » :

Taille système	Moyen et gros
Liaison	Série Ethernet
Interface réseau	UART Interface Ethernet (CS8900A)
Composant	• 68EZ328, ColdFire • NIOS
Connectivité IP minimale	PPP – IP – UDP ICMP/ARP
Connectivité IP de confort	TCP et plus
Interaction minimale	Par une application spécifique
Interaction de confort	Miniserveur web, SNMP, email
Besoin d'un OS/RTOS	Recommandé Linux embarqué : µClinux

CONNECTIVITE IP : MOYEN ET GROS SYSTEME

- Solution « clé en main » :

Taille système	**Moyen et gros**
Liaison	**Série** **Ethernet** **GSM, courant porteur**
Composant matériel	**Suivant le module**
Connectivité IP minimale	**PPP – IP – UDP ICMP/ARP**
Connectivité IP de confort	**Suivant le module**
Interaction	**Suivant le module : serveur web, email, SNMP…**
Solutions	**Linux embarqué : cartes dédiées (ColdFire…)** **Produits eDevice, Webdyn**

CHAPITRE 5 :
LES SYSTEMES EMBARQUES ET LA SECURITE

VULNERABILITE DES SYSTEMES EMBARQUES

- Les systèmes embarqués mettant en œuvre la connectivité IP sont aujourd ’hui potentiellement vulnérables à une attaque par le réseau.
- Les attaques concernent actuellement les routeurs, les imprimantes réseau... mais rien n ’empêche une attaque d ’une maison individuelle avec son réseau domotique ou d ’une voiture connectées à Internet !
- L ’aspect sécurité d ’un système embarqué doit être maintenant pris en compte lors de sa conception. Ce n ’est pas encore dans la mentalité des concepteurs de systèmes embarqués...

VULNERABILITE DES SYSTEMES EMBARQUES

- Les crackers exploitent :
  - Les erreurs de conception matérielle.
  - Les vulnérabilités logicielles :
    - « Backdoor » mise en place par le programmeur à des fins de tests et laissée dans la version finalisée.
    - Ignorance des standards usuels.
    - Mauvaise programmation : « buffer overflow », « stack overflow », test de validité des paramètres d 'entrée…
    - Serveur HTTP vulnérable car léger !

VULNERABILITE DES SYSTEMES EMBARQUES

- Exemples d 'attaques (sur routeur, imprimante réseau) :
  - Modification de l 'adresse IP en utilisant SNMP (communauté par défaut : write).
  - Datagramme UDP spécial pour fermer un port socket.
  - Authentification faible : username=laserjet.
  - Mot de passe en clair accessible par SNMP.
  - Reset par SNMP.
  - L 'écriture par SNMP d 'une grande chaîne de caractères crashe l 'équipement.

VULNERABILITE DES SYSTEMES EMBARQUES

- Soyons optimiste : les « exploits » sur systèmes embarqués sont rares par rapport à ceux concernant les systèmes classiques (PC, routeur…).
- Il y a peu de documentation accessible à disposition (en ligne) pour le cracker sur le fonctionnement interne (matériel et logiciel) d ’un système embarqué.
- Les attaques classiques par shell code sur buffer overflow sont inexistantes par il n ’y a pas de shell (sauf avec linux :-( ).
- Le pire des cas est un crash du système embarqué ou son reboot (ce qui est préférable)...

CHAPITRE 6 :
LINUX EMBARQUE

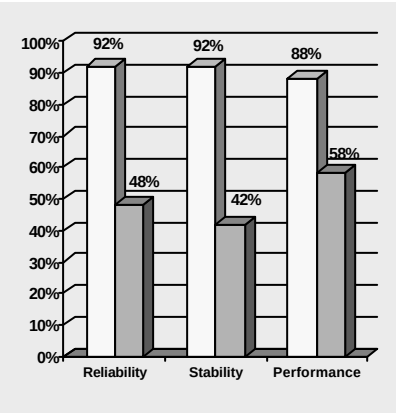
PARTIE 1 :
LE BESOIN D 'EMBARQUER LINUX

L 'OPPORTUNITE DE LINUX SUR MARCHE DE L 'EMBARQUE

- Beaucoup sont passés d 'un OS propriétaire (Microsoft) à Linux pour l 'embarqué malgré encore quelques réticences archaïques :
  - Quelque chose de gratuit est de la camelote (voir le prix plancher psychologique d 'un produit au supermarché).

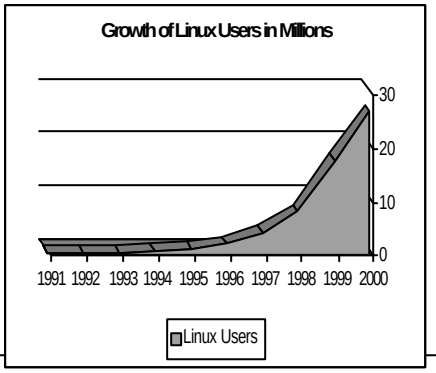
L ’OPPORTUNITE DE LINUX SUR MARCHE DE L ’EMBARQUE

- On retiendra les comparaisons suivantes (d ’après www.survey.com) :



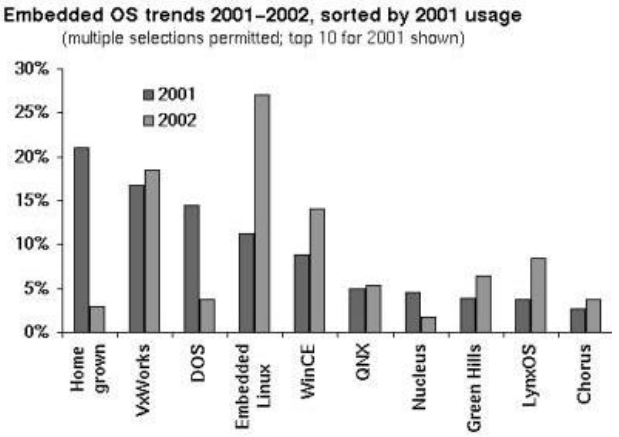
L ’OPPORTUNITE DE LINUX SUR MARCHE DE L ’EMBARQUE

- En 2000, il y avait 27 millions d ’utilisateurs de Linux. IDC prévoit une croissance de 25 % par an. WR Hambert prévoit un chiffre d ’affaire de 2 milliards USD en 2000 à 12 milliards USD en 2003 !



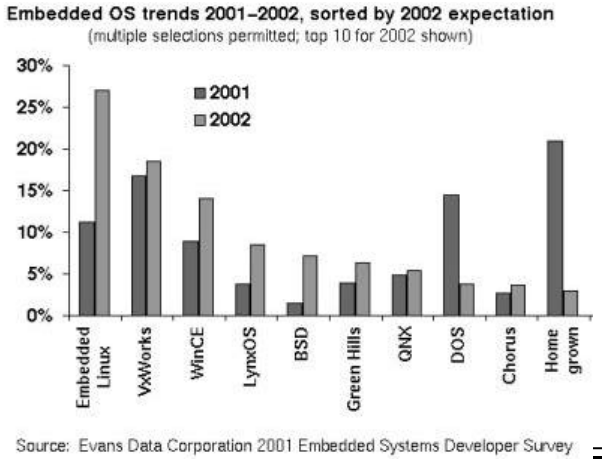
LE MARCHE DE L ’EMBARQUE

- Panorama du marché de l ’embarqué en 2001.



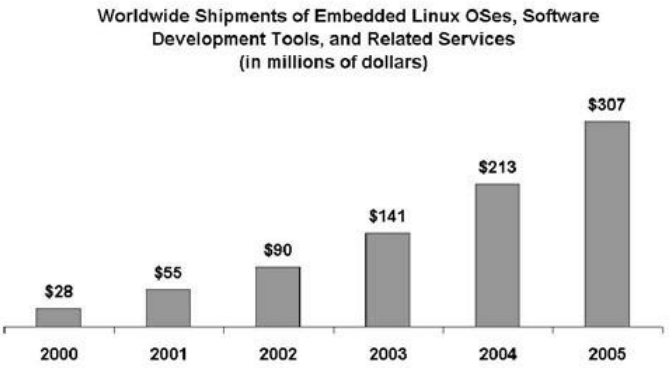
LE MARCHE DE L ’EMBARQUE

- Panorama du marché de l ’embarqué en 2002. Fin 2002, linux embarqué devient la principale plateforme de l ’embarqué !



LE MARCHE DE L ’EMBARQUE

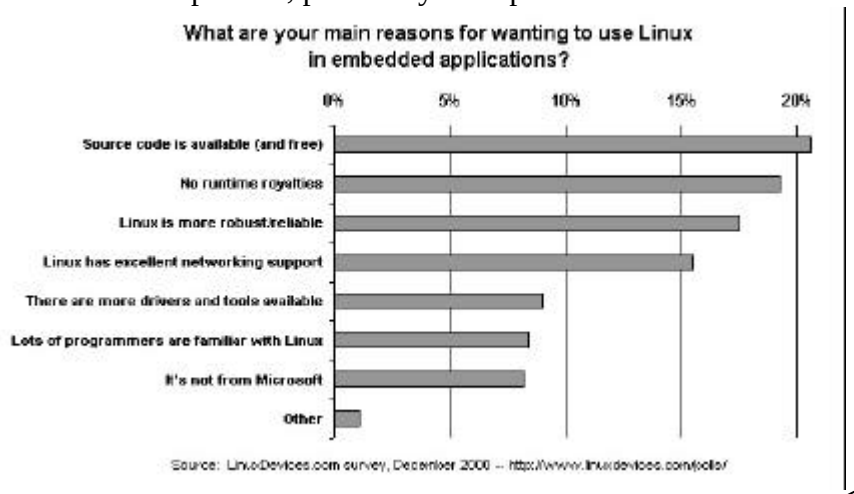
- Croissance de Linux embarqué :



Source: Venture Development Corporation (VDC) 2000 Embedded Linux Market Study

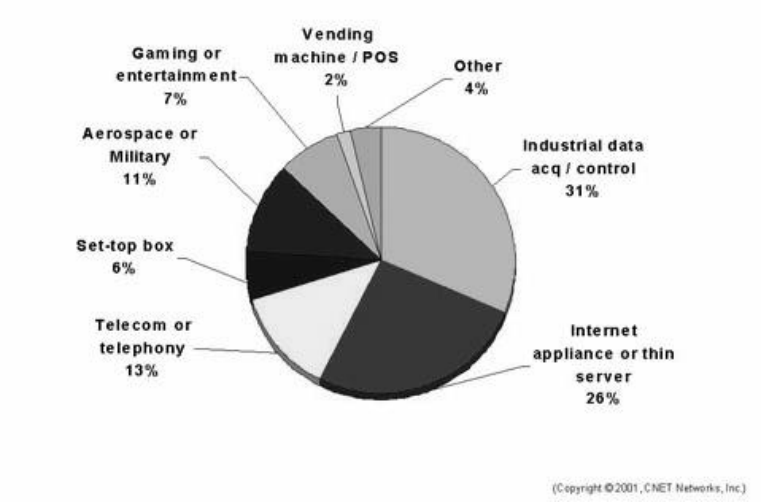
POINTS FORTS CITES

- Code source disponible, pas de royalties pour Linux :



Source: LinuxDevices.com survey, December 2000 -- http://www.linuxdevices.com/polls/

APPLICATIONS VISEES PAR LINUX



QU ’EST-CE QUE LINUX ?

- Linux est un système d ’exploitation libre de type UNIX lancé par le finlandais Linus Torvalds en 1991 avec l ’assistance de milliers de développeurs dans le monde pour son évolution.
- Son succès tient au fait qu ’il est développé sous licence GPL (*General Public License*), ce qui signifie que le code source Linux est disponible à tout le monde et gratuit.
- Son emblême est un pingouin : le *tux*.

QU ’EST-CE QUE LINUX ?

- Linux correspond au cœur du système d ’exploitation : le noyau.
- Linux est stable et robuste.
- Linux tourne originellement sur plateforme i386 et supérieure avec 8 Mo de RAM.
- **IL FAUT DONC UN PROCESSEUR 32 BITS AVEC MMU (OU A DEFAUT 32 BITS SANS MMU AVEC µClinux)**

QU ’EST-CE QUE LINUX ?

- Linux est complété des outils/logiciels GNU (*Gnu is Not UNIX*).
- Linux est disponible sous forme de distributions : Debian, RedHat, Mandrake, SuSE, Slackware...
- Linux est utilisé avec une interface graphique comparable à Microsoft Windows : Gnome, KDE

POURQUOI UTILISER LINUX ?

- Linux est open source :
  - Le code source est disponible au public.
  - Le code source inclut :
    - Le noyau Linux.
    - Les pilotes de périphériques (*drivers*).
    - Un ensemble de petits utilitaires (MAKEDEV…).
- On peut ainsi voir directement à travers les fichiers sources ce que fait le noyau Linux voire modifier son comportement au besoin. On n 'a donc pas une boîte noire (avec comme seul interlocuteur une hot-line !).

POURQUOI UTILISER LINUX ?

- Linux est fiable :
  - Grâce à une gestion mémoire optimisée, Linux peut tourner sur une machine des années sans plantage et sans « écran bleu de la mort ».
- Linux est extensible :
  - Une application Linux écrite pour une plateforme PC peut être facilement portée sur une plateforme Linux embarquée.
  - Cette même application peut être aussi facilement portée sur un cluster Linux (grappe d 'ordinateurs coopératifs).

POURQUOI UTILISER LINUX ?

- Linux est sécurisé :
  - Linux est recommandé par le NSA américain.
  - Linux est conçu pour que les processus ne puissent pas lire en mémoire code et données sans provoquer une violation des règles de sécurité du système (*segmentation violation*). Cela permet de confiner les programmes malicieux.
  - Sécurisation du système de fichiers avec des droits d 'accès.
  - Sécurisation d 'accès physique à la plateforme.
  - Sécurisation de l 'accès réseau.

POURQUOI UTILISER LINUX ?

- Linux supporte la plus large palette de protocoles réseau testés et éprouvés (indispensable pour la connectivité IP dans l 'embarqué) :
  - TCP/IP networking.
  - Routing/Firewalling.
  - Web Server.
  - FTP Server.
  - Telnet Server.
  - SMB.
  - NFS.
  - protocoles WAN : X.25, AX.25, HDLC, ATM.
  - ...

POURQUOI UTILISER LINUX ?

- Linux possède un support efficace à travers la communauté de développeurs.
- On trouve toujours une application Linux correspondant à son besoin (ou très proche).
- On capitalise son expérience UNIX en travaillant sous Linux car Linux est UNIX like d 'où des coûts de formation réduits.

POURQUOI UTILISER LINUX ?

- Les coûts de mise en œuvre de Linux sont réduits :
  - Toutes les distributions Linux sont disponibles gratuitement au téléchargement par Internet.
  - On peut acheter une distribution (< 150 euros) avec la documentation papier et un service support de 30 jours généralement.
  - Les outils de développement (compilateurs, IDE…) sont disponibles à faible coût ou gratuits (GNU).

POURQUOI UTILISER LINUX ?

- Linux est **sans royalties** à payer pour chaque produit vendu à base de Linux.
- Ce point est une (r)évolution dans le domaine de l ’embarqué où les outils (OS, IDE…) sont chers et où l ’on paye en plus des royalties non négligeables sur chaque produit conçu avec.

LINUX ET LE LOGICIEL LIBRE

- Linux est un logiciel libre : cela donne le pouvoir aux utilisateurs d ’utiliser ce logiciel comme ils l ’entendent :
  - **“Free software is a matter of liberty, not price …‘free’ as in ‘free speech,’ not as in ‘free beer’…”. Free Software Foundation**
- Le développement n ’est pas contrôlé par un petit groupe de développeurs donc pas de despotisme possible.
- Il est possible de gagner de l ’argent avec le logiciel libre (formation, assistance…).

L ’OPEN SOURCE

- L *’open source* : accès aux sources du logiciel.
- L ’open source permet :
  - Une interopérabilité entre applications et les différentes plateformes.
  - La formation par analyse des sources.
  - L ’accès aux sources permet d ’optimiser des parties de code pour des performances accrues.
  - Les idées et algorithmes deviennent des standards et sont disponibles à tous sans brevet.
  - Des distributeurs développent et vendent leurs fonctionnalités au dessus de logiciels open source.
  - Le terme *open source* est plus vendeur que logiciel libre.

LOGICIEL LIBRE

- Définition technique (*free software*) d ’après la FSF (*Free Software Foundation)* :

Users have the freedom to :

(1) run the software, for any purpose;

(2) study how the program works and adapt it to their needs;

(3) redistribute copies;

(4) improve the program and release improvements to the public

Access to source code is necessary for (2) and (4) so “Free” can include “Open Source”

LOGICIEL LIBRE

- En conséquence, le logiciel libre peut être modifié, utilisé et même vendu.
- Vendre un logiciel libre correspond à ajouter un service, un bonus :
  - Outils d'installation, de packaging de logiciels.
  - Aide, support, formation.
  - Adaptation de logiciel à un besoin spécifique.
  - Driver d'un matériel sous forme d'un module Linux (founiture du fichier objet .o).
- Des améliorations d'un logiciel libre peuvent être proposées par tous sous forme d'une nouvelle release.

LOGICIEL LIBRE : DROITS (LIBERTES)

- Liberté d'exécuter le programme.
- Liberté d'étudier et de modifier le programme afin de l'adapter à vos besoins.
- Liberté de copier et de redistribuer des copies avec ou sans modifications.
- Liberté de modifier (ou faire modifier) le code source et rendre public les modifications.

LOGICIEL LIBRE : OBLIGATIONS

- Mise à disposition du code source.
- Les modifications apportées au programme doivent être clairement indiquées et datées (Changelog).
- Un programme sous GPL reste un programme sous GPL.

LOGICIEL LIBRE : OBLIGATIONS

Pour distribuer un programme sous GPL :

- Transmettre tous les droits que vous possédez.
- S'assurer que les destinataires reçoivent le code source ou peuvent se le procurer.
- Leur remettre la licence GPL afin qu'eux aussi connaissent leurs droits.

LICENCE OPEN SOURCE

- Une licence logicielle précise ce que l 'utilisateur peut faire avec un logiciel et son code
- Une licence traditionnelle (commerciale) précise strictement l 'utilisation du logiciel acheté.
- Une licence open source indique comment le code peut être utilisé, réutilisé et redistribué. La licence généralement mise en œuvre est la licence GPL.

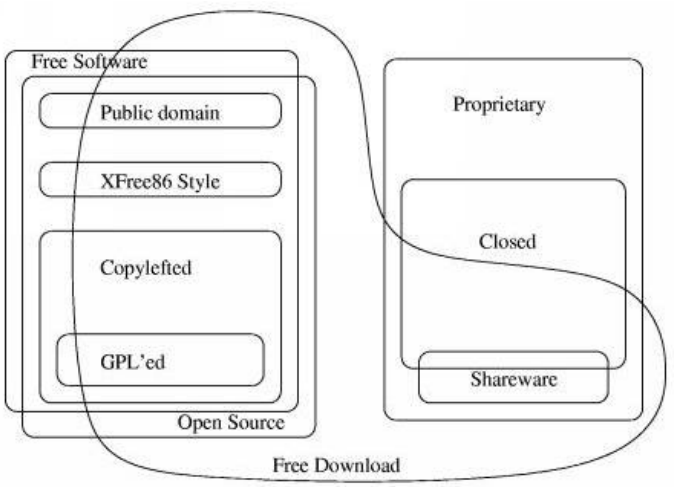
LICENCE OPEN SOURCE GPL

- La licence GPL est le modèle de distribution idéal d 'un logiciel proposé par la FSF et le projet GNU.
- Les développeurs peuvent choisir de licencier leur logiciel sous licence GPL. Cela exige que les utilisateurs maintiennent le code source originel et indiquent clairement les changements opérés avant toute redistribution.
- Le code source est disponible, les utilisateurs peuvent le modifier, le compiler comme ils veulent.
- *Copyleft* : les utilisateurs possèdent les mêmes droits pour toute version du logiciel.

AUTRES LICENCES

- Il existe d ’autres types de licences :
  - MIT.
  - BSD.
  - X11, XFree86.
  - Netscape.
  - W3C.

AUTRES LICENCES



Source: Free Software Foundation

LINUX ET LE TEMPS REEL

- Un système d ’exploitation est dit Temps Réel s ’il est capable de répondre à des sollicitations ou événements (internes ou externes) dans un temps maximum.

- On parle de Temps Réel mou (*Soft Real Time*) quand les événements sont traités trop tardivement ou perdus et sans conséquence catastrophique pour la bonne marche du système.

- On parle de Temps Réel dur (Hard *Real Time*) quand les événements traités trop tardivement ou perdus provoquent des conséquences catastrophiques pour la bonne marche du système (perte d ’informations, plantage…).

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - Le noyau Linux possède de longues sections de codes où tous les événements extérieurs sont masqués (non interruptible).

    ET
  - Le noyau Linux n ’est pas préemptible durant l ’exécution d ’un appel système. Un processus Linux de faible priorité pour l ’application fait un appel système. En cours d ’exécution de l ’appel système, un événement extérieur active un processus de plus forte priorité qui ne sera exécuté qu ’à la fin de l ’exécution complète de l ’appel système : inversion de priorité fatale à tout système Temps Réel.

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - L ’ordonnanceur de Linux essaye d ’attribuer de façon équitable le CPU à l ’ensemble des processus (ordonnancement de type *old aging* mise en œuvre pour favoriser l ’accès CPU aux processus récents). C ’est une approche égalitaire. Un ordonnanceur Temps Réel donnera toujours la main à la tâche de plus forte priorité prête. C ’est ici un approche plus totalitaire.

LINUX ET LE TEMPS REEL

- Le noyau Linux standard peut être considéré comme Temps Réel mou si l ’on travaille avec une réactivité de l ’ordre de la centaine de ms et plus.

- Il existe des solutions Linux Temps Réel dur (et mou)...

LINUX ET LA PORTABILITE

- Linux (et ses applications) est fortement portable.

- Une même application peut être utilisée (portée) sur :
  - Un nombre important de processeurs : x86, Alpha, ARM, StrongARM, MIPS, PowerPC, SPARC, m68k...
  - Un nombre important de plateformes ou BSP (*Board support Package*).
  - Un nombre important d 'interfaces physiques avec le driver adéquat.
- Linux est donc capable d 'exécuter la même application du PDA à l 'ordinateur de bureau.
- Linux est un système d 'exploitation de choix pour les systèmes embarqués. On parle de Linux embarqué.

LINUX EMBARQUE

- Linux embarqué est une adaptation du noyau Linux à un système embarqué. Suivant les capacités du système, on ne retrouve qu 'une partie des fonctionnalités du noyau :
  - Moins de services disponibles.
  - Moins de mémoire requise (< 8 Mo).
  - Boot depuis une mémoire ROM.
  - Pas de clavier ou de souris requis.
  - Logiciels spéciaux pour piloter les périphériques du système (écran LCD, flash disk, Disk On Chip DOC, touch screen…).

LINUX EMBARQUE

- Une version de Linux embarqué peut être spécialement configurée pour coller à une plateforme ou application précise :
  - Linux embarqué pour routeur IP.
  - Linux embarqué sur PDA.
  - Linux embarqué pour microcontrôleur sans MMU.
  - Linux embarqué sur processeur 80286 et inférieur.
  - ...

OUTILS POUR LINUX EMBARQUE

- On utilise pour le développement sous Linux embarqué les outils traditionnels GNU :
  - (cross) compilateurs C/C++. C est préférable pour limiter la taille des exécutables.
  - IDE.
  - GDB.
  - Simulateur.

OUTILS POUR LINUX EMBARQUE

- On utilise pour le développement sous Linux embarqué un PC de développement sous Linux (l 'hôte) avec une chaîne de compilation croisée en fonction du processeur embarqué sur le système (la cible).
- L 'exécutable ainsi produit est téléchargé dans la cible pour pouvoir y être testé. On utilisera alors GDB pour débugger l 'application par le réseau que l 'on pourra coupler avec une interface graphique de type DDD.
- Un montage NFS depuis la cible d 'un répertoire du PC hôte permet de simplifier la phase de téléchargement.

OUTILS POUR LINUX EMBARQUE

- Il existe des simulateurs tournant sur le PC hôte pour simuler la cible :
  - Simulateur pour émuler une grande marque de pocket PC.
- Il est possible d 'utiliser d 'émuler complètement un système sur le PC hôte en utilisant le projet UML (*User Mode Linux*). UML permet de créer une machine virtuelle tournant un Linux embarqué correspondant à la cible et à sont type de processeur. Cela permet alors de compiler une application directement en natif si l 'on se connecte à cette machine virtuelle…

  http://user-mode-linux.sourceforge.net/

OUTILS POUR LINUX EMBARQUE

- Il est possible d 'utiliser des IDE commerciaux :
  - CodeWarrior de Metrowerks. Fonctionne avec les versions Linux embarqué de LynuxWorks (BlueCat), Lineo/Metrowerks/Motorola et Montavista.
  - Microsoft Visual Studio. Fonctionne avec la version Linux embarqué de LynuxWorks.

OUTILS POUR LINUX EMBARQUE

- Java est aussi supporté.
- Il est possible aussi d 'utiliser des interfaces graphiques légères :
  - Microwindows.
  - Nano-X
  - Qt Embedded de Troltech (et dérivés Qtopia, OPIE).
  - … (frame buffer)

LE CHOIX D 'UN PROCESSEUR POUR L 'EMBARQUE

Besoin	Miniature	Petit	Moyen	Haut de gamme	PC embarqué	Embarqué haute disponibilité
Taille RAM	<0.1 Mo	0.1-4 Mo	2-8 Mo	8-32 Mo	16-64 Mo	> x Mo
Taille ROM/FLASH	0.1-0.5 Mo	0.5-2 Mo	2-4 Mo FLASH	4-16 Mo FLASH	xx Mo	Go-To
Processeurs	DragonBall 68K Mcore ColdFire ARM		MIPS Hitachi SH x86 PowerPC			Pentium PowerPC
Caractéristiques matérielles	MMU optionnelle		Ardoise Internet Carte unité centrale System on Chip (SoC)			CompactPCI
Exemples d'applications	Caméra numérique PDA Téléphone		Routeur Décodeur Stockage en réseau Imprimante en réseau			Commutateur téléphonique Routeur haute performance Serveur central

- Choix suivant puissance de calcul, taille mémoire...

PROCESSEURS SUPPORTES POUR LINUX EMBARQUE

- Cela dépend essentiellement de la distribution Linux embarqué :

- Par exemple, MontaVista supporte :
  - Intel (x86).
  - PowerPC.
  - MIPS.
  - StrongARM.
  - Hitachi Super-H.

http://www.mvista.com/products/hardware.html

PROCESSEURS SUPPORTES POUR LINUX EMBARQUE

- Par exemple, LynuxWorks BlueCat Linux supporte :
  - x86.
  - Motorola PowerPC.
  - MIPS R3 & R4.
  - StrongARM.
  - Hitachi Super-H.

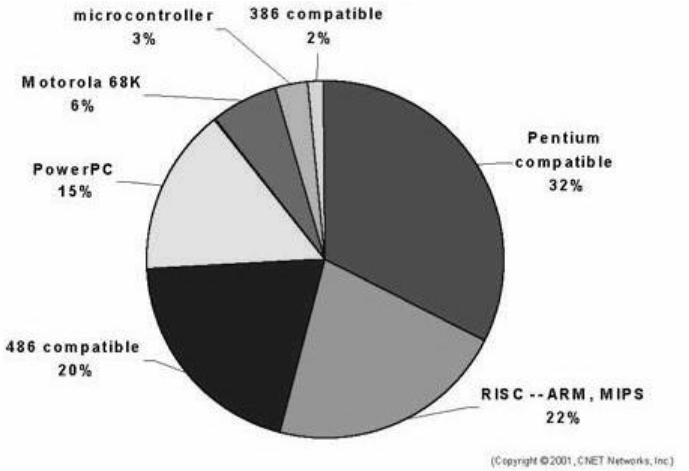
http://www.linuxworks.com/bluecat/index.html

PROCESSEURS SUPPORTES POUR LINUX EMBARQUE

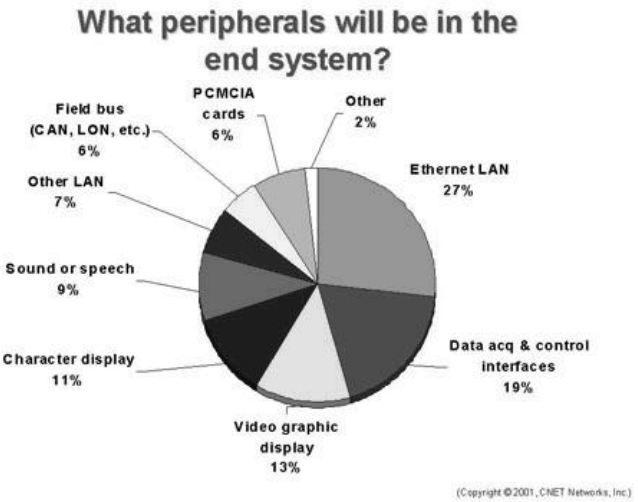
- Par exemple, Lineo/Metrowerks/ Motorola supporte :
  - x86.
  - PowerPC.
  - StrongARM.
  - **Motorola 683xx et ColdFire.** (Lineo était à l'origine du projet µClinux)

http://www.metrowerks.com/embedded/

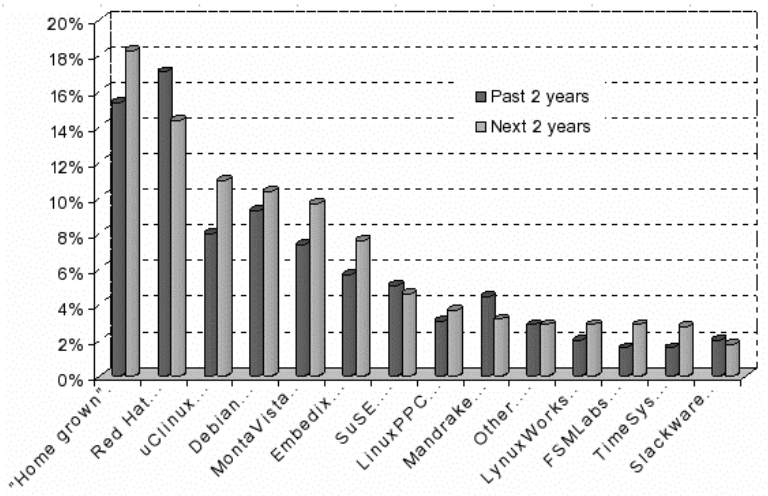
CHOIX DU PROCESSEUR POUR LINUX EMBARQUE



PERIPHERIQUES POUR LINUX EMBARQUE

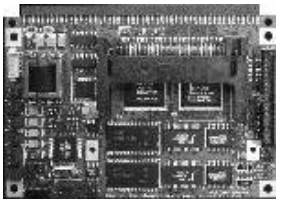


CHOIX D ’UN LINUX EMBARQUE



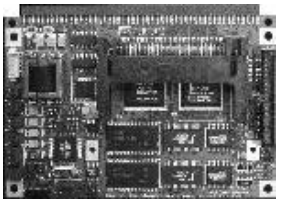
- Enquête linuxdevices.com juin 2003

CARTES POUR LINUX EMBARQUE



- Little Board (5.75 x 8.0 in.) -- complete systems on a single compact board, expandable with plug-on function modules
- ISA "slot boards" (full-length, 13.8 x 4.8 in.; half-length, 7.1 x 4.8 in.) -- IBM PC plug-in cards which could function as standalone SBCs backplanes)
- PC/104 modules (3.6 x 3.8 in.) -- compact, rugged, self-stacking modules featuring a reliable pin-and-socket board-to-board expansion bus

CARTES POUR LINUX EMBARQUE



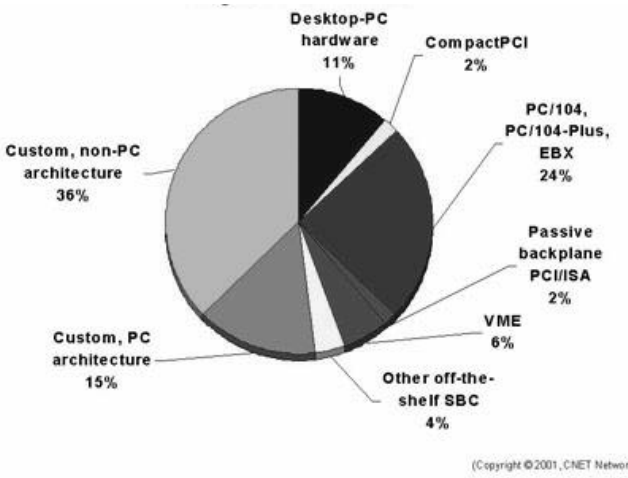
- Bus PCI en plus :

  PC/104-Plus -- PCI added to PC/104

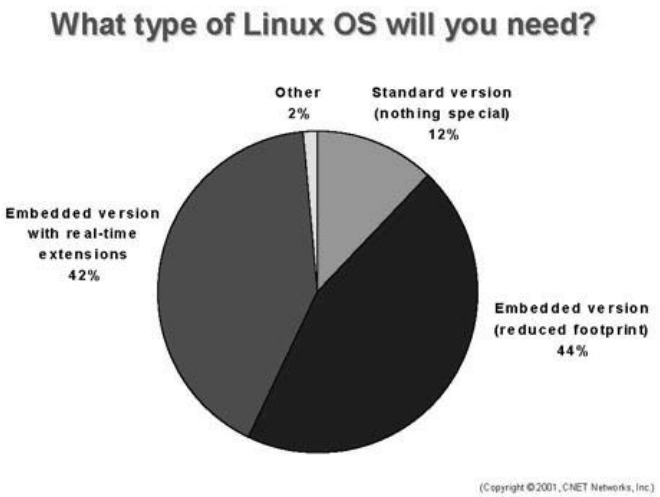
  EBX -- PC/104-Plus added to Little Board

- Cartes au format industriel VME, VXI, PXI...

FORMAT DES CARTES CHOISI POUR LINUX EMBARQUE



CHOIX D ’UNE VERSION LINUX EMBARQUE



POINTS FAIBLES DE LINUX EMBARQUE

- Les drivers Linux pour un périphérique donné ne sont pas toujours disponibles.
- Le manque de standards : window manager, GUI, extensions Temps Réel…
- Le manque d’une cohérence marketing.
- Le manque d’outils de qualification d’un système sous Linux (tests de conformité de l’API POSIX pour le Temps Réel ?).
- Le modèle de la licence GPL mal compris (droits et surtout obligations).

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- On pourrait être effrayé de prime abord par la multitude de l'offre Linux embarqué pour faire un choix correspondant à son besoin (s'il est bien défini !).
- Le consortium ELC (Embedded Linux Consortium) a proposé un document de standardisation des applications Linux embarqué.
- Le document ELC Platform Specification (ELCPS) de propose de définir les environnements de programmation basé sur un système Linux embarqué.

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- Le document ELCPS est basé sur un ensemble de standards :
  - The Linux Standards Base 1.2 (LSB)
  - IEEE POSIX 1003.1-2001
  - The Single UNIX Specification v3
- Le document ELCPS de propose de promouvoir :
  - Le développement de systèmes et d'applications Linux embarqué
  - La portabilité des applications

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- Le document ELCPS est basé sur un ensemble de standards :
  - The Linux Standards Base 1.2
  - IEEE POSIX 1003.1-2001
  - The Single UNIX Specification v3
- Le document ELCPS de propose de promouvoir :
  - Le développement de systèmes et d'applications Linux embarqué
  - La portabilité des applications

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- Le document ELCPS définit 3 types d'environnements système :
  - 1. Environnement système minimal : pas de stockaage de masse , pas d'interaction, profondément enfoui. Monoprocessus.
  - 2. Environnement système intermédiaire : stockage de masse (donc système(s) de fichiers). Cela paut être aussi des systèmes de fichiers en mémoire FLASH. Multiprocessus.
  - 3. Environnement système complet : système général, support réseau, GUI...

VERS UNE STANDARDISATION DE LINUX EMBARQUE ?

- En fonction de ces 3 environnements, des groupes d'appels système de l'API Linux sont inutiles, obligatoires, optionnels:
  - ELC_C_LANG_MATH : acos(), cos()…
  - ELC_DEVICE_IO : getc(), fflush(), puts(), read()…
- Exemple : IPC obligatoire pour un système moyen ou complet.
- Le document ELCPS est disponible sur le site de ELC.

PARTIE 2 : LES OFFRES LINUX EMBARQUE

LES OFFRES LINUX EMBARQUE

- Les offres de version de Linux embarqué (et Temps Réel) peuvent être rangées dans l'une des 3 catégories suivantes :
  - Les distributions Linux classiques : RedHat , Mandrake, Caldera, Debian, Slackware, Suse…
- Suivant la quantité de mémoire "disque" du système embarqué, il est possible d'édulcorer une distribution classique (<100-150 Mo). Cela tient dans une mémoire Compact Flash (512 Mo…).
- Le projet LFS (Linux From Scratch) explique comment construire son Linux pas à pas depuis rien suivant ses besoins : http://www.linuxfromscratch.org/

LES OFFRES LINUX EMBARQUE

- Les offres de version de Linux embarqué (et Temps Réel) peuvent être rangées dans l'une des 3 catégories suivantes :
  - Les distributions Linux embarqué commerciales :
    - non TR : Montavista/Professional or Carrier Grade or Consumer Electronics Edition (ex Hard Hat Linux), Lineo-Metrowerks-Motorola/Creation Suite for Linux (ex Embeddix), LynuxWorks/BlueCat, RedHat/Embedded
    - TR : FSMLabs/RTLinux Pro, Montavista/ Professional or Carrier Grade or Consumer Electronics Edition (ex Hard Hat Linux), LynuxWorks/BlueCat RT, TimeSys/Linux RTOS Professional or Standard Edition, Lineo-Metrowerks-Motorola/Creation Suite for Linux (ex Embeddix)
    - autres : REDSonic…

LES OFFRES LINUX EMBARQUE

- Les offres de version de Linux embarqué (et Temps Réel) peuvent être rangées dans l'une des 3 catégories suivantes :
  - Les distributions Linux embarqué libres :
    - non TR : µClinux, Embedded Debian Project, PeeWeeLinux, Embedded Linux Workshop (ELW)
    - TR : FSMLabs/RTLinux/free (ex OpenRTLinux GPL), RTAI
    - autres : ADEOS, KURT (TR), Linux-SRT (TR), patchs low latency sur noyau standard (Temps Réel mou), eCOS (TR), ELKS, LEM, LOAF, LRP, Freesco...

LES OFFRES LINUX EMBARQUE

- Voir une liste exhaustive à :
  - http://www.linuxdevices.com/articles/AT9952405558.html
  - http://www.linuxdevices.com/articles/AT8073314981.html

LINUX EMBARQUE COMMERCIAL

- MontaVista/Professional or Carrier Grade or Consumer Electronics Edition :
  - Solution générale (et TR) pour l'embarqué
  - http://www.mvista.com/
  - kit d 'évaluation disponible (preview kit)
- MontaVista Linux Professional Edition
  - This industry-leading comprehensive embedded operating system and cross development environment is our flagship product. It provides a common source and binary platform across a broad range of processor architectures. The Professional Edition includes a modern OS featuring real-time functionality, multi-process and multi-threaded with extensive bundled software components including rich networking.

LINUX EMBARQUE COMMERCIAL

- MontaVista Linux Carrier Grade Edition
  - This innovative product is the industry standard COTS (Commercial-Off-The-Shelf) Carrier Grade Linux platform providing functionality specifically for Telecom and Datacom with high availability, hardening and real-time performance.
- MontaVista Linux Consumer Electronics Edition
  - The latest addition to MontaVista Software's product line is the world's first embedded Linux product targeted at advanced consumer electronics devices. It combines new functionality and tools with rich support of reference platforms to enable the rapid development of a wide range of consumer electronics products.

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de MontaVista/Professional Edition :
- Broad Hardware Support
  - Support for over seventy popular COTS, Evaluation, and Reference boards
  - Support for seven target CPU families with more than 25 CPU variants
- MontaVista Development Environment
  - KDevelop IDE
  - MontaVista Target Configuration Tool
  - MontaVista Library Optimizer Tool
  - Graphical binary and source-level debug
  - Graphical kernel configuration tool
  - Kernel debug (KGDB and hardware debuggers)
  - File system populator

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de MontaVista/Professional Edition :
- Rich Complement of target-based Software Components
- Deployable utilities, libraries, drivers, and other run-time components
- Real-time Support
  - MontaVista Linux Preemptible Kernel
  - MontaVista Linux Real-time Scheduler with up to 1024 levels of priority
- Rich Networking
  - Extensive complement of clients and servers
  - Rich support for the TCP/IP Suite
  - Broad support for routing, security, tunneling
  - cPCI backplane networking

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de MontaVista/Professional Edition :
- File Systems
  - Conventional and Journaling Filesystems
  - Disk, flash and network-based options
- Development Hosts
  - Linux (Red Hat, Mandrake, SuSE)
  - Solaris 7.0, 8.0
  - Windows 2000/XP (command-line and VMWare)

LINUX EMBARQUE COMMERCIAL

- Lineo-Metrowerks-Motorola/Creation Suite for Linux :
  - http://www.metrowerks.com/
  - kit d ’évaluation disponible
- Caractéristiques de Metrowerks Platform Creation Suite for Linux
- Full-featured and integrated embedded developer tool suite targeting multiple processor families for Linux operating system development.
- Target Wizard Configure, build and deploy
- Package Editor Import open source or binary components
- Linux Kernel Import Tool (LKIT) Import a new linux kernel
- Debian Binary Import Tool (DBIT) Extend embedded linux with a full desktop solution
- CodeWarrior IDE Linux hosted IDE environment

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de Metrowerks Platform Creation Suite for Linux
- BSPWerks - Linux Board Support Packages (BSP)
- GPL Compliance Toolset (Analyzes 38 different open source license types)
- Graphical Remote Process Analyzer (GRPA)
- CodeWarrior Development Studio Embedded Linux Edition
- CodeTEST

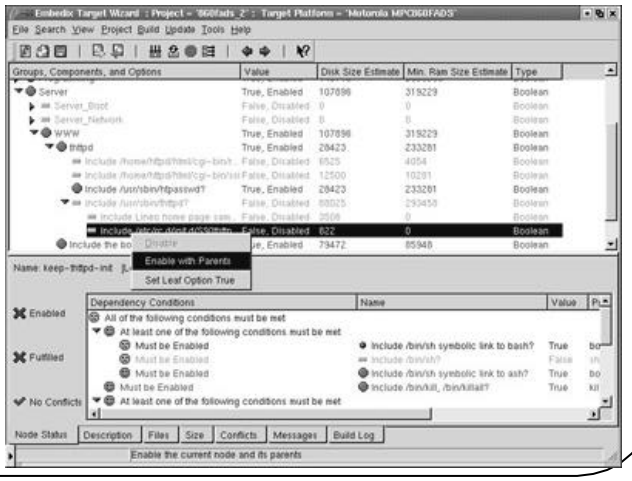
- CodeWarrior Development Studio, Embedded Linux Edition
  - CodeWarrior quality tools for increased productivity in the development of applications designed to run the embedded Linux OS.

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de Metrowerks Platform Creation Suite for Linux
- The Target Wizard Tools are a core element of the Metrowerks Platform creation Suite for Linux OS. These GUI tools manage the configuration, build and deployment of Linux OS components, to fully support your specific product requirements. The Target Wizard tools allow you to build a product completely from the Linux components included within the Board Support Package (BSP). They also provide you with the ability to extend that baseline Linux environment with thousands of downloadable open source or binary applications, GNU tools (e.g. compilers), new Linux kernels or kernel enhancements for improved device support, security, real-time performance, quality of service, or networking. This capability gives you the ability to customize the Linux operating system exactly the way you want for your product and your production hardware.

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de Metrowerks Platform Creation Suite for Linux
- The Target Wizard Tools



LINUX EMBARQUE COMMERCIAL

LYNUXWORKS BlueCat

- LynuxWorks/BlueCat :
  - Solution générale pour l'embarqué
  - http://www.bluecat.com/
  - kit d 'évaluation disponible (cible x86)
- Caractéristiques de LynuxWorks/BlueCat :
- BlueCat Linux is an enhanced implementation of the Linux model, made viable for use in a wide range of embedded systems.Even as BlueCat Linux delivers the flexibility and cost benefits of open-source software, it's uniquely constructed to be a stable, commercial-grade embedded Linux operating system release providing immediate productivity and optimized performance through:

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de LynuxWorks/BlueCat :
- Packages that are tailored to your varying requirements for tools and technical support
- A comprehensive set of tools and board support packages for developing, debugging and deploying Linux into embedded environments
- Based on the Linux 2.4.18 kernel, BlueCat Linux scales from small consumer-type devices to large-scale, multi-CPU systems.

LINUX EMBARQUE COMMERCIAL

- RedHat/Embedded :
  - http://www.redhat.com/embedded/
  - Offres de service, logiciels de développement...

LINUX EMBARQUE COMMERCIAL

- Caractéristiques de RedHat/Embedded :
- OEMs
  - Red Hat provides embedded build-time, run-time, and field-management solutions that enable embedded device manufacturers to quickly develop and add new features to their Internet-ready products.
- Embedded Developers
  - For embedded developers who rely on robust, flexible, and standards-compliant tools, Red Hat offers services for embedded Linux development and the GNUPro developer tools, which cover a wide range of host and target platforms.
- Chip Manufacturers
  - For semiconductor manufacturers who produce leading edge technology and want massive adoption of their processors by device manufacturers, Red Hat provides software development tools, runtime environments, and device management solutions that make their processors and cores compelling to OEMs.

LINUX EMBARQUE OPEN SOURCE

- µClinux :
  - Pour processeur 32 bits **sans MMU**.
  - http://www.uclinux.org

- Caractéristiques de µClinux (voir après pour plus de détails) :

•Lineo's uClinux is the ideal OS for non-MMU microprocessors and high-volume embedded systems featuring posix-4, real-time functions, and TCP/IP. uClinux includes a complete TCP/IP stack supporting Ethernet, PPP and SLIP as well as many wireless protocols. uClinux is perfect for remote sensing, monitoring and control applications. And, because uClinux is an open source product, you will never be stuck on a dead end development path.

LINUX EMBARQUE OPEN SOURCE

- Embedded Debian Project :
  - Outil de génération d'un Linux embarqué (OS+FS).
  - http://www.emdebian.org/

- Caractéristiques de Embedded Debian Project :

• The first effort will attempt to capture the current state of the art by exploring the tools and techniques used by other embedded Linux distributions. The primary product of this effort will be the development of a "Guide to Embedding Debian", a comprehensive guide to getting the most (or least, depending on how you look at it) out of Debian for embedded systems.

LINUX EMBARQUE OPEN SOURCE

- Caractéristiques de Embedded Debian Project :

• The second effort involves determining the best ways to extend Debian's reach into the embedded space. This will involve things such as:

  – Creating a set of build-tools for embedded system developers to easily package just the parts they need. Emdebsys and ipkg seem to be the basis to work from.

  – Specifying new packages tailored specifically for embedded systems

  – Ensuring new embedded packages work across as many platforms as possible

  – Involvement in Linux standardization activities involving embedded Linux.

  –Work with debian proper to integrate embedded requirements into the debian infrastructure.

• EmDebSys a system for the configuration and generation of both a Linux kernel *AND* an operating system (i.e. root filesystem). EmDebSys is being designed to assist embedded Linux developers in configuring and generating small (1 to 10Mb) Linux target systems (ARM, PowerPC, SPARC, Intel x86, Alpha and Motorola 680x0).

LINUX EMBARQUE OPEN SOURCE

• PeeWee Linux :
  – Outil de génération d'un Linux embarqué (OS+FS).
  – http://peeweelinux.com/

• Caractéristiques de PeeWee Linux :

• PeeWeeLinux is an ongoing development effort to provide an environment that makes the configuration and installation of a Linux operating system on an embedded platform as easy and painless as possible. Some of the key features of PeeWeeLinux are:
  – Developed on a RedHat 6.2 platform
  – Packages build and maintained using rpm
  – Packages are customized to minimize memory footprint
  – Ncurses driven graphical configuration and installation tools
  – 2.2.x kernel enhanced for embedded applications
  – USB support, PCMCIA support, XFree86 support

LINUX EMBARQUE OPEN SOURCE

• Caractéristiques de PeeWee Linux :

• The configuration utility is menu driven and allows for complete packages, or a subset of files from packages, to be included in the target system. Target system using syslinux or lilo bootloaders are supported. Targets can consist of root ramdisks, read-only root partitions and conventional single read-write root partitions. Projects can be saved for later use; thus making it very easy to test several different configurations.

• Projet similaire Embedded Linux Workshop ELW :
  – Outil de génération d'un Linux embarqué (OS+FS).
  – http://elw.sourceforge.net/

LINUX EMBARQUE OPEN SOURCE SUR DISQUETTE(S)

- Tom's Boot Root :
  - http://www.toms.net/~toehser/rb/
  - Boot/root rescue/emergency floppy image with more stuff than can fit. Bzip2, 1722Mb formatting, and tight compilation options helped jam a lot on. It is useful for "learn unix on a floppy" as it runs from ramdisk, includes the man-pages for everything, and behaves in a generally predictable way.

- Linux Router Project :
  - http://www.linuxrouter.org/
  - Non maintenu
  - LRP is small enough to fit on a single 1.44MB floppy disk, and makes building and maintaining routers, access servers, thin servers, thin clients, network appliances, and typically embedded systems next to trivial.

LINUX EMBARQUE OPEN SOURCE

- ELKS :
  - http://elks.sourceforge.net/

  - Le projet ELKS (Embeddable Linux Kernel Subset) consiste en une version Linux minimale tenant sur une (une disquette de boot et partition /root sur disque dur) ou deux disquettes 3'1/2 (une disquette de boot et une disquette /root). Son intérêt principal est d'utiliser les vieux PC 80286 et inférieurs. ELKS a besoin d'un PC "honnête" pour le crossdéveloppement (génération des disquettes). ELKS étant en version 0.x.x, les outils de développement embarqués sur le PC ELKS sont inexistants. On ne peut donc utiliser que les exécutables générés sur le PC de crossdéveloppement que l'on peut quand même tester sur ce dernier en générant un objet qui va bien. ELKS n'a pas encore intégré les couches réseaux ni l'accès aux ports d'E/S.

LE CHOIX D ’UN LINUX EMBARQUE

• Le choix est à faire en fonction de ses compétences en interne et des TTM à respecter.

• Choisir un linux embarqué commercial est rassurant. Cela a aussi un coût.

PARTIE 3 :
BILAN

LE CHOIX D ’UN LINUX EMBARQUE

Complexité de mise en œuvre maximale

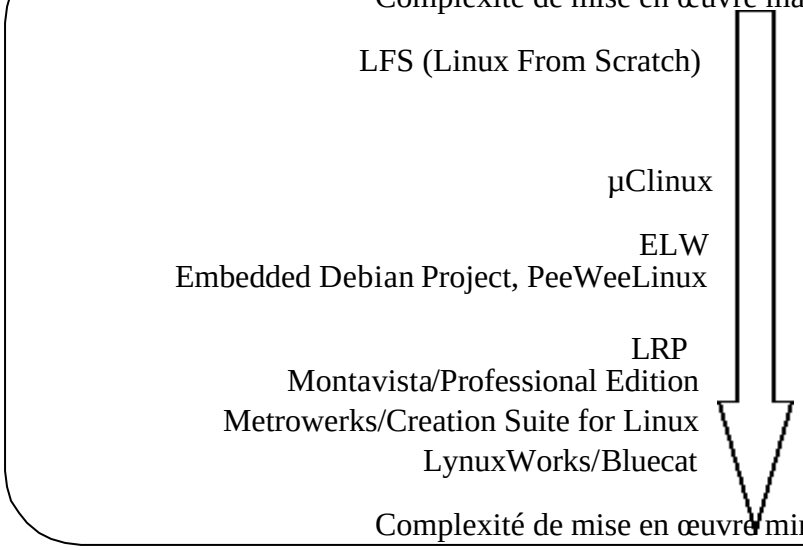
LFS (Linux From Scratch)

µClinux

ELW
Embedded Debian Project, PeeWeeLinux

LRP
Montavista/Professional Edition
Metrowerks/Creation Suite for Linux
LynuxWorks/Bluecat

Complexité de mise en œuvre minimale



PARTIE 4 :
PACKAGES UTILITAIRES
POUR LINUX EMBARQUE

UTILITAIRES LINUX EMBARQUE

- Utilitaires :
  - ae (Anthony's Editor) : éditeur de texte. http://packages.debian.org/unstable/base/ae.html
  - ash : très petit Bourne shell. http://packages.debian.org/unstable/shells/ash.html
  - busybox : petit programme binaire qui fournit la quasi totalité des commandes de base (cat, cp, mv, ls, cd, mount…). http://www.busybox.net/
  - elvis-tiny : un autre éditeur de texts de style vi. http://packages.debian.org/unstable/base/elvis-tiny.html

UTILITAIRES LINUX EMBARQUE

- Utilitaires :
  - iproute : équivalent de ifconfig, route... http://packages.debian.org/unstable/net/iproute.html
  - TinyLogin : suite d'utilitaires permettant de se logguer sur le système et faire la maintenance des utilisateurs. Il utilise les shadow password. http://tinylogin.busybox.net/
  - miniserveur Web boa. http://www.boa.org/
  - miniserveur Web/ftp thttpd. http://www.acme.com/software/thttpd/

UTILITAIRES LINUX EMBARQUE

- Utilitaires :
  - uClibc : une bibliothèque libc à faible empreinte mémoire.
    http://www.uclibc.org
  - udhcp : client/serveur DHCP léger.
    http://udhcp.busybox.net

PARTIE 5 :
ETUDE ET MISE EN ŒUVRE
DE LINUX EMBARQUE : µClinux

PRESENTATION DE µClinux

- µClinux est originellement un dérivé du noyau Linux 2.0 pour les microcontrôleurs sans MMU (*Memory Management Unit*) (février 1998).
- µClinux se prononce "you-see-linux » : µC pour microcontrôleur.
- µClinux a été porté en premier lieu sur le microcontrôleur Motorola MC68328 DragonBall.

- Le premier système sous µClinux a été un ordinateur de poche 3Com PalmPilot using.

PRESENTATION DE µClinux

- La maintenance de µClinux est assurée au départ par Rt-Control, à Toronto, Canada, racheté par Lineo, Inc puis Arcturus Networks !
- Tous les portages µClinux suivant le type du processeur sont sous licence GPL. C ’est un logiciel libre.
- http://www.uclinux.org

PRESENTATION DE µClinux

- µClinux est porté sur les processeurs suivants :
  - Motorola DragonBall (M68EZ328), M68328, M68EN322, ColdFire, QUICC (Quad Integrated Communications Controller).
  - ARM7TDMI.
  - MC68EN302.
  - Axis ETRAX.
  - Intel i960.
  - PRISMA.

µClinux vs Linux

- L'absence de MMU impose quelques limitations d'usage par rapport à l'environnement Linux :
  - La mémoire virtuelle n'existe pas.
  - L'appel système *fork()* n'est pas supporté. Il faudra utiliser alors une implémentation de l'appel système *vfork()* d'UNIX BSD (le processus parent est suspendu jusqu'à ce que le processus fils appelle *exec()* ou *exit()*) .
  - L'appel système *exec()* ne peut pas charger actuellement une image binaire supérieure à 256 Ko.
  - La taille de la pile est fixe pour chaque processus .

CARACTERISTIQUES DE µClinux

- API compatible avec Linux.
- Taille du noyau inférieure à 512 Ko.
- Taille du noyau et commandes Linux inférieures à 900 Ko.
- µClinux possède une pile TCP/IP complète ce qui assure une connectivité IP au système embarqué sous µClinux !
- µClinux supporte les systèmes de fichiers NFS, SMB, ext2, MS-DOS, et FAT16/32 et flash disk JFFS2 (MTD).

CARACTERISTIQUES DE µClinux

- 1997 : µClinux est basé sur le noyau Linux version 2.0.33. Février 1998 : µClinux est mis à jour avec le noyau Linux version 2.0.34.
- 1998 : µClinux est mis à jour avec le noyau Linux version 2.0.38. C 'est la version la plus employée (notée 2.0).
- Fin 2000, µClinux est mis à jour avec le noyau Linux version 2.4.17.
- 2001 : la bibliothèque C standard GNU libc utilisée par µClinux est réécrite et édulcorée pour prendre moins de place en mémoire : bibliothèque uClibc (http://www.uclibc.org/).

PLATEFORME µCsimm POUR µClinux

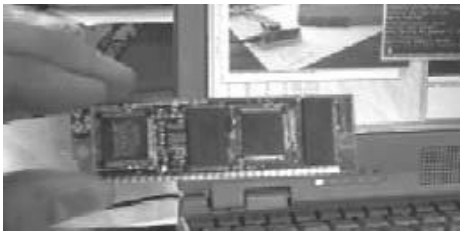
- Les responsables du projet µClinux ont proposé originellement une plateforme matérielle à des fins de tests : le kit µCsimm.
- Ce kit comprend un module qui se présente sous la forme d'une barrette SIMM à 30 broches.
- Le module µCsimm se compose de :
  - Un microcontrôleur DragonBall EZ (68EZ328) basé sur un cœur Motorola 68000 à 16 MHz. Il intègre un contrôleur de DRAM, un port série (interface RS.232), une interface SPI, un contrôleur LCD (résolution QVGA), une sortie Timer et une sortie PWM.

PLATEFORME µCsimm POUR µClinux

- Le module µCsimm se compose de :
  - Jusqu'à 18 E/S parallèles.
  - Une horloge Temps Réel – calendrier.
  - De la mémoire dynamique DRAM de 8 Mo.
  - De la mémoire FLASH de 2 Mo.
  - Une interface Ethernet IEEE 802.3 10BaseT.
- Le module µCsimm est commercialisé sous forme d'un kit de développement comprenant le module µCsimm, une carte d'assemblage / développement, le CD de développement et l'alimentation au prix de 495 USD.

PLATEFORME µCsimm POUR µClinux

- La mémoire FLASH contient un moniteur (*bootloader*) qui permet de télécharger une image via la liaison série en mémoire DRAM et éventuellement la recopier en mémoire FLASH. L'image contient par défaut le noyau µClinux et son systèmes de fichiers avec un petit nombre d'utilitaires (client NFS, miniserveur Web…).



PLATEFORME µCsimm POUR µClinux

- Le kit µCacademix est un kit uCsimm avec un livre d 'exercices réservé à l 'enseignement pour 295 USD.

PLATEFORME µCdimm POUR µClinux

- Le kit µCdimm est une version plus performante du kit originel µCsimm.
- Le module µCdimm se présente sous la forme d'une barrette DIMM à 144 broches (1.7‘’x 2.7‘’).
- Le module µCdimm se compose de :
  - Un microcontrôleur DragonBall VZ (68VZ328) à 33 MHz. Il intègre un contrôleur de DRAM, deux ports série, deux interfaces SPI, un contrôleur LCD (résolution jusqu’à 640x512), une sortie Timer et une sortie PWM.

PLATEFORME µCdimm POUR µClinux

- Le module µCdimm se compose de :
  - Une horloge Temps Réel – calendrier.
  - De la mémoire dynamique DRAM de 8 Mo.
  - De la mémoire FLASH de 2 Mo.
  - Une interface Ethernet IEEE 802.3 10BaseT.
- Les autres éléments du kit sont les même que ceux du kit µCsimm.
- Le kit µCdimm est proposé à 895 USD.
- Une version à base de ColdFire 5272 existe (1295 USD).

PLATEFORME ARM7TDMI POUR µClinux

- Le Le kit ARM7TDMI ressemble fort aux précédents. Il est idéal pour développer des solutions embarquées à base de processeur ARM offrant plus de puissance CPU que les deux kits précédents.
- Le kit est composé de la carte d'évaluation ATMEL EB40-LS à base du processeur ATMEL AT91 ARM7TDMI avec une carte fille pour la connectivité Internet.

PLATEFORME ARM7TDMI POUR µClinux

- L'ensemble forme une plateforme matérielle avec :
  - Un processeur ARM7TDMI avec 128Ko de SRAM interne.
  - De la mémoire statique SRAM de 2 Mo.
  - De la mémoire FLASH de 128 Ko.
  - Deux ports série.
  - Un port JTAG.
  - Une interface Ethernet 10BaseT IEEE 802.3.
- Le kit ARM7TDMI est proposé 495 USD.

PLATEFORME OpenHardware POUR µClinux

- Le projet OpenHardware consiste à proposer des solutions libres de design de cartes électroniques comme il existe des logiciels libres.

- Les schémas électroniques, les nomenclatures, les fichiers GERBER pour la fabrication du circuit imprimé et la notice de montage sont en libre accès. Il est à noter que OpenHardware n'a pas la prétention de proposer des cartes au design optimisé pour une production de masse mais propose plutôt un design pleinement testé comme point de départ à une reprise de CAO…

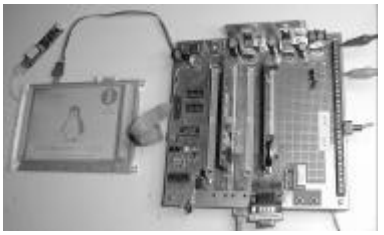
PLATEFORME OpenHardware POUR µClinux

- OpenHardware propose ainsi un design à base du microcontrôleur DragonBall EZ (68EZ328) supportant µClinux.

- Il se compose :
  - D'une carte mère avec 4 sockets SIMM 72 broches dont 2 sont libres d'utilisation avec un connecteur RJ45 pour l'accès Ethernet IEEE 802.3 10BaseT, une interface pour écran LCD et une interface " *Touch Panel* ".
  - D'un module SIMM 72 broches comportant :
  - Le processeur DragonBall EZ.
  - 8 Mo de DRAM.

PLATEFORME OpenHardware POUR µClinux

- Il se compose :
  - 2, 4 ou 8 Mo de mémoire FLASH.
  - Horloge Temps Réel avec sauvegarde par pile bouton.
  - 9 E/S parallèles.
  - D'un module SIMM 72 broches comportant le contrôleur Ethernet.



PLATEFORMES ColdFire POUR µClinux

- Le processeur ColdFire est le dernier né de Motorola dont la première version est sortie en 1994.

- Le processeur ColdFire est le digne successeur du célèbre processeur Motorola 68000 dont la production s'est arrêtée avec le microprocesseur 68060 il y a quelques années.

- Il reprend le jeu d'instructions du microprocesseur 68020 où l'on a supprimé les instructions qui ne servaient guère ainsi que certains modes d'adressage.

- Le processeur ColdFire reprend une architecture RISC.

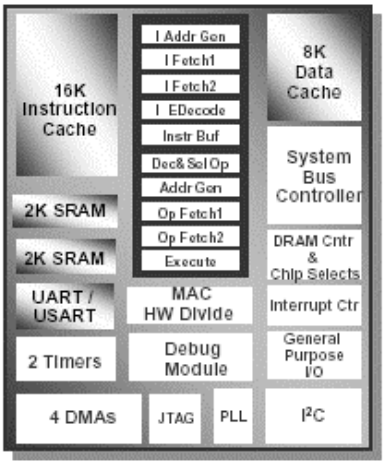
PLATEFORMES ColdFire POUR µClinux

- Suivant les versions de processeur ColdFire, Motorola a rajouté des instructions de type MAC (*Multiply and ACCumulate*) que l'on retrouve dans les processeurs de traitement du signal (*Digital Signal Processing*).

- Il a aussi intégré des périphériques ; ce qui confère au processeur ColdFire les propriétés d'un microcontrôleur / DSP. Le marché visé par Motorola est clairement celui des télécommunications où l'on retrouve ce genre de besoins.

PLATEFORMES ColdFire POUR µClinux

- Processeur ColdFire 5407 :



PLATEFORMES ColdFire POUR µClinux

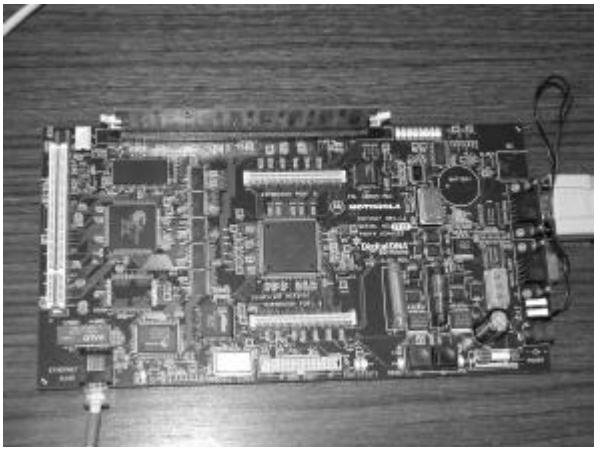
- µClinux est porté sur la cartes ColdFire suivantes :
  - Arnewsh Inc SBC5206 development board
  - Arnewsh Inc SBC5307 development board
  - Motorola M5206eLITE development board
  - Motorola M5206C3 development board
  - Motorola M5249C3 development board
  - Motorola M5272C3 development board
  - Motorola M5307C3 development board
  - Motorola M5407C3 development board
  - SnapGear (also NETtel and SecureEdge) router platforms
  - Moreton Bay eLIA 5307 based development platform
  - Netburner CFV2-40 development board

PLATEFORMES ColdFire POUR µClinux

- La mise en œuvre de µClinux sera faite sur carte ColdFire Motorola EVB M5407C3.

- La carte d'évaluation M5407C3 possède :
  - Un processeur MCF5407 à 150 MHz (257 Dhrystone MIPS).
  - 32 Mo de mémoire SDRAM.
  - 2 Mo de mémoire FLASH dont 0,3 Mo occupé par le moniteur de la carte.
  - 2 ports série.
  - Une liaison Ethernet IEEE 802.3 10BaseT.
  - Un port de debug BDM (*Background Debugger Module*).
  - Un slot PCI (version 2.1).

PLATEFORMES ColdFire POUR µClinux



MISE EN ŒUVRE DE µClinux

- Installation du cross compilateur C pour ColdFire (fichiers binaires).
- En étant root :

```
# cd /
# tar xvzf m68k-elf-20020410.tar.gz
```

- Les outils sous installés sous /usr/local/bin.
- Les bibliothèques sont installées sous /usr/local/lib.
- les fichiers .h sont installés sous /usr/local/include.

MISE EN ŒUVRE DE µClinux

- En étant simple utilisateur, dans son home directory, installer le package µClinux :

  ```
  % cd ~
  % tar xvzf uClinux-dist-XXXXXXXX.tar.gz
  ```

- Un répertoire ~/ uClinux-dist est créé. Se placer dans ce répertoire :

  ```
  % cd uClinux-dist
  ```


MISE EN ŒUVRE DE µClinux

- Configurer le noyau µClinux comme sous Linux (choix de la carte, des applications à embarquer…) :

  ```
  % make xconfig
  ```

- Construire les dépendances :

  ```
  % make dep
  ```

- Construire l ’image à télécharger dans la carte :

  ```
  % make
  ```

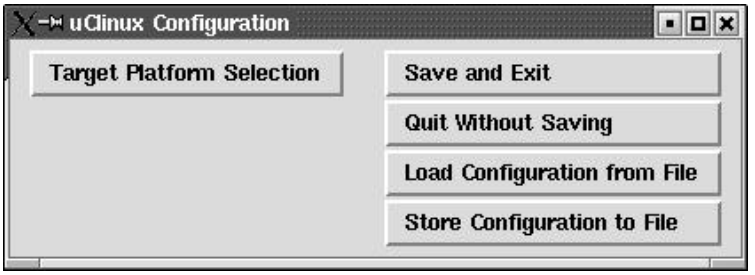
- Télécharger l ’image dans la carte depuis le moniteur (par le réseau, par la liaison série) et lancer le noyau µClinux.

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

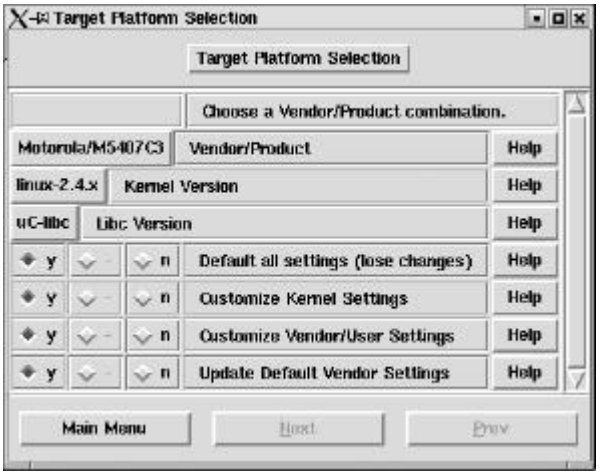
- Durant la phase % make xconfig, il convient de choisir les bons paramètres correspondant à la carte Motorola.
- On laissera les autres paramètres avec leurs valeurs par défaut.

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Choix de la plateforme ColdFire : M5407C3.
- Noyau 2.4.x, bibliothèque uClibc :

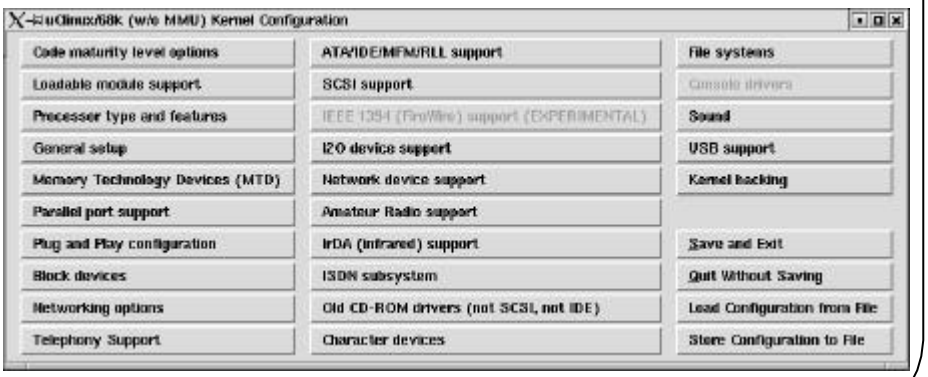


MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3



MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

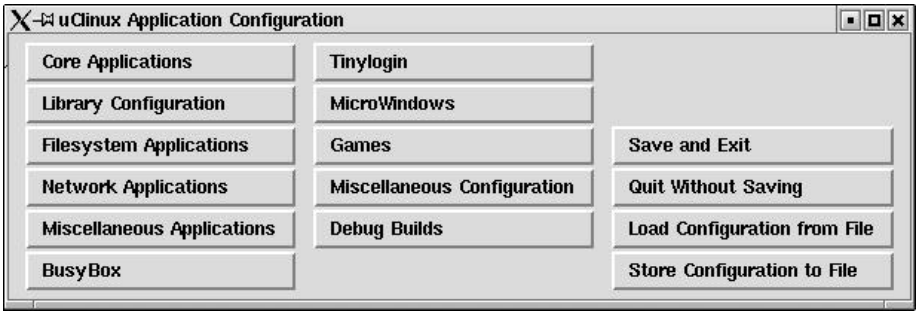
- Configuration du noyau :



MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

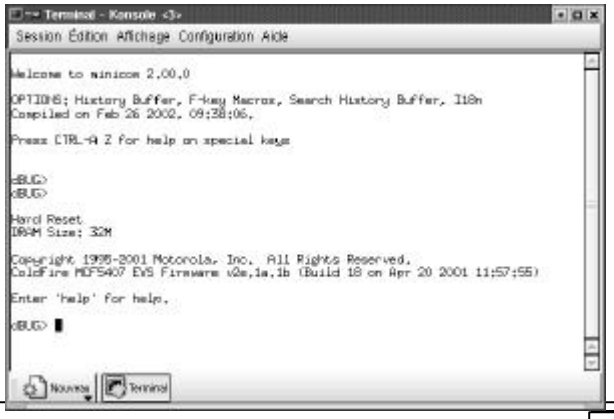
- Choix des applications standards et « *userland* » :



MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- La carte ColdFire possède un moniteur embarqué pour des opérations élémentaires accessibles depuis le port série :



MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

MNEMONIC	SYNTAX	DESCRIPTION
ASM	asm <<addr> stmt>	Assemble
BC	bc addr1 addr2 length	Block Compare
BF	bf <width> begin end data <inc>	Block Fill
BM	bm begin end dest	Block Move
BR	br addr <-r> <-c count> <-t trigger>	Breakpoint
BS	bs <width> begin end data	Block Search
DC	dc value	Data Convert
DI	di<addr>	Disassemble
DL	dl <offset>	Download Serial
DN	dn <-c> <-e> <-i> <-s <-o offset>> <filename>	Download Network
GO	go <addr>	Execute
GT	gt addr	Execute To
HELP	help <command>	Help
IRD	ird <module.register>	Internal Register Display
IRM	irm module.register data	Internal Register Modify
LR	lr<width> addr	Loop Read
LW	lw<width> addr data	Loop Write
MD	md<width> <begin> <end>	Memory Display
MM	mm<width> addr <data>	Memory Modify
MMAP	mmap	Memory Map Display
RD	rd <reg>	Register Display
RM	rm reg data	Register Modify
RESET	reset	Reset
SD	sd	Stack Dump
SET	set <option value>	Set Configurations
SHOW	show <option>	Show Configurations
STEP	step	Step (Over)
SYMBOL	symbol <symb> <-a symb value> <-r symb> <-C|s>	Symbol Management
TRACE	trace <num>	Trace (Into)
UPDBUG	updbug	Update dBUG
UPUSER	upuser <bytes>	Update User Flash
VERSION	version	Show Version

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Le fichier image (fichier image.bin) du noyau µClinux est recopié sous /tftpboot. Ce fichier sera ensuite téléchargé par le réseau depuis le moniteur de la carte ColdFire par le protocole TFTP.
- Il convient donc d'avoir un serveur TFTP actif sur le PC hôte.
- Il est toujours possible de télécharger l'image µClinux par la liaison série (à 19200 b/s). Celle-ci doit au préalable être transformée en un fichier au format Motorola S-Record :

  % m68k-elf-objcopy --input-target=binary --output-target=srec \
  image.bin image.srec

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Téléchargement du fichier image image.bin par TFTP :

```
[root@kiwi1 uclinux]# kermit
Connecting to /dev/ttyS0, speed 19200. The escape character is Ctrl-\
   (ASCII 28, FS) Type the escape character followed by C to get back,
   or followed by ? to see other options.
Hard Reset
DRAM Size: 32M

Copyright 1995-2001 Motorola, Inc.  All Rights Reserved.
ColdFire MCF5407 EVS Firmware v2e.1a.1b (Build 18 on Apr 20 2001
   11:57:55)
```


MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Téléchargement du fichier image image.bin par TFTP :

```
dBUG> show
        base: 16
        baud: 19200
      server: 192.9.203.1
      client: 192.9.203.16
     gateway: 255.255.255.255
     netmask: 255.255.255.0
    filename: image.bin
    filetype: Image
         mac: 00:00:00:00:00:06
```


MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Téléchargement du fichier image image.bin par TFTP :

```
dBUG> dn image.bin
Address:  0x00020000
Eth Mac Addr is 00:00:00:00:00:06
Downloading Image 'image.bin' from 192.9.203.1
TFTP download successful
Read 1596184 bytes (3118 blocks)
```


MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Boot du noyau µClinux :

```
dBUG> go 20000
Linux version 2.4.17-uc0 (uclinux@kiwi1) (gcc version 2.95.3 20010315

uClinux/COLDFIRE(m5407)
COLDFIRE port done by Greg Ungerer, gerg@snapgear.com
Flat model support (C) 1998,1999 Kenneth Albanowski, D. Jeff Dionne
On node 0 totalpages: 8192
zone(0): 0 pages.
zone(1): 8192 pages.
zone(2): 0 pages.
Kernel command line:
Calibrating delay loop... 149.50 BogoMIPS
Memory available: 30424k/32768k RAM, 0k/0k ROM (581k kernel code, 197k
   data)
```

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Boot du noyau µClinux :

```
Command: cat /etc/motd
Welcome to
          ____ _  _
         /  __| ||_|
    _   _| |  | | _ ____  _   _  _  _
   | | | | |  | || |  _ \| | | |\ \/ /
   | |_| | |__| || | | | | |_| |/    \
   |  ___\____|_||_|_| |_|\____|\_/\_/
   | |
   |_|
For further information check:
http://www.uclinux.org/
Execution Finished, Exiting
Sash command shell (version 1.1.1)
```


MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Configuration réseau et tests :

```
/> ifconfig eth0 192.9.203.16
/> ping 192.9.203.1
PING 192.9.203.1 (192.9.203.1): 56 data bytes
64 bytes from 192.9.203.1: icmp_seq=0 ttl=255 time=10.0 ms
64 bytes from 192.9.203.1: icmp_seq=1 ttl=255 time=0.0 ms
64 bytes from 192.9.203.1: icmp_seq=2 ttl=255 time=0.0 ms
64 bytes from 192.9.203.1: icmp_seq=3 ttl=255 time=0.0 ms

--- 192.9.203.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.0/2.5/10.0 ms
```

MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Utilisation :

```
/> ps
  PID PORT STAT SIZE SHARED %CPU COMMAND
    1      S     21K     0K  0.8 init
    2      S      0K     0K  0.0 keventd
    3      R      0K     0K  0.0 ksoftirqd_CPU0
    4      S      0K     0K  0.0 kswapd
    5      S      0K     0K  0.0 bdflush
    6      S      0K     0K  0.0 kupdated
   11      S     21K     0K  0.4 dhcpcd -p -a eth0
   12   S0 R     24K     0K  0.0 /bin/sh
   13      S     14K     0K  0.0 /bin/inetd
   14      S     37K     0K  0.0 /bin/boa
```


MISE EN ŒUVRE DE µClinux SUR CARTE EVB5407C3

- Utilisation :

```
/> route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use
   Iface
192.9.203.0     *               255.255.255.0   U     0      0        0
   eth0
127.0.0.0       *               255.0.0.0       U     0      0        0
   lo
```

CONFIGURATION DE µClinux

- Configuration réseau :

```
# ifconfig eth0 192.168.0.232
# route add -net 192.168.0.0 netmask 255.255.255.0
```

Where the units IP address is set to 192.168.0.232 on a network address of 192.168.0.0, and setting the netmask to be 255.255.255.0.

You may also want to set up a gateway route if you have a local router :

```
# route add default gw 192.168.0.1
```


CONFIGURATION DE µClinux

- Ports série :

The mcfserial.c driver supports both internal UARTs of the ColdFire processors. Baud rates up to at least 115200 can be generated. You should also be able to use any parity, data bit and stop bit combinations.

There is a simple callout program for manual serial port use, called tip. It is similar to cu or the usual tip, only cut down a little. You can talk directly to attached modems with something like:

```
# tip -s 57600 /dev/cua1
```


CONFIGURATION DE µClinux

- NFS :

NFS is configured into the kernel by default on most platforms. If not you will need to configure the Linux kernel build to include it. To mount NFS filesystems from the network, just do something like:

```
# /bin/mount -n -t nfs X.X.X.X:/YYYY /mnt
```

Note that you must use a specific path to mount (eg /bin/mount) since the sash shell has a built in mount (but it is not NFS capable).

It is really nice being able to mount your development systems files, and run new binaries directly after compiling!

CONFIGURATION DE µClinux

- SMB Windows :

The SMB filesystem is not configured into the kernel by default, so you will need to re-configure the Linux kernel to include it. To mount a file share then you would use something like:

```
# /bin/smbmount //MACHINE/SHARENAME / mnt -n -c
MACHINE -I X.X.X.X -U username -P passwd
```

where MACHINE is the system that you want to mount from, SHARENAME is the name of an exported share from the machine, X.X.X.X is the IP address of the MACHINE system, and username/passwd is an authorized network user.

CONFIGURATION DE µClinux

- Serveurs WWW :

There is currently 3 different web servers ported to uClinux/ColdFire.

- The simplest is the uClinux web server, httpd. It can only deliver static pages.
- The Boa web server is a light weight nearly full featured web server. It has cgi-bin and authentication support. It is also single tasking - not spawning off multiple processes to handle simultaneous requests. Boa's memory footprint is extremely small (about 85k when running). I highly recommend using this one.
- The thttpd web server has also been ported to uClinux/ColdFire. It also has cgi-bin and authentication support. It does however spawn off multiple processes to support multiple connections, and it has a much larger memory foot print (about 150k when running).

CONFIGURATION DE µClinux

- Debug du noyau par GDB :

- Il faut utiliser le câble BDM *(Background Debugger Module)* fourni avec la carte à connecter sur la carte ColdFire et le port parallèle du PC hôte.

- Sur le PC hôte, on a besoin :
  - Du driver Linux pour piloter le module BDM.
  - La version patchée de GDB (Gnu DeBugger) pour le support du BDM.
  - Le fichier d'initialisation de GDB pour le processeur ColdFire 5407.

CONFIGURATION DE µClinux

- Debug d 'une application par GDB :

A gdbserver port exists now for debugging user applications over the network. You will need a cross gdb for debugging with this setup.

To debug an application remotely:

1. On uClinux/ColdFire "gdbserver :3000 app"
2. On host system "gdb app.gdb"
3. At gdb command prompt enter "target remote IP:3000"

CONFIGURATION DE µClinux

- Debug d 'une application par GDB :

This procedure uses port TCP port 3000, you can use any un-used port you like. IP is the IP address of the uClinux/ColdFire hardware.

Providing the application was compiled with debugging enabled (gcc -g option) you will get full symbolic and source level debugging. This is enabled automatically if the kernel "Full Symbolic/Source level debuging" flag is enabled in the "Kernel Hacking" section of the config.

CONFIGURATION DE µClinux

- Support du bus PCI :

Some ColdFire boards have PCI buses on them. uClinux/ColdFire supports the PCI buses on both these boards, simply enable the PCI bus support option when configuring your kernel build.

The kernel PCI support will scan the PCI bus at system startup, and will find any PCI devices present. The kernel will also assign memory and interrupt resources to the found devices.

To find out what the system has found and assigned on the PCI bus use the /proc/pci device. Simply do :

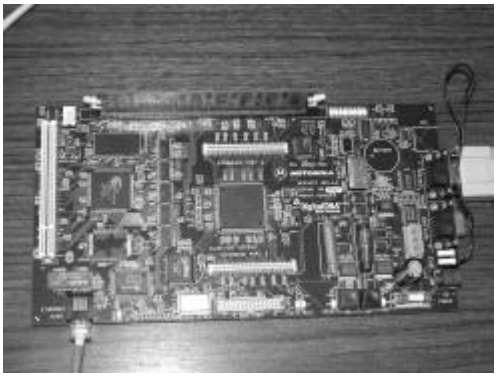
```
# cat /proc/pci
```


EXEMPLE ENSEIRB

- Intégration d 'un serveur web et d 'un agent SNMP sur carte ColdFire sous µClinux pour un contrôle par Internet pour le télécontrôle et télémaintenance d 'un système électronique.

EXEMPLE ENSEIRB

- Implémentation matérielle :
  - Carte Motorola ColdFire M5407C3 pour développement et tests.

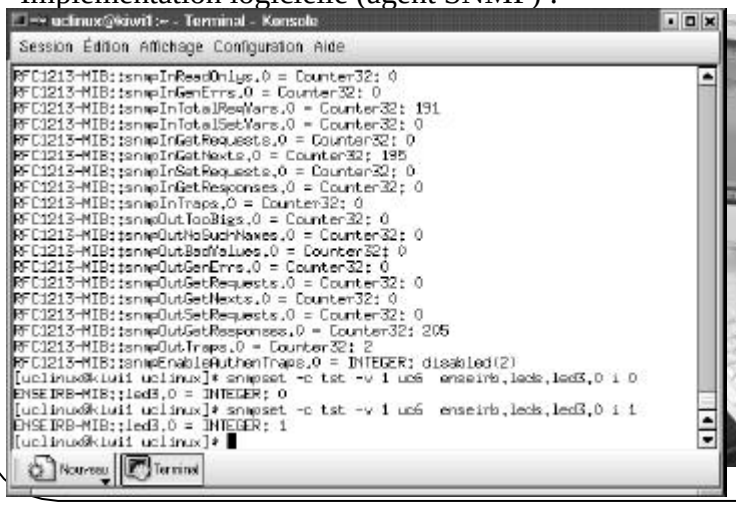


EXEMPLE ENSEIRB

- Implémentation logicielle :
  - Serveur web boa. Programmes CGI (écrits en langage C) pour piloter les périphériques de la carte (leds à des fins de tests).
  - Extension de l ’agent SNMP NET-SNMP pour piloter les périphériques de la carte (leds à des fins de tests).

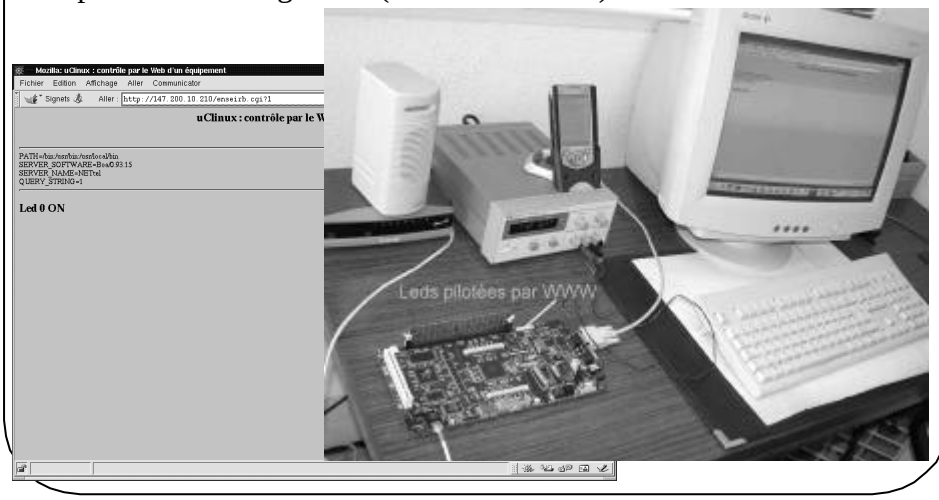
EXEMPLE ENSEIRB

- Implémentation logicielle (agent SNMP) :



EXEMPLE ENSEIRB

- Implémentation logicielle (serveur web boa) :



RESSOURCES µClinux

- Le site officiel : http://www.uclinux.org
- Le site des utilisateurs ucdot.org : http://www.ucdot.org/
- Le site « the uclinux directory » : http://home.at/uclinux/
- Le portage µClinux sur ColdFire :
  http://www.uclinux.org/ports/coldfire/
- La bibliothèque uClibc : http://www.uclibc.org/
- La page de l 'auteur :
  http://www.enseirb.fr/~kadionik/embedded/embedded.html

µClinux  uCdot Linux coldfire

RESSOURCES µClinux

- Plus d 'informations :
  - http://www.enseirb.fr/~kadionik/embedded/embedded.html
  - Linux Magazine. Le projet µClinux. P. Kadionik. Février 2002.

    http://www.enseirb.fr/~kadionik/embedded/uclinux/
  - Linux Magazine. Administrez facilement votre réseau. P. Kadionik. Octobre 2002.

    http://www.enseirb.fr/~kadionik/embedded/snmp/

CHAPITRE 7 :
LE TEMPS REEL SOUS LINUX
INTRODUCTION AU TEMPS REEL

PARTIE 1 :
INTRODUCTION

TEMPS REEL MOU

- Un système d ’exploitation est dit Temps Réel (dur) s ’il est capable de répondre à des sollicitations ou événements (internes ou externes) dans un temps maximum.
- On parle de Temps Réel mou (*Soft Real Time*) quand les événements traités trop tardivement ou perdus sont sans conséquence catastrophique pour la bonne marche du système. On ne garantit qu ’un pourcentage **moyen** d ’utilisation du temps CPU.
- Les systèmes à contraintes *souples* ou *molles* (*soft real time*) acceptent des variations dans le traitement des données de l'ordre de la demi-seconde (ou 500 ms) ou la seconde.

TEMPS REEL MOU

- On peut citer l'exemple des systèmes multimédia : si quelques images ne sont pas affichées, cela ne met pas en péril le fonctionnement correct de l'ensemble du système.
- Ces systèmes se rapprochent fortement des systèmes d'exploitation classiques à temps partagé.
- Ils garantissent un temps moyen d'exécution pour chaque tâche (un débit, une Bande Passante).
- On a ici une répartition **égalitaire** du temps CPU entre processus.

TEMPS REEL DUR

- On parle de Temps Réel dur (Hard *Real Time*) quand les événements traités trop tardivement ou perdus provoquent des conséquences catastrophiques pour la bonne marche du système (perte d ’informations cruciales, plantage…).
- Les systèmes à contraintes *dures* (*hard real time*) ne tolèrent qu’une gestion stricte du temps est nécessaire afin de conserver l'intégrité du service rendu.
- On citera comme exemples les contrôles de processus industriels sensibles comme la régulation des centrales nucléaires ou les systèmes embarqués utilisés dans l'aéronautique.

TEMPS REEL DUR

- Ces systèmes garantissent un temps maximum d'exécution pour chaque tâche.
- On a ici une répartition **totalitaire** du temps CPU entre tâches.
- On peut dire qu'un système temps réel doit être prévisible (*predictible* en anglais*)*, les contraintes temporelles pouvant s'échelonner entre quelques micro-secondes (µs) et quelques secondes.

TEMPS REEL DUR

- Les systèmes à contraintes dures doivent répondre à trois critères fondamentaux :
  - Le déterminisme *logique* : les mêmes entrées appliquées au système doivent produire les mêmes effets.
  - Le déterminisme *temporel* : un tâche donnée doit obligatoirement être exécutée dans les délais impartis, on parle d'*échéance*.
  - La *fiabilité* : le système doit être disponible. Cette contrainte est très forte dans le cas d'un système embarqué car les interventions d'un opérateur sont très difficiles voire même impossibles. Cette contrainte est indépendante de la notion de temps réel mais la fiabilité du système sera d'autant plus mise à l'épreuve dans le cas de contraintes dures.

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - Le noyau Linux possède de longues sections de code où tous les événements extérieurs sont masqués (non interruptible).

    ET
  - Le noyau Linux n ’est pas préemptible durant toute l ’exécution d ’un appel système (structure monolithique) par un processus et ne le redevient qu ’en retour d ’appel système (mode user).
  - Le noyau Linux n ’est pas préemptible durant l e service d ’une interruption (ISR). La routine ISR acquitte l ’interruption puis programme un « Bottom Half » (BH) pour le traitement des données. Le BH est exécuté de façon non préemptif immédiatement avant de retourner en mode user.

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - En cas d ’occurrence d ’une interruption durant l ’exécution d ’un appel système en mode noyau, le BH programmé par l ’ISR ‘ et evéentuellement les autres BH des autres ISR) ne sera exécuté qu ’à la fin de l ’exécution complète de l ’appel système d ’où un temps de latence important et non borné fatal à un système Temps Réel !

LINUX ET LE TEMPS REEL

- Linux n ’est pas un système d ’exploitation Temps Réel (dur) car :
  - L ’ordonnanceur de Linux essaye d ’attribuer de façon équitable le CPU à l ’ensemble des processus (ordonnancement de type *old aging* mise en œuvre pour favoriser l ’accès CPU aux processus récents). C ’est une approche égalitaire. Un ordonnanceur Temps Réel donnera toujours la main à la tâche de plus forte priorité prête. C ’est ici un approche plus totalitaire.

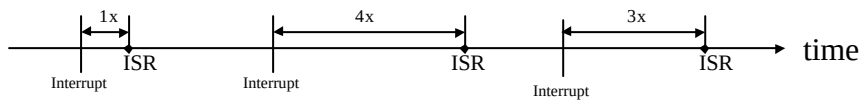
LINUX ET LE TEMPS REEL

- Le noyau Linux standard peut être considéré comme Temps Réel (mou) par définition si l 'on travaille avec une réactivité de l 'ordre de la centaine de ms ou plus.
- Il existe des solutions Linux Temps Réel dur pour une réactivité de quelques dizaines de µs…
- Il existe des solutions Linux Temps Réel mou par application de patchs dits préemptifs sur un noyau Linux standard pour une réactivité de quelques centaines de µs...

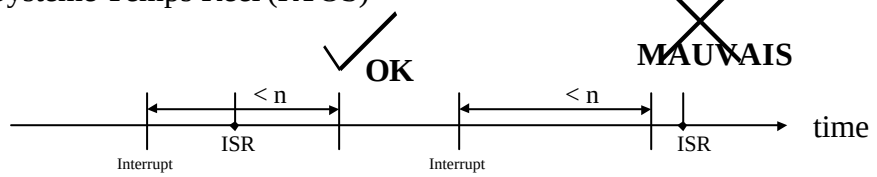
LINUX ET LE TEMPS REEL

- Traitement des interruptions et ISR (*Interrupt Sub Routine*) :

Linux



Système Temps Réel (RTOS)

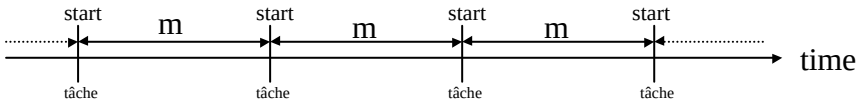


n: délai maximum

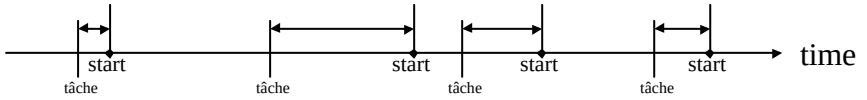
LINUX ET LE TEMPS REEL

- Traitement des tâches périodiques :

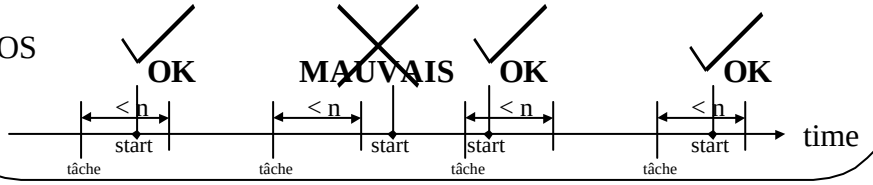
Idéalement



Linux



RTOS



UNE PETITE EXPERIENCE

- Génération d ’un signal périodique sur une broche du port parallèle.
- Le signal généré sur la broche 2 (bit D0) du port parallèle est théoriquement un signal périodique carré de demi-période T/2 de 50 ms.
- On observe à l'oscilloscope le signal suivant sur un système non chargé (AMD Athlon 1500+).

UNE PETITE EXPERIENCE

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <asm/io.h>

#define LPT 0x378

int ioperm();

int main(int argc, char **argv)
{
   setuid(0);
   if (ioperm(LPT, 1, 1) < 0) {
    perror("ioperm()");
    exit(-1);
   }
```

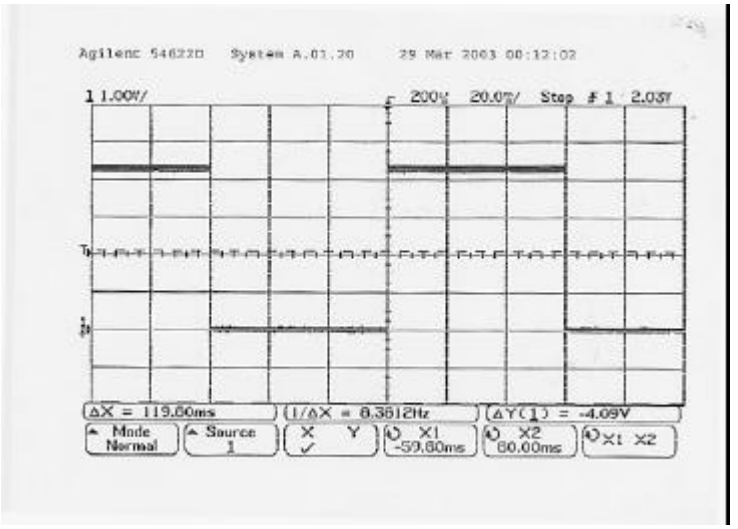

UNE PETITE EXPERIENCE

```
   while(1){
    outb(0x01, LPT);
    usleep(50000);

    outb(0x00, LPT);
    usleep(50000);
   }
   return(0);
}
```

Programme `square` (fichier C square.c)

UNE PETITE EXPERIENCE

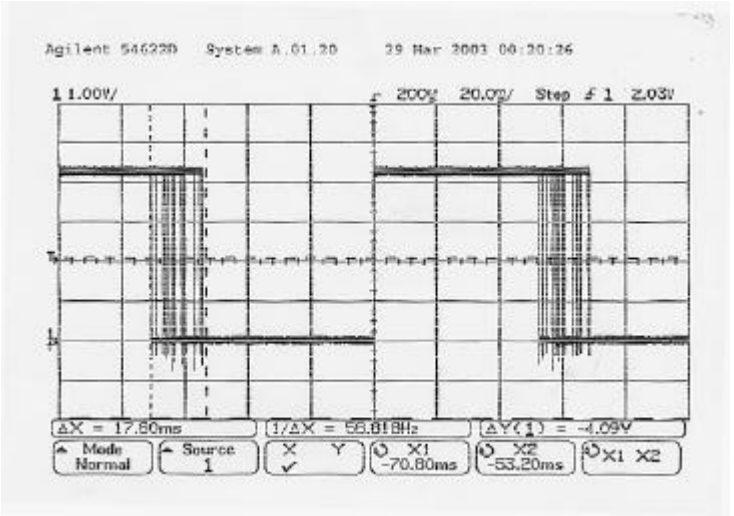


Génération d'un signal carré sous Linux non chargé

UNE PETITE EXPERIENCE

- On remarque que l'on n'a pas une période de 100 ms mais de 119,6 ms dû au temps supplémentaire d'exécution des appels système.
- Dès que l'on stresse le système (écriture répétitive sur disque d'un fichier de 50 Mo), on observe le signal suivant :

UNE PETITE EXPERIENCE



Génération d'un signal carré sous Linux chargé

UNE PETITE EXPERIENCE

- On observe maintenant une gigue (*jitter*) sur le signal généré. La gigue maximale sur la durée de l'expérience est de 17,6 ms.
- La forme du signal varie maintenant au cours du temps, n'est pas de forme carrée mais rectangulaire. LINUX n'est donc plus capable de générer correctement ce signal.
- Il faut noter aussi que le front montant sur la figure précédente apparaît sans gigue car il a servi comme front de synchronisation de l'oscilloscope. La gigue observée est donc à voir comme la contribution de la gigue sur front montant et sur front descendant.
- Si l'on diminue la valeur de la demi-période, la gigue devient aussi importante que cette dernière et dans ce cas, Linux ne génère plus aucun signal !

LINUX ET LES NORMES POSIX

- La complexité des systèmes et l'interopérabilité omniprésente nécessitent une standardisation de plus en plus grande tant au niveau des protocoles utilisés que du code source des applications. Même si elle n'est pas obligatoire, l'utilisation de systèmes conformes à *POSIX* est de plus en plus fréquente.

- POSIX est l'acronyme de *Portable Operating System Interface* ou interface portable pour les systèmes d'exploitation.

- Cette norme a été développée par l'IEEE (*Institute of Electrical and Electronic Engineering*) et standardisée par l'ANSI (*American National Standards Institute*) et l'ISO (*International Standards Organisation*).

LINUX ET LES NORMES POSIX

- Le but de POSIX est d'obtenir la portabilité des logiciels au niveau de leur code source.

- Un programme qui est destiné à un système d'exploitation qui respecte POSIX doit pouvoir être adapté à moindre frais sous n'importe quel autre système POSIX. En théorie, le portage d'une application d'un système POSIX vers un autre doit se résumer à une compilation des sources du programme.
- POSIX a initialement été mis en place pour les systèmes de type UNIX mais d'autres systèmes d'exploitation comme *Windows NT* sont aujourd'hui conformes à POSIX.

LINUX ET LES NORMES POSIX

- Le standard POSIX est divisé en plusieurs sous-standards dont les principaux sont les suivants :
  - IEEE 1003.1-1990 : POSIX Partie 1 : Interface de programmation (API) système. Définition d'interfaces de programmation standards pour les systèmes de type UNIX, connu également sous l'appellation ISO 9945-1. Ce standard contient la définition de ces fonctions en langage C.
  - IEEE 1003.2-1992 : Interface applicative pour le *shell* et applications annexes. Définit les fonctionnalités du shell et commandes annexes pour les systèmes de type UNIX.
  - IEEE 1003.1b-1993 : Interface de programmation (API) temps réel. Ajout du support de programmation temps réel au standard précédent. On parle également de POSIX.4.
  - IEEE 1003.1c-1995 : Interface de programmation (API) pour le multithreading.

LINUX ET LA NORME POSIX 1003.1b

- La norme POSIX 1003.1b définit la notion de Temps Réel pour un système d 'exploitation à des fins de normalisation :

  "Realtime in operating systems: the ability of the operating system to provide a required level of service in a bounded response time "

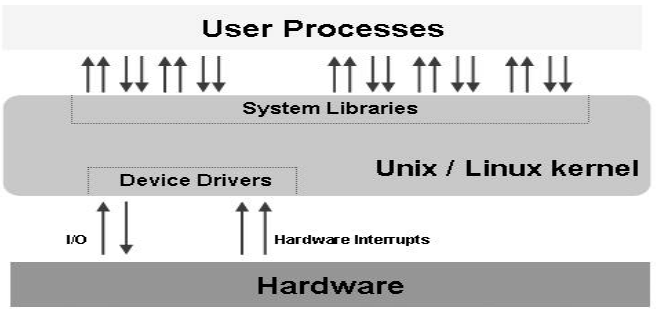
- Cela implique pour le système d'exploitation :
  - D'avoir un ordonnancement des tâches préemptif (avec priorité).
  - De coller en mémoire les pages virtuelles.
  - De supporter des signaux Temps Réel.
  - D'avoir des mécanismes de communication inter-processus IPC (*Inter Processus Communication*) performants.
  - D'avoir des timers Temps Réel.

LINUX ET LA NORME POSIX 1003.1b

- Linux ne supporte que partiellement la norme POSIX 1003.1b et ne peut être considéré en l 'état comme un système Temps Réel.
  - Appels système : *mlock(), setsched().*
- Il convient donc de modifier Linux pour le rendre Temps Réel et conforme à la norme POSIX 1003.1b.

EXTENSION TEMPS REEL POUR LINUX

- Implémentation du noyau Linux standard :
  - Pas de support du Temps Réel.
  - Séparation entre le matériel et les processus Linux.
  - ...



Linux standard

EXTENSION TEMPS REEL POUR LINUX

- Solution 1 pour une extension Temps Réel mou de Linux :
  - Modification du noyau Linux par application de patchs pour améliorer les contraintes Temps Réel : dévalider les interruptions le moins longtemps possible, appeler l 'ordonnanceur le plus souvent possible (fonction *schedule()* du noyau), en retour d 'interruption par exemple.
  - Les patchs dits " préemptifs " permettant d'améliorer le comportement du noyau LINUX en réduisant les temps de latence de ce dernier. Ces modifications ne transforment pas LINUX en noyau temps réel dur mais permettent d'obtenir des résultats satisfaisants dans le cas de contraintes temps réel molles.

EXTENSION TEMPS REEL POUR LINUX

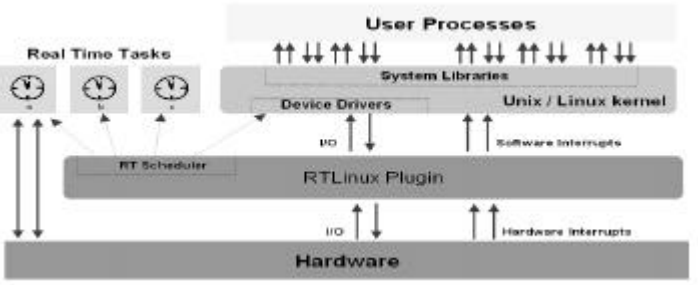
- Solution 1 pour une extension Temps Réel mou de Linux :
  - Cette technologie est disponible auprès de différents projets open source (projet KURT, Linux-SRT) et elle est également supportée commercialement par divers éditeurs spécialisés comme MontaVista, TimeSys.
  - La notion de noyau préemptif est intégrée dans le noyau de développement 2.5 (et suivant).

EXTENSION TEMPS REEL POUR LINUX

- Solution 2 pour une extension Temps Réel dur de Linux :
  - Ajout d ’un deuxième ordonnanceur TR de tâches et considérer le noyau Linux et ses processus comme tâche de fond. Plus difficile que la première solution.
  - Cette technique permet de mettre en place des systèmes temps réel durs.
  - Utilisé dans les projets RTLinux et RTAI par exemple.
  - On ne peut pas considérer Linux et son extension TR dans ce cas comme un véritable Noyau TR monolithique (pour les puristes du TR) et enfreint la « logique Linux » et la cohérence de l ’API Linux (pour les puristes Linux)...

EXTENSION TEMPS REEL POUR LINUX

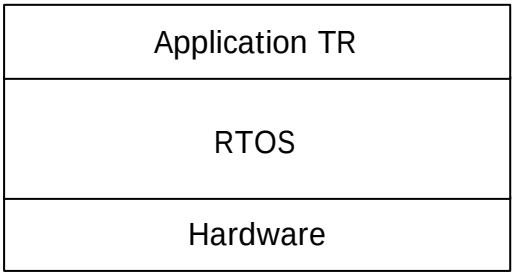
- Solution 2 pour une extension Temps Réel dur de Linux :
  - Ajout d ’une couche d ’abstraction entre le matériel et le noyau Linux.
  - Définition de tâches Temps Réel.
  - Pas de séparation entre le matériel et les tâches Temps Réel.



Linux et un extension TR de type RTLinux

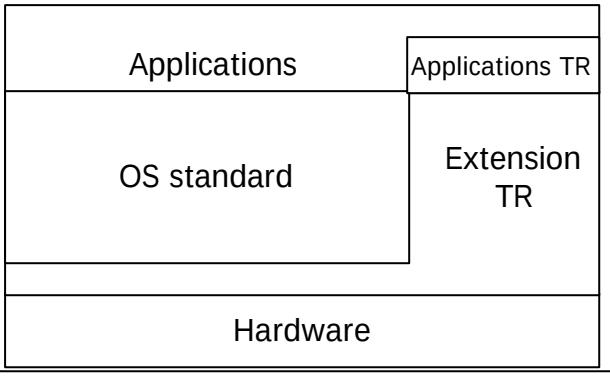
VERITABLE RTOS

- Avantages : simplicité, monolithique, fait pour le TR, petit overhead.
- Inconvénients : fonctionnalités limitées.
- Exemples : VxWorsks, QNX, eCos, pSOS, VRTX, µC/OS II...



RTOS PAR EXTENSION

- Avantages : de nombreuses fonctionnalités, coopération tâches TR et processus non TR.
- Inconvénients : n 'est pas un vrai RTOS monolithique
- Exemples : RTLinux, RTAI...



PARTIE 2 :
LES PATCHS PREEMPTIFS

PATCH DU NOYAU

- Il existe deux principaux patchs permettant d'améliorer les performances du noyau Linux 2.4 au niveau du temps de latence en diminuant la taille des sections du noyau non préemptibles :
  - Patch Preempt Kernel
  - Patch Low Latency

PATCH DU NOYAU

- Le patch Preempt Kernel est maintenu par Robert M. Love et soutenu par MontaVista :

  http://www.tech9.net/rml/linux

- Le principe du patch est de rendre le noyau totalement préemptible et de protéger les données du noyau par des mutexs ou spinlocks.

- A chaque fois qu ’un événement apparaît et rend un processus de plus forte priorité prêt, le noyau préempte le processus courant et exécute le processus de plus forte priorité.

PATCH DU NOYAU

- Ce mécanisme de préemption ne doit pas être mis en oeuvre :
  - lors du service d ’une interruption (interruption handling)
  - lors de la prise d ’un verrou spinlock, writelock, ou readlock (verrou utilisé pour le SMP : Symmetric Multi Processing)
  - lors de l ’exécution de l ’ordonnanceur lui-même
  - lors de l ’exécution du processing Bottom Half (BH)

- Voir http://www.linuxdevices.com/articles/AT4185744181.html pour plus de détails.

PATCH DU NOYAU

- Le patch Low Latency est maintenu par Andrew Morton : http://www.zip.com.au/~akpm/linux/schedlat.html

- Le principe est un peu différent car au lieu d'opter pour une stratégie systématique du noyau tout préemptif, les développeurs du patch ont préféré effectuer une analyse du noyau afin d'ajouter des points de préemption obligatoire (appel de schedule()) subtilement placés dans les sources du noyau afin de casser des boucles d'attente trop longues.

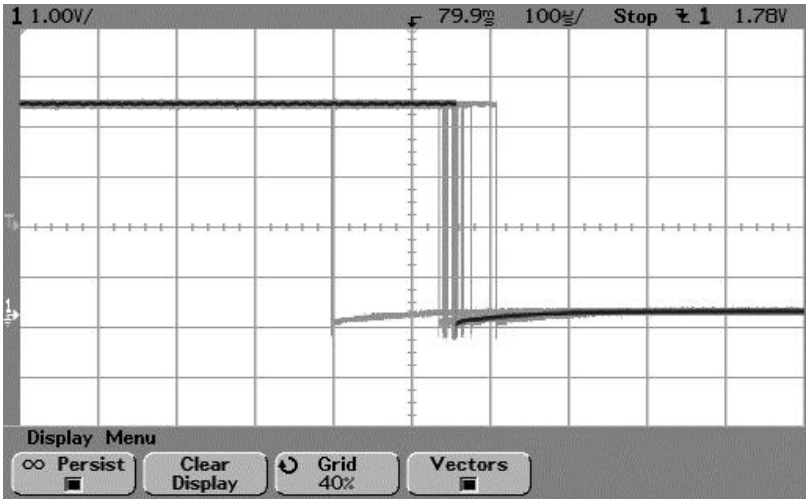
PETITE EXPERIENCE. SUITE

- Mesure effectuée à l'oscilloscope lors de l'utilisation du programme `square` décrit précédemment.

- Dans le cas du noyau 2.4.20 modifié par le patch *Preempt Kernel* et subissant la même charge que pour les autres mesures, nous obtenons la courbe suivante, indiquant une latence maximale légèrement supérieure à 200 µs.

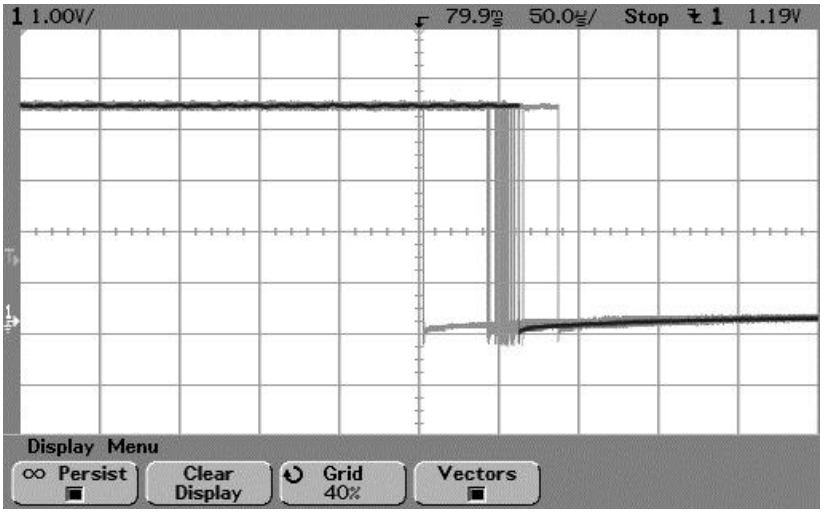
- Dans le cas du patch L*ow Latency*, nous obtenons un meilleur résultat avec une latence maximale d'environ 80 µs.

PETITE EXPERIENCE. SUITE



Exécution de `square` avec le patch Preempt Kernel

PETITE EXPERIENCE. SUITE



Exécution de `square` avec le patch Low Latency

CONCLUSION

- Les patchs précédents permettent d'améliorer les temps de latence sur le noyau Linux standard mais le concept se rapproche plus d'une amélioration de la qualité de service que du temps réel dur.

- La réactivité est maintenant de l'ordre de quelques dizaines à quelques centaines de µs au lieu de quelques dizaines à quelques centaines de ms voire plus pour un noyau Linux standard !

- Des mesures faites par Metrowerks à l'aide du Latency Benchmark de Systems Software Labs montrent que dans 99,5 % des cas le temps de latence est inférieur à 200 µs pour les 2 patchs (voir le whitepaper *Linux as a Real-Time operating System* de Metrowerks).

CONCLUSION

- Cela est à voir comme une solution intermédiaire où la mise en œuvre et la programmation restent simples pour développer des applications TR.

- Une fusion des deux patchs est envisageable dans le futur noyau 2.6.

PARTIE 3 :
LES OFFRES LINUX TEMPS REEL

LES OFFRES LINUX TEMPS REEL

- Les offres de version de Linux embarqué et Temps Réel peuvent être rangées dans l'une des 2 catégories suivantes :
  - Les distributions Linux Temps Réel commerciales.
  - Les distributions Linux Temps Réel libres.

LINUX TEMPS REEL COMMERCIAL

- Montavista/ Professional or Carrier Grade or Consumer Electronics Edition (ex Hard Hat Linux)
- Lineo-Metrowerks-Motorola/Creation Suite for Linux (ex Embeddix)
- LynuxWorks/BlueCat RT
- TimeSys/Linux RTOS Professional or Standard Edition
- RTLinux/Pro

- Il y a toujours les solutions TR commerciales non Linux
  - pSOS/VxWorks, QNX, LynxOS...

LINUX TEMPS REEL OPEN SOURCE

- FSMLabs/RTLinux/free (ex OpenRTLinux GPL)
  http://fsmlabs.com/community/
- RTAI : Real Time Application Interface
  http://www.aero.polimi.it/~rtai/

- eCOS
  http://sources.redhat.com/ecos/
- KURT. Kansas University Real-Time Linux
  http://www.ittc.ku.edu/kurt/
- ADEOS
  http://www.nongnu.org/adeos/
  The purpose of Adeos is to provide a flexible environment for sharing hardware resources among multiple operating systems, or among multiple instances of a single OS.

TEMPS REEL COMMERCIAL : VxWorks ET pSOS

- Solution commerciale TR non Linux. Noyau TR. WindRiver Systems.
  - http://www.wrs.com
- L'encore Numéro 1 dans le domaine du TR et de l'embarqué…

- **VxWorks :**
  - **Scalable (simple to complex designs)**
  - **Reliable (mission-critical applications, ABS)**
  - **CPU's PowerPc, 68K, CPU32, ColdFire, MCORE, 80x86 and Pentium, i960, ARM and StrongARM, MIPS, SH, SPARC, NECV8xx, M32 R/D, RAD6000, ST 20, TriCore)**
  - **Graphic Development Platform**
  - **Cross-development**
  - **Support/Documentation**
  - **POSIX 1003.1b compliant**
  - **Networking**
  - **Tornado II embedded development platform**

TEMPS REEL COMMERCIAL : QNX

- Solution commerciale TR non Linux. Système TR. QNX software Systems.
  - http://www.qnx.com

- **QNX :**
  - **Highly reliable**
  - **all generic x86 based processors(386+)**
  - **Scalable (modules)**
  - **Deterministic**
  - **"QNX Neutrino™ real-time OS, "the most advanced RTOS on the market"**
  - **Networking**
  - **Graphical development tools and debugger**
  - **Visual design tools (C code form cut and paste)**

TEMPS REEL COMMERCIAL : LynxOS

- Solution commerciale TR compatible Linux. Système TR. LynuxWorks Systems (ex Lynx).
  - http://www.lynxos.com/

- **LynxOS is unique in the real-time embedded software marketplace. It is a hard RTOS that combines performance, reliability, openness, and scalability together with patented technology for real-time event handling. Flexible scalability makes the LynxOS well suited for applications ranging from large and complex switching systems down to small highly embedded products. LynxOS is binary compatible with the BlueCat Linux, enabling users to take advantage of the best configuration for their needs. In addition, LynuxWorks also supports traditional UNIX and Java and supports processors from Intel, Motorola, and MIPS. LynxOS offers users a choice of software application interfaces, a large number of development tools, scalability and memory efficiency which reflect the many years of expertise LynuxWorks has in the real-time embedded systems market.**

LINUX TEMPS REEL COMMERCIAL

- MontaVista/Professional or Carrier Grade or Consumer Electronics Edition :
  - Solution générale (et TR) pour l'embarqué
  - http://www.mvista.com/
  - kit d 'évaluation disponible (preview kit)

- MontaVista Linux Professional Edition
- MontaVista Linux Carrier Grade Edition
- MontaVista Linux Consumer Electronics Edition

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 1 : application du patch Preempt Kernel
  - Modification de l 'implémentation des IPC (verrou ou sémaphore tournant ou spinlock) pour le SMP (Symmetric Multi Processing).
  - Rescheduling en retour d 'interruption
- Plus d 'infos : http://www.linuxdevices.com/articles/AT4185744181.html
- Le patch kpreempt de Montavista est sous licence GPL :
- http://www.tech9.net/rml/linux/

LINUX TEMPS REEL COMMERCIAL

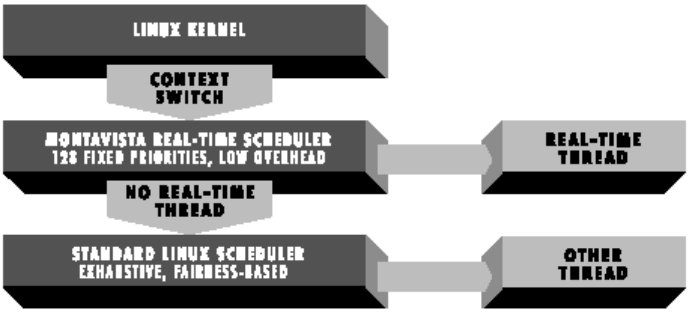
- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 1 : application du patch Preempt Kernel
  - MontaVista Software provides a Linux kernel patch. Through some very basic changes, this patch makes the Linux kernel generally preemptible (with short non-preemptible regions corresponding to the spinlocked regions in an SMP kernel). Process level responsiveness is dramatically improved, both on average and in the worst case. There are no changes whatsoever to Linux API's or semantics induced by the preemptible kernel patch. The preemptible kernel patch strictly affects the system performance domain.
  - The MontaVista Linux preemptible kernel is available and shipping for all supported architectures: PowerPC, x86, MIPS, StrongARM, XScale, SH and ARM.

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 2 : redéfinition des politiques d'ordonnancement (fixed overhead scheduler)
  - Scheduler TR qui gère tous les threads Linux marqués SCHED_FIFO et SCHED_RR basé sur leur priorité fixes (128). Les threads marqués différemment sont traités par le scheduler Linux standard.
  - On n 'étend pas l 'API Linux standard.
  - The MontaVista Real-Time Scheduler is a fully open, GPL-licensed source code module for Linux. This module for the Linux kernel addresses the serious process selection and dispatch time delays described in the previous section of this document.

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 2 : redéfinition des politiques d'ordonnancement (fixed overhead scheduler)



LINUX TEMPS REEL COMMERCIAL

- Caractéristiques TR de MontaVista/Professional Edition :
- Solution 2 : redéfinition des politiques d'ordonnancement (fiexed overhead scheduler)
  - Emulation de l 'API VxWorks et pSOS.
- Le scheduler RT de Montavista est sous licence GPL : http://sourceforge.net/projects/rtsched

LINUX TEMPS REEL COMMERCIAL

- Lineo-Metrowerks-Motorola/Creation Suite for Linux :
  - http://www.metrowerks.com/
  - kit d 'évaluation disponible
- L 'outil de configuration du noyau LKIT (Linux Kernel Import Tool) supporte l 'application de patchs du noyau donc les patchs préemptifs

LINUX TEMPS REEL COMMERCIAL

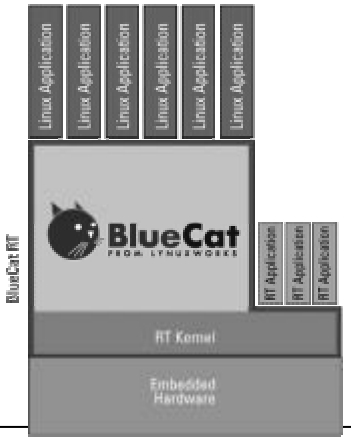
- LynuxWorks/BlueCat RT:
  - Extension Temps Réel de BlueCat
  - http://www.bluecat.com/products/bluecat-rt/bluecat-rt.php3

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques de LynuxWorks/BlueCat RT :
- Mise en œuvre de l'extension TR par double noyau. Utilisation d'un noyau TR (licence RTLinux/Pro)

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques de LynuxWorks/BlueCat RT :



LINUX TEMPS REEL COMMERCIAL

- TimeSys/Linux RTOS Professional ou Standard Edition :
  - http://www.timesys.com
- Les 2 produits sont basé sur un noyau Linux modifié pour les contraintes Temps Réel :
  - TimeSys Linux GPL
- Application d'un patch propriétaire de type kpreempt (Montavista) pour rendre le noyau préemptif : temps réel mou

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques de TimeSys/Linux RTOS Edition :
- Royalty-free real-time capabilities that transform Linux into a real-time operating system (RTOS)
- TimeSys Linux Board Support Package (BSP)
  - o TimeSys Linux GPL kernel, based on the 2.4.18 Linux kernel, that delivers:
    - + Full kernel preemption
    - + Unlimited process priorities
    - + Enhanced schedulers
    - + Priority schedulable interrupt handlers and Soft IRQs
    - + High Availability/Carrier Grade features
    - + POSIX Message Queues
  - o Lowest-latency Linux kernel on the market

LINUX TEMPS REEL COMMERCIAL

- Caractéristiques de TimeSys/Linux RTOS Edition :
- Unique priority inversion avoidance mechanisms (Priority Inheritance, Priority Ceiling Emulation Protocol)
- High resolution timers
- **True soft to hard real-time predictability and performance**
- Available ready-to-run on more than 65 specific embedded development boards spanning 8 processor architectures and 35 unique processors
- Powerful, multi-threaded local and remote debugging with gdb
- Detailed user and API documentation

LINUX TEMPS REEL OPEN SOURCE

- RTlinux/Free Pro :
- Solution libre et commerciale d'extension TR de Linux. Système TR. FSM Labs
- http://www.rtlinux.org
- Mise en place d 'une couche d 'abstraction.
- Mise en service sous forme de modules Linux.
- Linux apparaît comme la tâche de fond de plus faible priorité.
- RTLinux propose une API cohérente pour une programmation TR.
- Existe en une version tenant sur une disquette : projet miniRTL.

RTAI

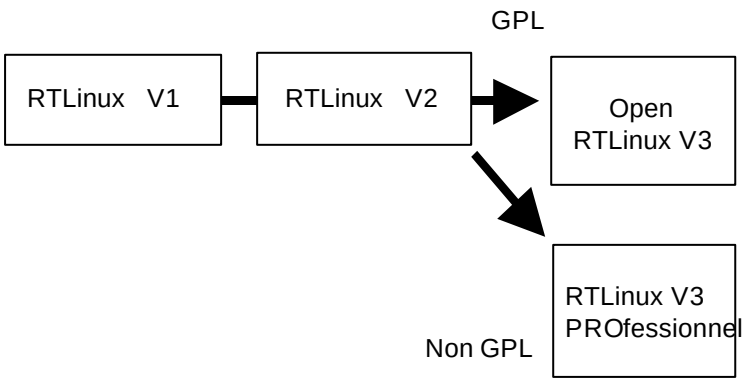
- Solution libre d'extension TR de Linux. Système TR. Université de Milan en Italie.
- http://www.aero.polimi.it/~rtai/
- Mise en place d 'une couche d 'abstraction
- Mise en service sous forme de modules Linux :
  - 3 entités/modules de base :
    - Dispatcher d'interruptions entre tâches TR ou noyau Linux.
    - Ordonnanceur TR.
    - FIFOS de communication.

PARTIE 4 :
PRESENTATION DETAILLEE DE RTLINUX

HISTOIRE DE RTLinux

- RTLinux a été développé originellement par un chercheur de l'université de New Mexico avec l'aide d'un étudiant : Victor Yodaiken et Michael Barabanov.
- D 'abord sous licence GPL, un brevet (US Patent No. 5,995,745) a été déposé sur le principe de fonctionnement de RTLinux, ce qui est incompatible avec la notion de logiciel libre. Une entreprise privée FSMLabs a été créée pour distribuer RTLinux.
- Devant le tollé général, FSMLabs décide de distribuer une version GPL OpenRTLinux et une version commerciale RTLinux/PRO plus complète

HISTOIRE DE RTLinux



HISTOIRE DE RTLinux

- RTLinux V 1 :
  - Noyau Linux 2.0.x.
  - Pas de support SMP (*Symmetric Multi Processor*).
  - API simple de près 15 fonctions.
- RTLinux V2 :
  - Noyau Linux 2.2.x.
  - SMP.
  - API style POSIX.
- RTLinux V3 :
  - Noyau Linux 2.2.19 &2.4.40 (26/11/2001).
  - Version GPL et PRO.

FONCTIONNALITES DE RTLinux

- Mise en service sous forme de modules Linux.
- Les tâches TR sont chargées comme des modules Linux.
- Linux apparaît comme la tâche de fond.
- RTLinux propose une API simple pour une programmation TR. L'API POSIX thread est supportée pour l'écriture de tâches TR.
- **Les tâches TR sont périodiques. Il n'y a pas de support de tâches apériodiques !**
- Les communications entre processus Linux et les tâches TR se font par des FIFOS.
- Le dernière version stable supporte les processeurs x86, PowerPC et Alpha.

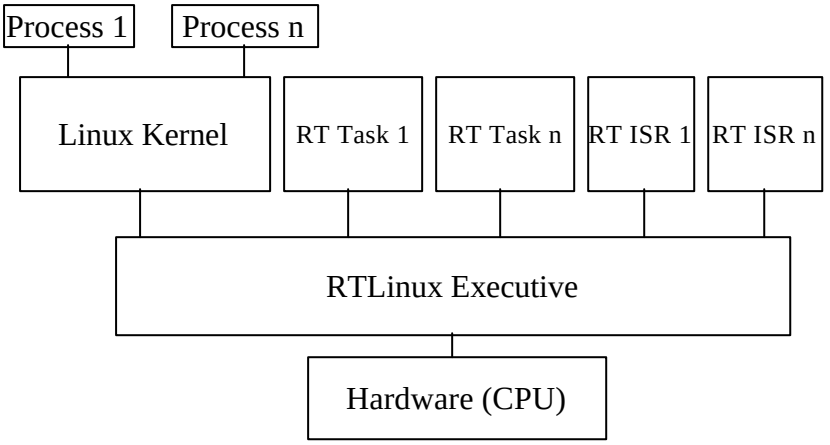
PRINCIPES DE RTLinux

- Une couche logicielle est mise en place entre le noyau Linux et le contrôleur d'interuptions afin de prévenir les masquages d'interruptions depuis Linux (condition requise pour des performances TR).
- Les macros et instructions ASM CLI, STI et IRET sont remplacées par leur version d'émulation logicielle correspondante.
- Linux apparaît comme la tâche TR de fond de plus faible priorité.
- Toute tâche TR est préemptible.
- Tout handler d'interruption (ISR) ne peut pas être ralenti par des opérations non TR.

TACHES RTLinux

- Une tâche TR RTLinux apparaît comme un module Linux. Elle est chargée dynamiquement en mémoire.
- Toutes les tâches sont donc dans le même espace d'adressage (espace noyau) et sont exécutées en mode privilégié.
- On peut donc planter très facilement le système en cas de mauvaise programmation (comme un mauvais module Linux).
- Le changement de contexte de tâche se fait sur les registres entiers du processeur par défaut (pas de sauvegarde des registres FPU).
- Allocation de ressource (mémoire par *kmalloc()*) de façon statique à la création de la tâche TR.

TACHES RTLinux



ORDONNANCEUR RTLinux

- L'ordonnanceur (*scheduler*) est un module Linux à charger en mémoire.
- Par défaut, l'ordonnanceur est basé sur le traitement des priorités fixes des tâches TR à leur création. La tâche TR de plus forte priorité prête est exécutée.
- Il y a d'autres politiques d'ordonnancement possibles (chargement du module Linux correspondant :
  - ordonnanceur EDF (*Earliest Deadline First*) : ordonnanceur basé sur les priorités de tâches où la tâche TR prête ayant une deadline la plus proche sera exécutée.
  - Ordonnanceur RM (*Rate Monotonic*) : ordonnanceur statique pour tâches périodiques où la tâche ayant la plus petite période se verra assigner la plus forte priorité.

IPC RTLinux

- Les Communications Inter Processus IPC se font par des FIFOS RT.
- Les buffers des FIFOS RT sont alloués dans l'espace noyau.
- Le nombre maximal de FIFOS est fixe (fixé à la compilation du noyau).
- Une FIFO est unidirectionnelle. Pour des communications bidirectionnelles, il en faut donc 2.

IPC RTLinux

- Des IPC par mémoire partagée sont possibles entre processus Linux et tâches RTLinux.
- Il convient de charger le module mbuff et de travailler avec le périphérique /dev/mbuff.
- Un processus Linux peut mapper une zone mémoire allouée dans l'espace noyau dans son propre espace d'adressage.
- La mémoire allouée dans l'espace noyau n'est pas forcément physiquement contigue. Elle ne peut être swappée.

IPC RTLinux

- FIFOS vs mémoire partagée :

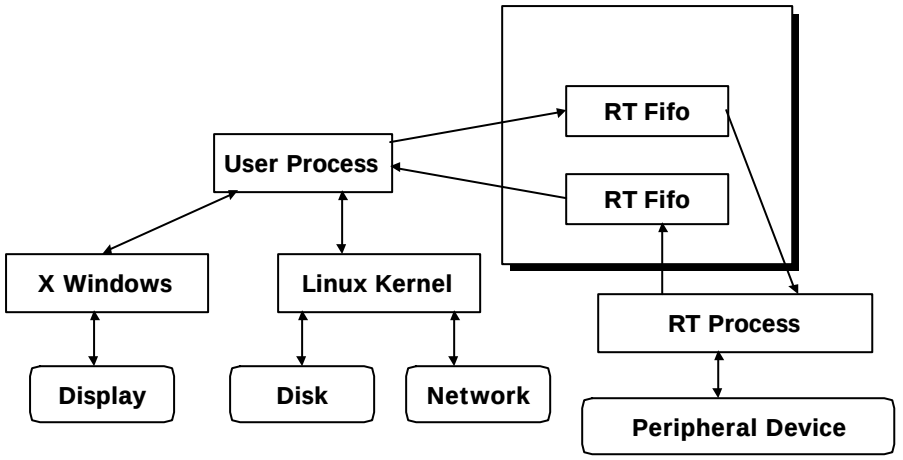
Queue data, no protocol needed to prevent overwrites.	No queuing of data. Need to follow explicit protocol.
Message boundaries not maintained.	Can write structured data into memory.
Support blocking for synchronization, no polling required.	Need to poll for synchronization.
Fifos are point-to-point channels.	Can have any number of tasks sharing memory.
Maximum number of fifos statically defined.	Physical memory is the only limit.

DEVELOPPEMENT SOUS RTLinux

- Le développement d'applications sous RTLinux suit les règles suivantes :
  - Découpage en 2 parties : TR et non TR.
  - La partie TR doit être la plus simple et la plus courte possible.
  - Le reste est non TR et sera développé dans l'espace user sous forme de processus Linux.
  - Les processus Linux et les tâches TR pourront communiquer par des FIFOS ou par mémoire partagée.

- L'API POSIX thread existant déjà sous Linux a été portée sous RTLinux. Cela facilite la migration d'un développeur Linux vers RTLinux.

DEVELOPPEMENT SOUS RTLinux

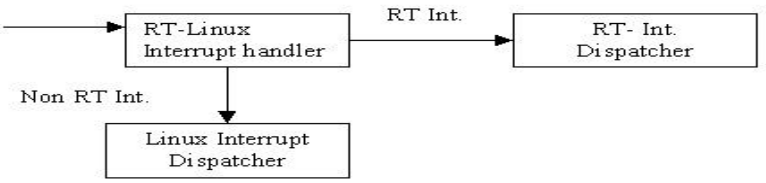


INTERRUPTIONS SOUS RTLinux

- Il y a 2 types d'interruptions :
  - Interruption logicielle (*soft interrupt*). Pas de performances TR. Des fonctions noyau peuvent être utilisées dans le handler d'IT. Peut être inhibée sur une longue période de temps. Servie lorsque le système TR rend la main à Linux (tâche TR de fond).
  - Interruption matérielle (*hard interrupt*). Interruption par le matériel. Le handler d'IT RT est d'abord exécuté et s'il est partagé avec Linux, l'interruption sera ensuite servie sous Linux. Peu de fonctions noyau peuvent être utilisées ici.

INTERRUPTIONS SOUS RTLinux

- Dispatcher d'interruptions de RTLinux :



INTERRUPTIONS SOUS RTLinux

- Une interruption IRQ (*Interrupt ReQuest*) peut être demandée sous RTLinux :
  - *rtl_request_irq(irq, handler, regs)*.
- Et libérée :
  - *rtl_free_irq(irq).*
- Le handler d'IT est exécuté avec les interruptions inhibées. Pour les revalider, il faut faire appel à la fonction :
  - *rtl_hard_enable_irq(irq).*

CALCUL REEL SOUS RTLinux

- Par défaut, une tâche RTLinux ne peut pas faire des opérations sur des nombres réels à cause du changement de contexte sur les registres entiers du processeur par RTLinux.
- Pour valider les opérations sur des nombres réels et modifier le changement de contexte de la tâche TR, on utilise la fonction :
  - *pthread_setfp_np(thread,flag).*
  - flag=1 : calcul sur réels validé.
  - flag=0 : calcul sur réels dévalidé.

MUTUELLE EXCLUSION SOUS RTLinux

- RTLinux supporte à travers l'API thread des fonctions de mutuelle exclusion :
  - fonctions pthread_mutex_xxx().
- Cela permet d'implémenter des sémaphores.
- Il n'y a pas de support par RTLinux des problèmes de *deadlock* et d'inversion de priorité. Il faut donc faire attention...

RTLinux vs RTAI

- Compatibilité de l 'API :
  - RTLinux implémente l 'API POSIX 1003.13 "minimal realtime operating system".
  - RTAI a sa propre API dérivée de RTLinux V1. Pas trop de cohérence dans l 'API. On peut faire la même chose avec différentes primitives. RTAI maintient une compatibilité avec l 'API RTLinux V1. Il existe un module pour implémenter POSIX 1003.1c. (PThreads) et 1003.1b (Pqueues).
- Processeurs :
  - RTLinux : i386, PPC, ARM.
  - RTAI : i386, MIPS, PPC, ARM, m68k-nommu.

RTLinux vs RTAI

- Etat de développement :
  - RTLinux : version 3.1 (Mai 2001). Il y a des bugs connus. Noyau Linux supporté : 2.2.19, 2.4.4 (non officiel 2.4.18, pas de SMP).
  - RTAI : noyau Linux 2.4.18.
- Debug :
  - RTLinux : au niveau source avec GDB et DDD avec support SMP. Traçage du noyau et des événements de l 'application.
  - RTAI : kgdb, au niveau source par liaison série avec un hôte. Outil *Linux Trace Toolkit* (LTT).

RTLinux vs RTAI

- Gestion mémoire :
  - RTLinux : par défaut, la mémoire doit être allouée avant le lancement de la tâche. Pas d 'allocation dynamique. Mémoire partagée entre RTLinux et Linux gérée par le module non POSIX mbuff (*mbuff_alloc(), mbuff_free()*).
  - RTAI : allocation dynamique de la mémoire non TR (*rt_malloc(), rt_free()*). Mémoire partagée RTLinux /Linux gérée par le module MBUFF (*mbuff_alloc(), mbuff_free()*) ou le module non POSIX SHMEM (*rtai_malloc(), rtai_free(), rtai_kmalloc(), rtai_kfree()*).

RTLinux vs RTAI

- IPC :
  - RTLinux : FIFOS non POSIX (vu comme un pipe UNIX).
  - RTAI : FIFOS non POSIX. Mailbox équivalent à une FIFO où l 'on peut y lire/écrire des messages structurés. Remote Procedure Calls (RPC) entre 2 tâches (net_rpc entre 2 tâches TR de 2 machines distantes). Les IPC sont très riches (contrairement à RTLinux, ce qui est une des raisons du développement de RTAI) redondants et incompatibles…C 'est le point faible de RTAI !

RTLinux vs RTAI

- Synchronisation :
  - RTLinux : Mutex POSIX. Sémaphores POSIX.
  - RTAI : Mutex POSIX. Sémaphores POSIX et sémaphores RTAI (*rtf_sem_init(), rtf_sem_wait()…*).
- Temps Réel en mode utilisateur :
  - RTLinux : signaux TR. Une interruption matérielle peut être traitée par un handler en mode *user space*. On ne peut pas utiliser dans le handler des appels système Linux ou RTLinux. Implémenté dans RTLinux/Pro.
  - RTAI : module LXRT permet d 'utiliser les services RTAI en mode *user space*.

PARTIE 5 :
MISE EN ŒUVRE DE RTLINUX

INSTALLATION DE RTLinux

- Téléchargement depuis le serveur FTP anonyme ftp.fsmlabs.com de RTLinux/Free :
  - Fichier rtlinux-3.1.tar.gz.

- Décompactage :

```
# tar zxvf rtlinux-3.1.tar.gz
# mv rtlinux-3.1 /usr/src/
```

- Vérification des versions des patchs Linux pour RTLinux :
  - kernel_patch-2.2.19.
  - kernel_patch-2.4.4.

INSTALLATION DE RTLinux

- Téléchargement depuis le serveur FTP anonyme ftp.kernel.org du noyau Linux :
  - Fichier linux-2.4.4.tar.bz2.

- Décompactage :

```
# bunzip2 linux-2.4.4.tar.bz2
# tar xvf linux-2.4.4.tar
# mv linux /usr/src/
```


INSTALLATION DE RTLinux

- Patch du noyau Linux :

```
# cd /usr/src/linux
# patch –p1 < /usr/src/rtlinux-3.1/kernel_patch-
  2.4.4
```

- Compilation du noyau Linux patché :

```
# make config / menuconfig / xconfig
# make dep
# make bzImage
# make modules
# make modules_install
# cp arch/i386/boot/bzImage /boot/rtzImage-2.4.4
```

INSTALLATION DE RTLinux

- Configuration de LILO (*LInux LOader*). Edition du fichier /etc/lilo.conf :

```
Image=/boot/vmlinuz-2.4.18-0.13
    label=linux
    Initrd=/boot/initrd-2.4.18-0.13.img
    read-only
    root=/dev/hda2
Image=/boot/rtzImage-2.4.4
    label=rtlinux
    initrd=/boot/initrd-2.4.18-0.13.img
    read-only
    root=/dev/hda2
```

- Installation :

```
# lilo
```


INSTALLATION DE RTLinux

- Compilation de RTLinux et installation :

```
# cd /usr/src/rtlinux-3.1
# ln –sf ../linux linux
# make config  ou menuconfig ou xconfig
# make
# make devices
# make install
```

INSTALLATION DE RTLinux

- Reboot de la machine. Choisir le noyau Linux TR rtlinux.
- Installation des modules RTLinux :
  `# modprobe -a rtl rtl_time rtl_sched rtl_posixio rtl_fifo`
- Notes :
  - Exemples : dans /usr/rtlinux/examples.
  - Documentation RTLinux: dans /usr/src/rtlinux-3.1/doc.

DEVELOPPEMENT D ’UNE APPLICATION RTLinux

- On créera préférentiellement une application TR RTLinux utilisant l ’API POSIX thread.
- Un thread est vu comme une tâche TR.
- Un thread est en fait un programme en cours d ’exécution qui partage avec les autres threads son espace d ’adressage et donc ses variables. C ’est ce qui se passe avec 2 tâches RTLinux !
- On doit avoir quelques bases sur l ’écriture de modules Linux...

DEVELOPPEMENT D ’UNE APPLICATION RTLinux

- L ’application TR est composée de tâches TR ou threads.
- Un thread est créé à l ’aide de la fonction *pthread_create()*. Cette fonction doit être appelée dans la fonction *init_module()*. On rappelle que l’on doit développer un module Linux pour créer des threads TR.

```
#include <pthread.h>
   int pthread_create(pthread_t * thread, pthread_attr_t * attr, void
   *(*start_routine)(void *), void * arg);
```


DEVELOPPEMENT D ’UNE APPLICATION RTLinux

- Un thread est créé avec des attributs (objet attr). Si attr est NULL, alors le thread est créé avec les attributs par défaut.
- On utilisera les fonctions POSIX :
  - *pthread_attr_init()*
  - *pthread_attr_setschedparam()*
  - *pthread_attr_getschedparam()*

DEVELOPPEMENT D ’UNE APPLICATION RTLinux

- Un thread est détruit à l ’aides des fonctions POSIX dans la fonction *cleanup_module()* du module Linux :
  - pthread_cancel(pthread_t thread);
  - pthread_delete_np(pthread_t thread)

DEVELOPPEMENT D ’UNE APPLICATION RTLinux

- Manipulation du temps :

```
#include <rtl_time.h>

int clock_gettime(clockid_t clock_id, struct timespec *ts);
   hrtime_t clock_gethrtime(clockid_t clock);

struct timespec {
     time_t tv_sec; /* seconds */
     long tv_nsec; /* nanoseconds */
   };

hrtime_t timespec_to_ns(const struct timespec *ts); struct timespec
   timespec_from_ns(hrtime_t t) const struct timespec *
   hrt2ts(hrtime_tvalue);
```

DEVELOPPEMENT D 'UNE APPLICATION RTLinux

- Contrôle de l 'ordonnancement :

```
int pthread_setschedparam(pthread_t thread, int policy, const struct
    sched_param *param);

int pthread_make_periodic_np(pthread_t thread, hrtime_t        start_time,
    hrtime_t period);

int pthread_wait_np(void);

int sched_get_priority_max(int policy);
int sched_get_priority_min(int policy);
```


DEVELOPPEMENT D 'UNE APPLICATION RTLinux

- Exemple : HELLO :

```
#include <rtl.h>
#include <time.h>
#include <pthread.h>
pthread_t thread;
void * start_routine(void *arg)
{ struct sched_param p;
  p . sched_priority = 1;
  pthread_setschedparam (pthread_self(), SCHED_FIFO, &p);
  pthread_make_periodic_np (pthread_self(), gethrtime(),    500000000);
  while (1) {
        pthread_wait_np();
        rtl_printf("I'm here; my arg is %x\n", (unsigned) arg);
  }
  return 0;
}
```


DEVELOPPEMENT D 'UNE APPLICATION RTLinux

- Exemple : HELLO :

```
int init_module(void)
{
  return pthread_create (&thread, NULL, start_routine, 0);
 }
void cleanup_module(void)
{
  pthread_cancel (thread);
  pthread_join (thread, NULL);
}
```


DEVELOPPEMENT D 'UNE APPLICATION RTLinux

- Exemple : HELLO :
- Installation RTLinux :

  # modprobe -a rtl rtl_time rtl_sched rtl_posixio rtl_fifo

- Installation de l 'exemple :

  # insmod my_program.o (# rmmod my_program)

- Lancement de l 'application HELLO :

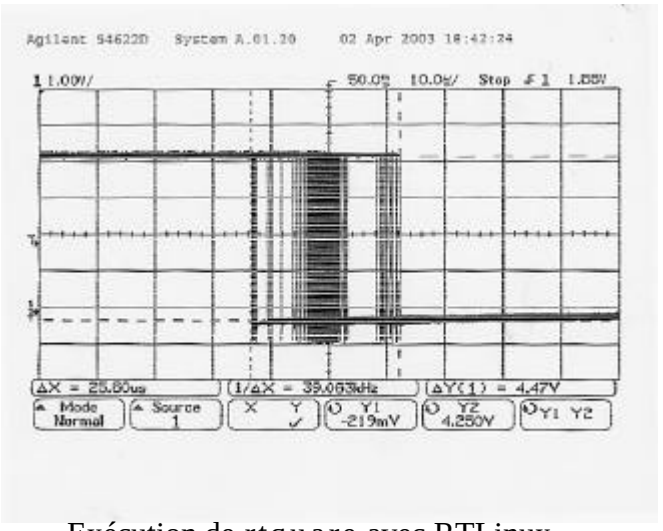
  # rtlinux start my_program
  # rtlinux stop my_program
  # rtlinux status my_program

LA PETITE EXPERIENCE. SUITE ET FIN

- Mesure effectuée à l'oscilloscope lors de l'utilisation du programme `rtsquare` décrit précédemment modifié pour s'exécuter comme tâche Temps Réel sous RTLinux.

Dans les conditions de stress du système (écriture continu d'un fichier de 50 Mo), on obtient le résultat ci-dessous à l'oscilloscope. La mesure du jitter donne la valeur de 25.6 µs comparés aux 17.6 ms du noyau LINUX standard, ce qui correspond environ à un rapport 1000.

LA PETITE EXPERIENCE. SUITE ET FIN



Exécution de `rtquare` avec RTLinux

LA PETITE EXPERIENCE. SUITE ET FIN

```
#include <rtl_time.h>
#include <rtl_sched.h>
#include <asm/io.h>

/* Adresse du port parallèle */
#define LPT 0x378

/* Période de sollicitation de 50 ms */
int period=50000000;

/* Valeur envoyée sur le port parallèle */
int nibl=0x01;

/* Identifiant du thread applicatif */
pthread_t thread;
```


LA PETITE EXPERIENCE. SUITE ET FIN

```
/* Corps du thread applicatif */
void * bit_toggle(void *t)
{
    /* On programme un réveil de la tâche toute les 50 ms */
    pthread_make_periodic_np(thread, gethrtime(), period);

    while (1){
        /* Ecriture de la valeur sur le port // */
        outb(nibl,LPT);

        /* Calcul de la valeur suivante (négation) */
        nibl = ~nibl;

        /* Attente du réveil */
        pthread_wait_np();
    }
}
```


LA PETITE EXPERIENCE. SUITE ET FIN

```
int init_module(void)
{
    pthread_attr_t attr;
    struct sched_param sched_param;

    /* Priorité à 1 */
    pthread_attr_init (&attr);
    sched_param.sched_priority = 1;
    pthread_attr_setschedparam (&attr, &sched_param);

    /* Création du thread applicatif */
    pthread_create (&thread, &attr, bit_toggle, (void *)0);

    return 0;
}
```


LA PETITE EXPERIENCE. SUITE ET FIN

```
void cleanup_module(void)
{
    pthread_delete_np (thread);
}
```

Programme TR `rtsquare` pour RTLinux (fichier C rtsquare.c)

PARTIE 6 :
BILAN

LE CHOIX D 'UN LINUX EMBARQUE TEMPS REEL

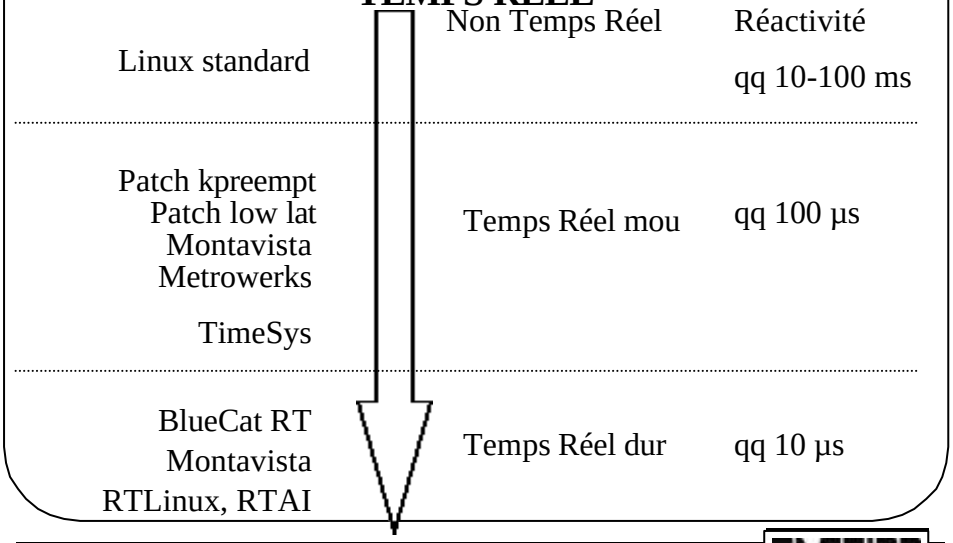
• Le choix est à faire en fonction des contraintes Temps Réel que doit respecter le système :

– Pas de contrainte. Best effort. Réactivité de qq 10ms à qq 100 ms.

–Temps Réel mou. Réactivité de qq 100 µs.

–Temps Réel dur. Réactivité de qq 10 µs

LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL



LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL

• LINUX n'est pas fondamentalement un système d'exploitation temps réel car trop généraliste.

• Par application de différents patchs (*low latency, preempt kernel*), il est possible d'avoir un système LINUX temps réel mou.

• Il est possible d'avoir un système LINUX temps dur en utilisant les extensions temps réel RTLinux, RTAI, Montavista...

LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL

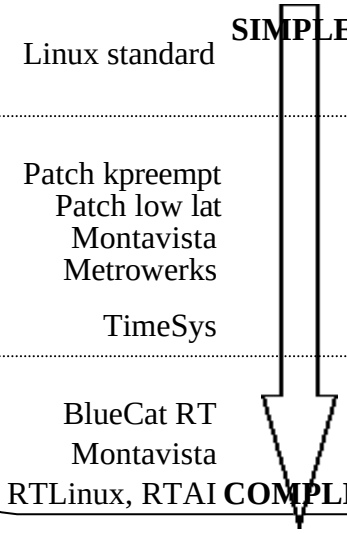
• Le choix final se fera en fonction des contraintes temporelles imposées par le processus à contrôler depuis LINUX en prenant aussi en compte la complexité de mise en oeuvre dans chaque cas :

– Configuration du noyau.

– Ecriture de l'application temps réel.

•Choisir là aussi un linux embarqué Temps Réel commercial est rassurant. Cela a aussi un coût.

LE CHOIX D ’UN LINUX EMBARQUE TEMPS REEL



RESSOURCES Linux Temps Réel

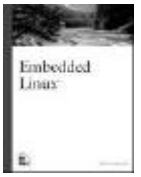
- Real-Time and Embedded Guide. H. Bruyninckx.
  http://people.mech.kuleuven.ac.be/~bruyninc/rthowto/
- Embedded Linux Howto
  http://linux-embedded.org/howto/Embedded-Linux-Howto.html
- Linux Magazine. Linux Temps Réel. Où en est-on aujourd ’hui ?
  P. Kadionik et Pierre Ficheux. Juillet-août 2003.
- http://www.enseirb.fr/~kadionik/embedded/linux_realtime/linuxrealtime.html
- La page de l ’auteur :
  http://www.enseirb.fr/~kadionik/embedded/embedded.html

THAT ’S ALL FOLKS !

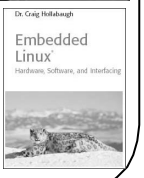
REFERENCES BIBLIOGRAPHIQUES

- **Linux embarqué. P. Ficheux. Editions Eyrolles. LA REFERENCE !**

- Embedded Linux. J. Lombardo. Editions New Riders.

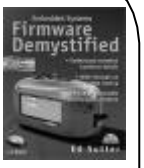


- Embedded Linux. C. Hollabaugh. Editions Addison Wesley



REFERENCES BIBLIOGRAPHIQUES

- Embedded Systems. Firmware Demystified. E. Sutter. Editions CMP Books



- The Art of Designing embedded Systems. J. Ganssle. Editions Butterworth-Heinemann

- Embedded Systems Design. A S. Berger. Editions CMP Books

REFERENCES BIBLIOGRAPHIQUES

- http://www.enseirb.fr/~kadionik/linux.html
- http://www.enseirb.fr/~kadionik/embedded.html
- Et l'ensemble des adresses www citées précédemment...