TOUT OU RIEN... MAIS PAS TOUT SIMPLE

Quoi de plus bête qu'un simple contact, direz-vous. Et pourtant, quand on y consacre quelque attention, les entrées « tout ou rien » apparaissent multiforme, exigeantes, voire tout à fait agaçantes.
Il n'y a pas d'entrées/sorties simples, il n'y a que des montages bien pensés !

Un bouton de sonnette

Les ouvrages d'initiation introduisent souvent la logique avec des illustrations « mécaniques » comme celles de la **figure 1.** On représente les variables logiques par des contacts à poussoir, disons des boutons de sonnette ; deux en série font un ET, deux en parallèle forment un OU. Evident, non ?

Grave erreur.

Les **vrais** boutons-poussoirs sont tout ce que l'on voudra sauf des éléments logiques fiables.

Une petite expérience nous en convaincra, avec le montage de la **figure 2** où l'on retrouve une vieille connaissance : la bascule 74LS74 bouclée en compteur. De telle sorte que **théoriquement** ladite bascule doit changer d'état (allumer/éteindre la LED témoin) à chaque pression sur le bouton. Plus précisément, quand on le relâche : ce qui donne la transition « ↗ » qui déclenche la bascule.

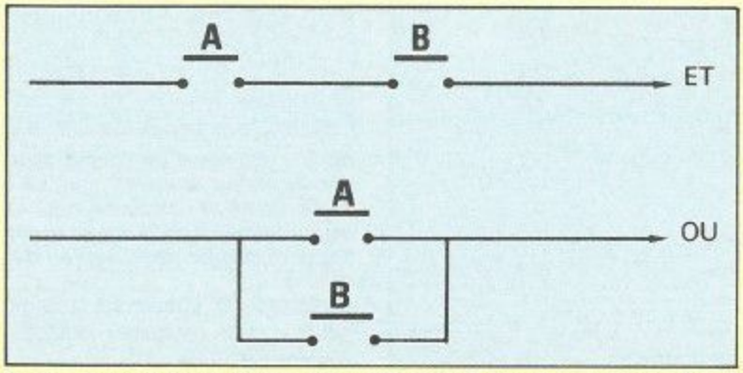


Fig. 1. – Images mécaniques des fonctions ET et OU. Pratiques pour la pédagogie, ces illustrations sont inutilisables pour attaquer des montages électroniques, à cause des rebonds de contact.

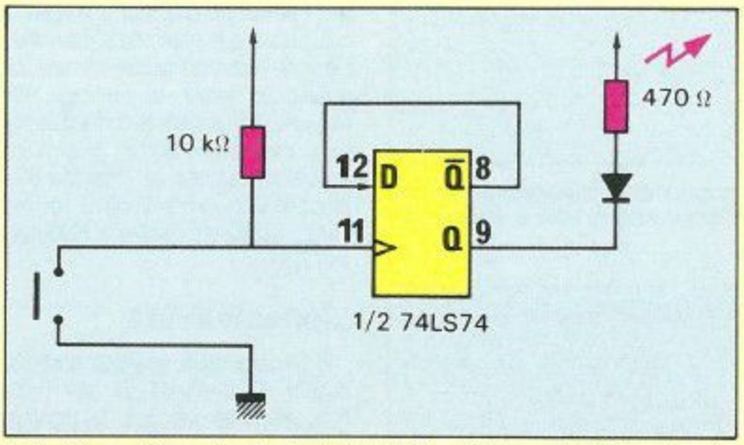


Fig. 2. – Mise en évidence des rebonds : la diode LED devrait s'allumer et s'éteindre alternativement, à chaque manœuvre du poussoir. En pratique, le résultat est aléatoire selon que le nombre de rebonds est pair ou impair...

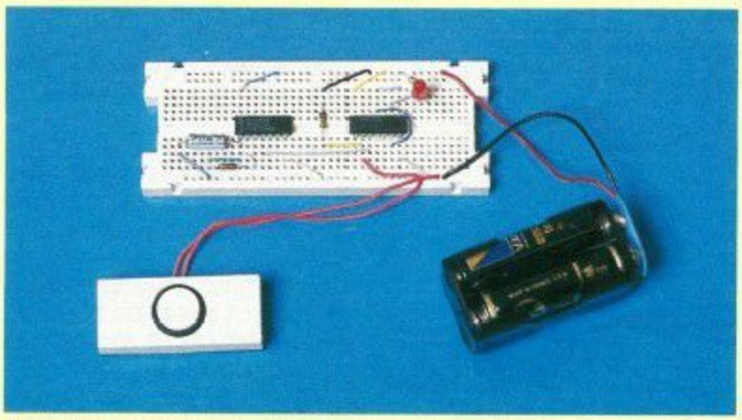
Des ratés

Le moins que l'on puisse dire, quelque doigté qu'on y mette, c'est que le dispositif a des ratés !

Certes, à chaque frappe sur le poussoir, il se passe quelque chose, la diode luminescente change d'état, parfois fugitivement, parfois définitivement après une « hésitation »...

La raison de cette contradiction entre théorie et pratique est bien connue. Le contact mécanique métal/métal a des **rebonds** très brefs lorsqu'il s'établit et lorsqu'il se rompt : c'est-à-dire que, dans la plupart des cas, ils durent au plus quelques millisecondes.

Une milliseconde de va-et-vient, c'est insignifiant quand on sonne à la porte, mais c'est énorme par rapport au temps de réponse des circuits logiques usuels. Notre bascule change d'état autant de fois qu'il y a de rebonds ; le montage ne donne le résultat attendu que lorsque (par chance) nous en provoquons un nombre impair...



Un montage anti-rebond par échantillonnage.

Anti-rebonds : les classiques

Les ouvrages d'électronique suggèrent invariablement le recours à des montages anti-rebonds qui reposent sur des mémoires en anneau **(fig. 3).**

Le plus cité fait appel à un set/reset que l'on fabrique avec des NAND ou des NOR. Le contact est à **deux positions,** ce qui correspond aux éléments **commutateurs,** au sens commun du terme. Le fonctionnement est évident, et parfaitement fiable : lorsque le contact change de position, le point opposé du set/reset est relié à la masse, et la mémoire change d'état.

Les rebonds n'ont aucune incidence, car le contact mobile ne peut que provoquer plusieurs mises à la masse d'affilée du même côté, donc ne fera que « confirmer » le changement effectué.

Le meilleur choix pour ce type de montage est le circuit 74LS279 avec ses quatre bistables : on traite quatre entrées avec un seul circuit intégré, contre deux seulement si on se contente de 74LS00.

Une variante intéressante emploie un simple anneau à deux inverseurs ; on traite trois contacts avec un circuit très bon marché et très répandu, comme le 74LS04.

Trois fils ou deux fils ?

Les anti-rebonds de contact *(switch debounce circuits)* sont très efficaces pour traiter les commutateurs qui comportent **trois points.** Si le commutateur est dans la même boîte que le montage électronique, par exemple en face avant, on acceptera de câbler trois fils **(fig. 4).**

Le câblage et les connecteurs coûtent cher. Or il suffit de **deux fils** pour véhiculer un bit (contact ouvert/fermé). Comment faire en ce cas pour se protéger des rebonds, puisque les montages basés sur des mémoires du genre set/reset sont impossibles sans le fameux troisième fil ?

Défense de répondre hâtivement : les solutions techniques seront différentes selon l'usage que l'on veut finalement avoir du fameux contact.

Entrée d'événement...

Sur le châssis d'un certain nombre d'ordinateurs, on trouve (au moins) un bouton-poussoir baptisé « interruption », dont la fonction est la même que celle d'un bouton d'appel d'ascenseur. Il s'agit d'« attirer l'attention » du processeur par **l'événement :** « appui sur le bouton ».

Ce n'est là qu'un exemple parmi beaucoup d'autres, où l'on souhaite que l'événement soit **mémorisé,** jusqu'à sa prise en compte ; s'il s'agit effectivement d'une interruption sur un

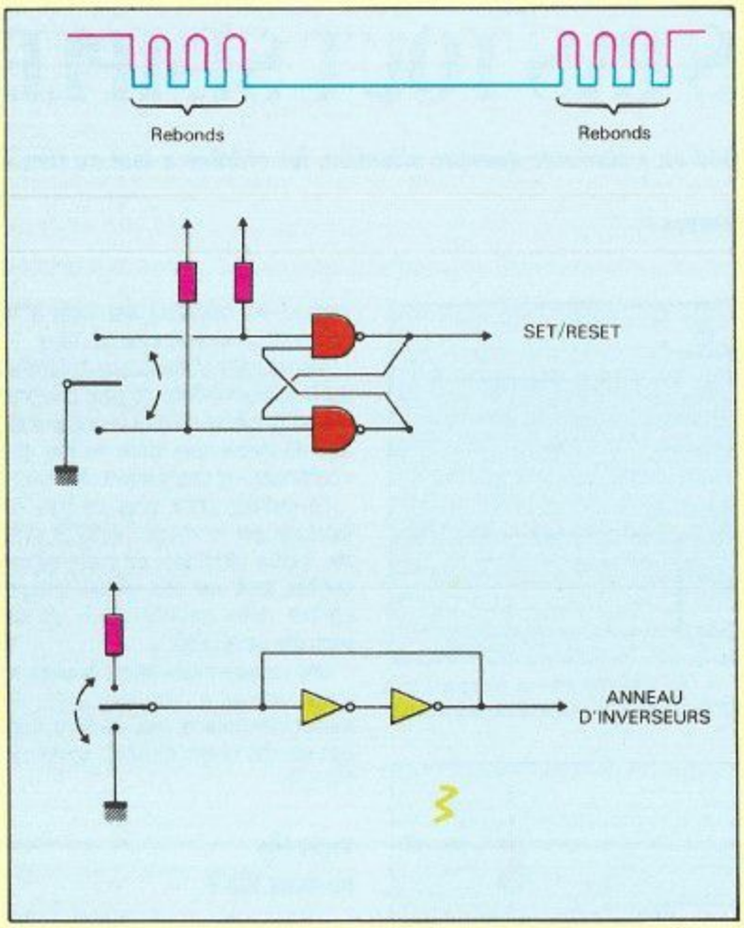


Fig. 3. – Méthodes classiques pour éliminer les rebonds grâce à des mémoires en anneau. Il faut des commutateurs avec un total de trois contacts (deux contacts de butée, un contact avec la pièce mobile).

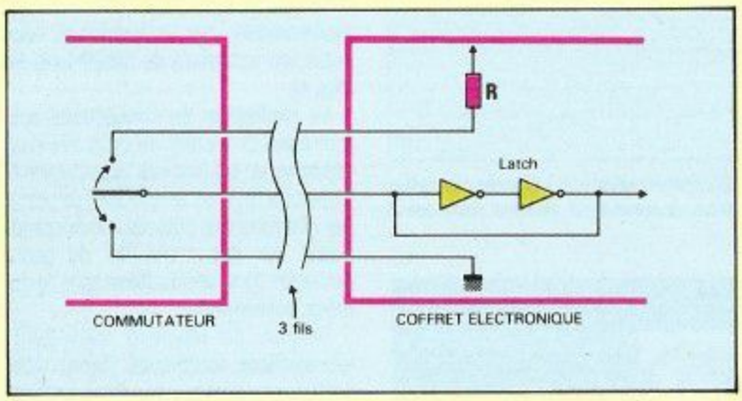


Fig. 4. – Pour déporter des contacts avec une logique anti-rebonds, il faut trois fils par contact.

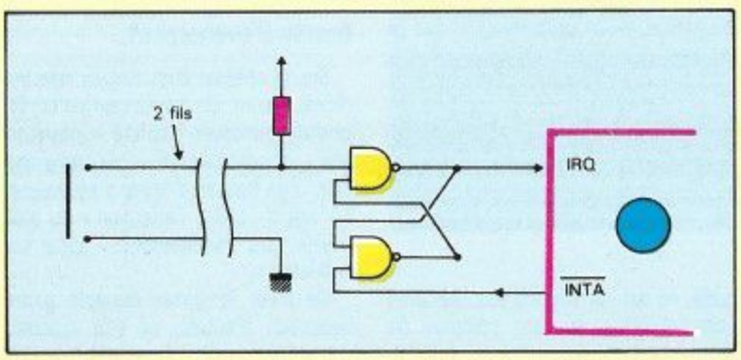


Fig. 5. – On peut se contenter de deux fils pour déporter un contact tel qu'un poussoir d'interruption. Ce dernier positionne la bascule, la remise à zéro est à la charge du processeur.

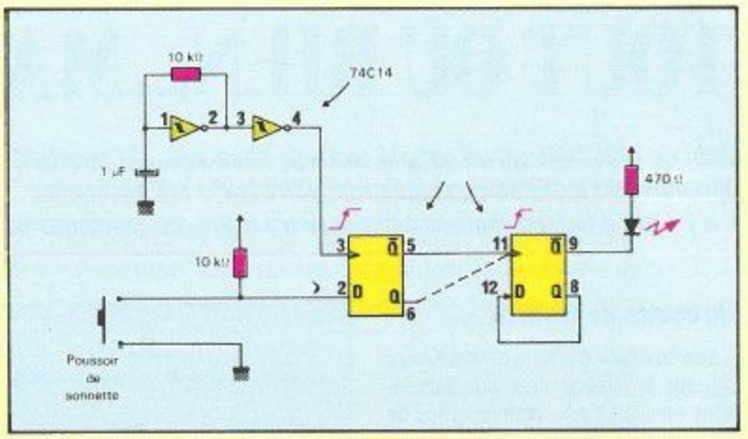


Fig. 6. – Anti-rebond par échantillonnage : l'oscillateur provoque environ 20 recopies par seconde de l'état du poussoir, dans une bascule D, qui commande à son tour la seconde bascule montée en « compteur binaire ». Les rebonds sont gommés, tandis que l'échantillonnage est plus rapide que les manœuvres (manuelles). Selon que l'on branche Q ou $\overline{Q}$ sur l'entrée horloge du compteur, celui-ci réagit au relâchement ou à l'enfoncement du poussoir.

processeur, on compte sur le logiciel pour accuser réception, « effacer » l'événement.

Dans un cas semblable, on peut utiliser un montage avec mémoire en anneau **(fig. 5)** qui a les mêmes propriétés anti-rebond que tout à l'heure ; à ceci près que le retour de la mémoire à l'état de repos est assuré par une impulsion en retour du montage. Par exemple, via un **port** de sortie d'un microprocesseur. Ou encore, grâce à une impulsion d'accusé de réception d'interruption qui existe d'origine sur certains processeurs comme le 8085 avec son $\overline{INTA}$.

... ou entrée d'état ?

Il se peut aussi que l'on n'ait pas besoin d'anti-rebond du tout parce que, de toute manière, le montage « utilisateur » va **scruter** l'entrée à des intervalles suffisamment éloignés pour que les rebonds soient sans conséquence. Le cas type, c'est lorsqu'un programme vient de loin en loin **tester** une entrée « tout ou rien ». Les intervalles de temps entre test sont tellement longs que le programme est aveugle aux rebonds (exemple : test toutes les 20 ms dans un sous-programme sur interruption d'horloge-secteur). En ce cas, on parle volontiers d'**entrée d'état** ; par opposition à une entrée d'événement.

En fait, la distinction est byzantine, comme on va le voir avec le montage de la **figure 6.** Avec ce montage, on se propose tout simplement de rendre fiable celui de la figure 2, grâce à un **échantillonnage** judicieusement dosé.

Le principe est simple : au lieu d'utiliser le contact à l'état brut, on va recopier son état dans une bascule D (1/2 de 74LS74), à des intervalles donnés par notre « horloge » habituelle à base de 74C14.

Choix d'une cadence

La question est : quelle fréquence donner à l'oscillateur ? Ou, en d'autres termes, quelle est la bonne cadence d'échantillonnage ?

Ce terme d'« échantillonnage » *(sampling)* convient bien pour nommer les copies, effectuées à intervalles réguliers, de notre signal d'entrée...

Eh bien, le délai entre « prises d'échantillons » sera déterminé :

– par le bon sens,
– par la nature du phénomène générateur.

Expliquons-nous.

Un but du montage est d'éliminer les rebonds. Il suffit pour cela que le délai (période) soit **supérieur** au temps **maximum** des **rebonds ;** même si le signal comporte plusieurs changements très rapides en rafale, on n'en verra qu'un à la sortie de la première bascule. Disons que 10 ms est un maximum de durée des rebonds.

D'un autre côté, on veut enregistrer **toutes** les manœuvres du bouton-poussoir. Même si vous êtes très vif, on doute que vous puissiez l'enfoncer plus de 10 fois par seconde ! Par conséquent, un échantillonnage plus rapide que 20 fois par seconde (*) « verra » à coup sûr **toutes** les transitions.

La période de l'oscillateur peut donc être choisie entre deux limites

$$\frac{1}{20}\,s > T > \frac{1}{100}\,s$$

Pour le montage, on a pris environ 20 ms de période (1/50 s), valeur obtenue à peu de chose près avec la capacité de 1 μF et la résistance de 10 kΩ :

$$T = 1{,}7\ RC = 17\ ms$$

(*) Note aux initiés : il s'agit d'une version simpliste du fameux Théorème de Shannon.

POUR "LIRE" LES VALEURS ANALOGIQUES: DES TECHNIQUES DE COMPTAGE DU TEMPS

Pour le professeur de philosophie de monsieur Jourdain, tout ce qui n'est point prose est vers, et tout ce qui n'est point vers est prose. Les automaticiens divisent ainsi les entrées/sorties industrielles en « TOR » et « ANA » (prononcer tel quel). TOR, nous connaissons, c'est l'abréviation de « tout ou rien ». ANA, c'est tout ce qui se mesure avec plus d'un bit : les variables analogiques...

Qu'est-ce qu'une entrée analogique ?

En vérité, il n'y a pas de frontière vraie et abrupte entre les entrées/sorties dites « analogiques » et les entrées/sorties « tout ou rien ». On passe des unes aux autres de manière graduée, par des dispositions et des techniques qui dépendent des exigences de précision, de rapidité...

Prenons pour exemple les températures et imaginons d'abord une sorte de thermostat ultra-simple.

Parmi les composants à base de silicium, on trouve des capteurs appelés **thermistors.** La partie active est une sorte de céramique (SiC pour le chimiste), dont la résistance varie rapidement avec la température. Un modèle courant fera 10 kΩ à 0 °C, et perdra environ 100 Ω pour chaque °C supplémentaire.

Chaud et froid

On peut se contenter d'une échelle de températures rudimentaire **(fig. 7),** avec deux valeurs : chaud et froid, délimitées par une température-frontière.

On mesure si la température est chaude ou froide avec un simple comparateur. Un premier pont diviseur de tension comporte le thermistor : la tension au point milieu va baisser avec la montée en température et vice-versa. Si l'on repère la tension caractéristique de la température-frontière, il est aisé de construire un second pont diviseur qui maintiendra cette tension sur l'autre entrée du comparateur (*).

La sortie du comparateur donne un bit d'information seulement : s'agit-il, dans ces conditions, d'une entrée tout ou rien, ou d'une forme dégénérée de convertisseur analogique/numérique ?

Davantage de bits

Laissons tomber cette question du genre sexe des anges, pour considérer un modèle de thermomètre à peine plus compliqué **(fig. 8).**

C'est simplement le montage de la figure 7, reproduit **trois** fois pour délimiter **quatre** domaines de température. Cette sorte de convertisseur donne un code à trois bits, mais qui n'a que quatre valeurs.

Si l'on veut traduire ce code en un nombre binaire « naturel » tel qu'on les pratique dans les ordinateurs, il suffit de quelques portes logiques **(fig. 9).**

Il est évident que l'on pourrait construire des convertisseurs du même genre aussi précis que l'on veut – théoriquement ! – en augmentant le nombre de positions du **comparateur en échelle** dont on vient de voir le principe (en anglais : *comparator ladder A/D).*

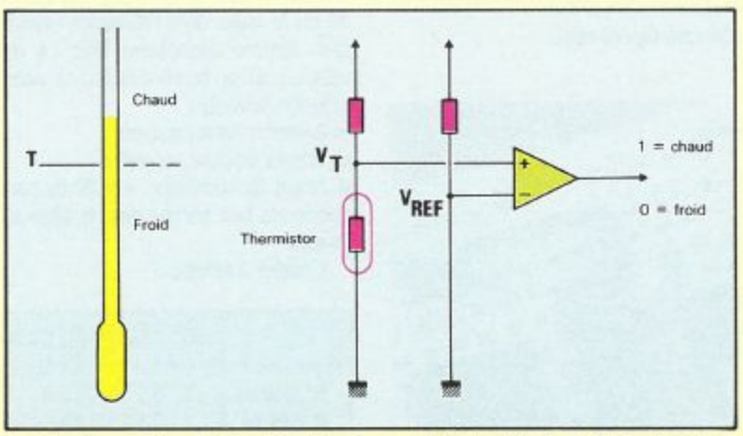


Fig. 7. – Thermomètre digital rudimentaire. Le seuil fixé V_{ref} délimite CHAUD et FROID. La tension V_T dépend de la résistance du thermistor traduisant la température.

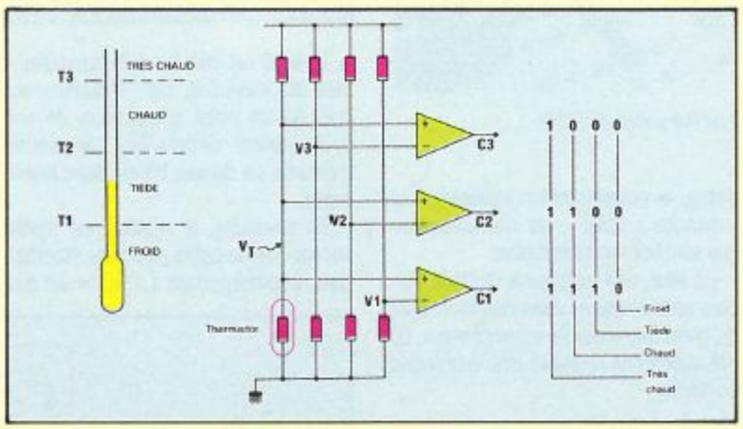


Fig. 8. – Perfectionnement du thermomètre : trois comparateurs délimitent quatre domaines de température. Le code obtenu sur 3 bits est redondant.

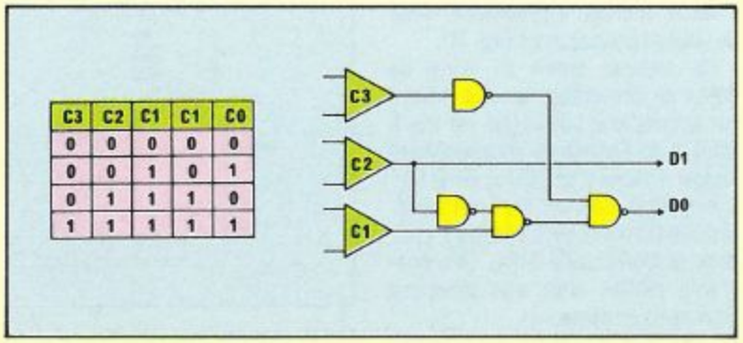


Fig. 9. – Complétés par un peu de logique, les trois comparateurs de la figure 9 donnent un code binaire « naturel ». Le montage est devenu un véritable convertisseur analogique/digital sur 2 bits.
N.B. : la rapidité de conversion est excellente !

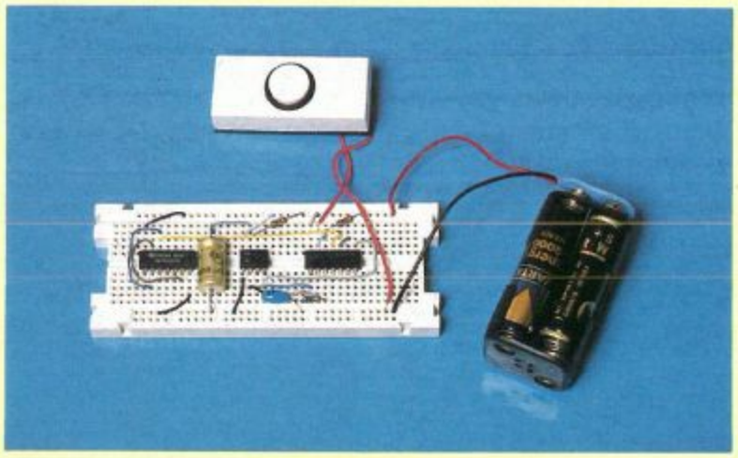
D'autres sortes de convertisseurs

Les convertisseurs à échelle de comparateur ont l'avantage de donner la valeur en sortie aussi vite que peuvent réagir les comparateurs, suivis d'une logique d'encodage. C'est le principe des convertisseurs « flash ».

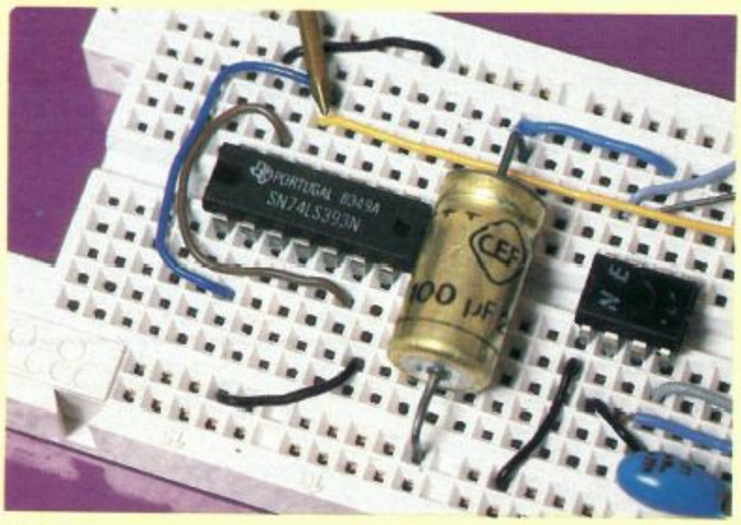
Beaucoup d'autres convertisseurs analogiques/numériques fonctionnent grâce à... un convertisseur numérique/analogique. Peu nous importe ici le détail de tels circuits intégrés, retenons seulement qu'ils donnent une tension en sortie qui est proportionnelle au nombre codé en binaire qui est appliqué sur leurs entrées.

Si l'on voulait mesurer la température grâce à ce composant, on le monterait de telle sorte qu'il fournisse la référence à un comparateur ; fonctionnellement, le convertisseur D/A remplace le pont diviseur de la figure 7, pour donner la **figure 10.**

Il y a diverses manières d'exploiter ce montage. L'une des plus rudimentaires consiste à **incrémenter** le nombre qui entre dans le convertisseur, « rampe » très facile à programmer, jusqu'à ce que le comparateur change d'état. Le nombre injecté dans le convertisseur à ce point est la « mesure » numérique de la température.



La réalisation du montage de conversion indirecte par comptage de temps.



Le pèse-signaux permet un relevé du compteur sur les huit sorties du 74393.

La conversion analogique/temps

Les montages sérieux d'entrées/sorties analogiques sont une spécialité fort subtile, que maîtrisent bien peu de laboratoires de niveau professionnel, et presque inaccessibles aux amateurs. Il faut en effet « dompter » des compromis entre toutes sortes d'exigences telles que :

- la rapidité d'échantillonnage/conversion ;
- la « mise à l'échelle » des signaux ;
- ne pas perturber le système observé, ou le moins possible ;
- protéger le système observateur contre le système observé (exemple : mesure de très hautes tensions) ;
- éliminer les fuites de courant par les masses : cauchemar récurrent des ingénieurs, etc.

Il existe cependant une famille de capteurs que tout un chacun peut traiter de manière satisfaisante avec des moyens techniques courants : tous ceux qui se comportent comme des **résistances variables.** Le thermistor déjà décrit, le potentiomètre solidaire d'un « manche à balai », les photorésistances sont autant d'exemples.

En effet, une résistance variable s'insère naturellement dans des montages du genre monostable ou oscillateur, où elle détermine la **durée** des impulsions en sortie.

Avec un « 555 » + un compteur

Notre montage d'expérience réunit de vieilles connaissances **(fig. 11).**

Un poussoir donne un signal de début de conversion, qui est envoyé sur le compteur LS393 (qui est mis à zéro) et sur l'entrée de déclenchement Trigger + Reset d'un 555 (cf. Fiche 5C).

Il apparaît alors sur OUT une impulsion dont la durée est d'environ 1,1 RC, donc proportionnelle à R_X. Le monostable réalise ainsi une première conversion en **durée.**

Cette impulsion libère un oscillateur obtenu par bouclage d'un trigger de Schmitt inverseur, plus précisément un NAND pris dans un boîtier 74LS132.

Les tops de cette **horloge** sont comptés par le LS393 agencé en compteur 8 bits, jusqu'à retombée de l'impulsion de durée variable dépendant de R_X.

Le compte est bon (?)

Le compte enregistré est d'évidence une **mesure indirecte** de la valeur de la résistance, puisqu'il matérialise la division :

$$\frac{\text{durée de l'impulsion}}{\text{période de l'oscillateur}}$$

et qu'à son tour la durée de l'impulsion est proportionnelle à la valeur de R.

Pour évaluer le montage, on fixe à 10 kΩ la valeur de la résistance « variable » (entrée analogique). Puis, on répète un certain nombre de fois la manipulation suivante :

- pression sur le poussoir ;
- attente de deux secondes ;
- relevé du compteur : il suffit de parcourir ses huit sorties avec le pèse-signaux.

L'auteur a obtenu :

	BIT 76543210
1re mesure	00111000
2e mesure	00111001
3e mesure	00111001
4e mesure	00111000
5e mesure	00111000

Le bit 0 est instable à la conversion : cela est inévitable, car l'oscillateur est pris en un point quelconque de son cycle quand débute/finit l'impulsion mesurée. Le dernier bit est donc inutilisable.

En revanche, le résultat est stable sur les bits de poids plus forts. Sachant que l'impulsion dure 1,1 RC = 1,1 s (à la précision des composants près) et que le comptage donne environ 56 (38 hexa), notre oscillateur a une période de l'ordre de :

$$\frac{1\,100}{56} \simeq 19{,}5 \text{ ms}$$

Amateurs de chiffres : en déduire la valeur maximum des résistances mesurables. Allergiques : retenez qu'un tel montage donne grosso modo le % dans les mesures, à condition de choisir des composants à tolérances serrées... et la bonne « gamme » !

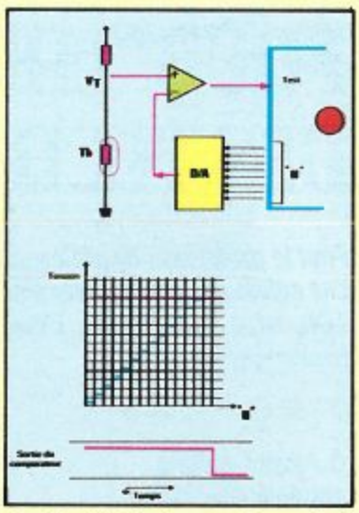


Fig. 10. – Principe des convertisseurs classiques. Un circuit de conversion digital/analogique (D/A), commandé par un processeur, fournit une référence commandée. On recherche V_T par diverses méthodes, la plus simple consistant en une « rampe » obtenue par des valeurs de commande progressant de 1 ; jusqu'à faire réagir le comparateur.

(*) En fait toutes les résistances varient avec la température ; mais beaucoup moins que le thermistor !

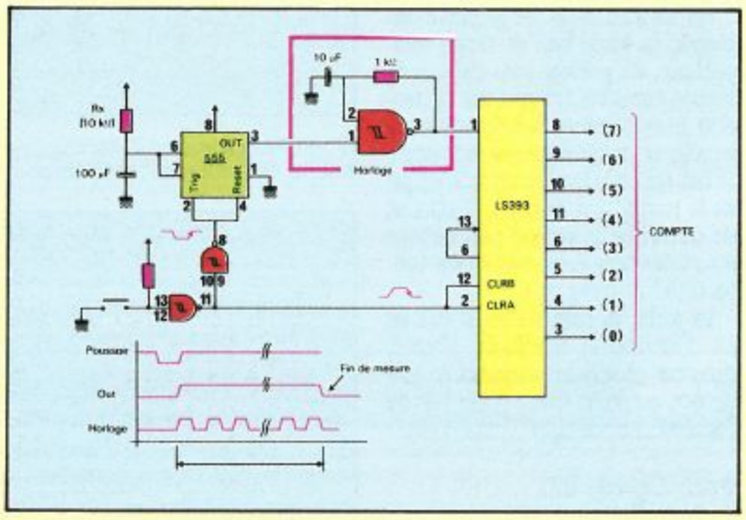


Fig. 11. – Montage expérimental de conversion indirecte par comptage de temps. Le capteur R_X détermine la largeur de l'impulsion sur OUT (montage monostable d'un 555). Le compteur LS393 est remis à zéro en même temps que débute l'impulsion. L'oscillateur d'horloge fait progresser le compteur tant que dure l'impulsion ; à la fin, l'état du compteur est une « mesure » de R_X.

MAITRISEZ LES INTERRUPTIONS

Last but not least.
Cette Fiche est la toute dernière : douze numéros de* Micro-Systèmes, *c'était notre projet initial.
Cette toute dernière fiche sera un peu particulière. Pas de montage, mais une réflexion approfondie sur les montages de qualité – hélas ! – très inégale, qui alimentent en interruptions nos processeurs favoris.

Qu'est-ce qu'une interruption ?

Question-piège typique du métier d'électronicien/informaticien. Tout le monde a sa réponse... jamais la même !

Demandez à plusieurs professionnels leur propre définition. S'il s'agit d'un programmeur-système, il vous expliquera qu'il s'agit d'un mécanisme (un peu mystérieux) qui « déroute » l'exécution sur une « routine d'interruption » quand « il se passe quelque chose dans les entrées/ sorties ». Si vous interrogez un habitué du fer à souder, il vous parlera d'un « signal d'événement » qui arrive « sur une patte du micro », etc.

Toutes ces interprétations sont complémentaires, mais pas complètes. Les procédés d'interruption sont à l'exacte frontière du logiciel et du matériel ; une description qui se veut « soft » ou « hard » exclusivement ne peut être satisfaisante à elle seule.

De la scrutation...

Un ordinateur est perpétuellement en attente **d'événements** « extérieurs », tels que la fin d'une conversion analogique/ numérique, la frappe d'une touche d'un clavier ou l'expiration d'un délai.

Si l'on n'a rien d'autre à faire dans le logiciel, on peut se contenter d'attendre un événement par une **boucle de scrutation (fig. 12).** Sous sa forme la plus simple, elle « lit » une entrée tout ou rien jusqu'à ce qu'elle soit dans l'état désiré ; il y a deux variantes à peine plus complexes :

- lecture cyclique d'une série d'entrées ;
- détection d'un événement parmi plusieurs, groupés par un « OU » logique.

Les trois schémas de base ainsi dégagés vont se retrouver dans les systèmes d'interruptions... qui ne sont en fait que des scrutations effectuées par le processeur au fil des instructions, et non par un programme !

... à l'interruption

C'est au tout début des années 1960 que l'on a commencé de « sous-traiter » au processeur matériel ce genre de travail. La scrutation cyclique existe toujours, mais elle est effectuée par le processeur **entre chaque instruction.**

Les instructions se déroulent normalement tant qu'il n'y a pas d'événement détecté sur l'une ou l'autre des entrées réservées aux signaux d'interruption. En revanche, on attend du processeur qu'il fasse le nécessaire pour que s'exécutent les instructions qui « traitent » l'événement peu après qu'il est été détecté **(fig. 13).**

Sur la quasi-totalité des microprocesseurs, le procédé consiste à appeler un **sous-programme.** Pour ce faire, le processeur range (au moins) le contenu du compteur ordinal dans une pile, puis force l'adresse du sous-programme désiré dans ce compteur.

Techniquement, il n'y a pas de véritable différence entre un CALL, JSR ou autre BSR (nom de quelques instructions d'appel de sous-programmes), et cet appel « automatique » effectué pour répondre à un signal d'interruption.

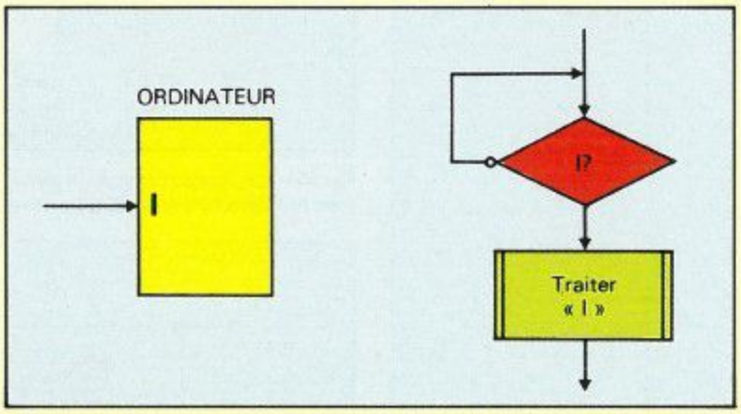


Fig. 12a. – Scrutation d'un seul signal d'« interruption » par logiciel : la boucle d'attente s'exécute jusqu'à ce que le signal I devienne actif.

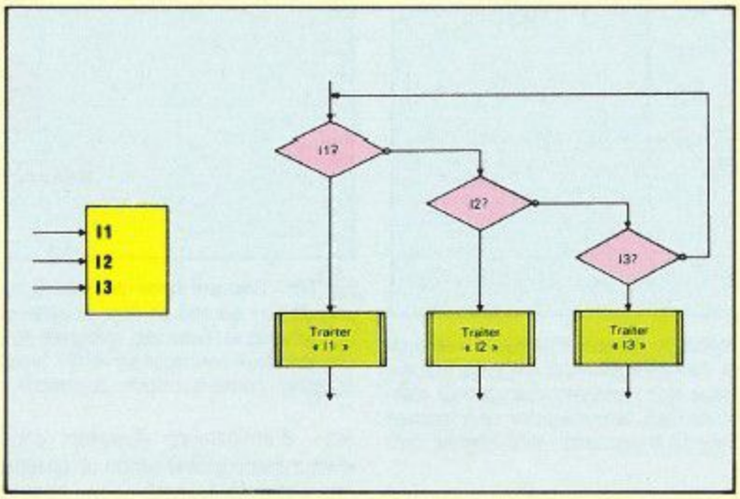


Fig. 12b. – Scrutation de plusieurs signaux : le cycle des tests se poursuit jusqu'à ce qu'un des signaux I1, I_2... soit actif. Le traitement correspondant est alors exécuté.

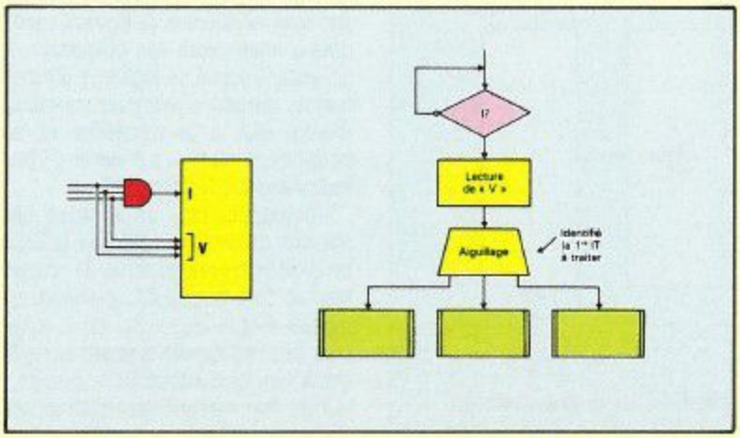


Fig. 12c. – Un seul signal I regroupe (OU) toutes les demandes, et est testé cycliquement. Si l'une des demandes au moins est active, le programme va identifier celle qu'il doit traiter (en premier) en lisant le « vecteur » V des demandes, puis en effectuant un aiguillage en fonction de V.

Première ligne de défense : disable/enable

En règle générale, le processeur dispose primitivement d'un mécanisme de défense contre les interruptions non désirées ; il existe, en effet, des circonstances où ces appels de sous-programme non contrôlés doivent être évités.

Pour ce faire, l'entrée d'interruption est barrée par un « ET », lui-même solidaire d'une bascule d'autorisation/ interdiction. Des instructions habituellement nommées EI *(Enable Interrupt)* et DI *(Disable Interrupt)* permettent au logiciel de faire barrage aux interruptions, ou au contraire de les autoriser **(fig. 14).** Au lieu de « faire barrage » on dit : masquer.

On parle d'interruption **non masquable** lorsqu'il n'existe aucun moyen **intégré** au processeur pour interdire le déroulement de programme correspondant. En fait, on peut toujours en refaire une interruption dûment masquable... en reproduisant à l'extérieur une logique similaire !

Une hiérarchie de masques

Les microprocesseurs actuels, c'est-à-dire l'essentiel des « unités centra-

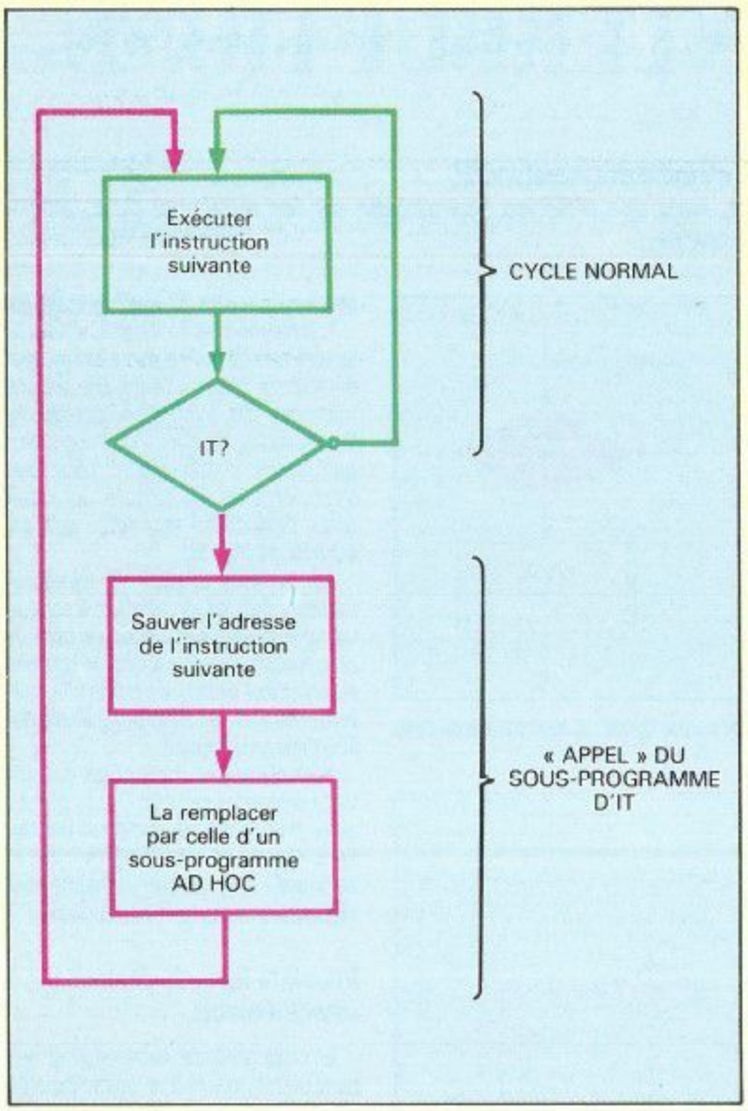


Fig. 13. – La scrutation des demandes est sous-traitée au processeur (véritables « interruptions »). Les programmes s'exécutent normalement, instruction par instruction, jusqu'à ce que le processeur détecte une demande valide. Il effectue alors l'équivalent d'un appel de sous-programme pour que l'exécution se poursuive avec les instructions adéquates. Le programme interrompu sera repris par un retour à l'adresse sauvée par le processeur ; habituellement, dans une pile.

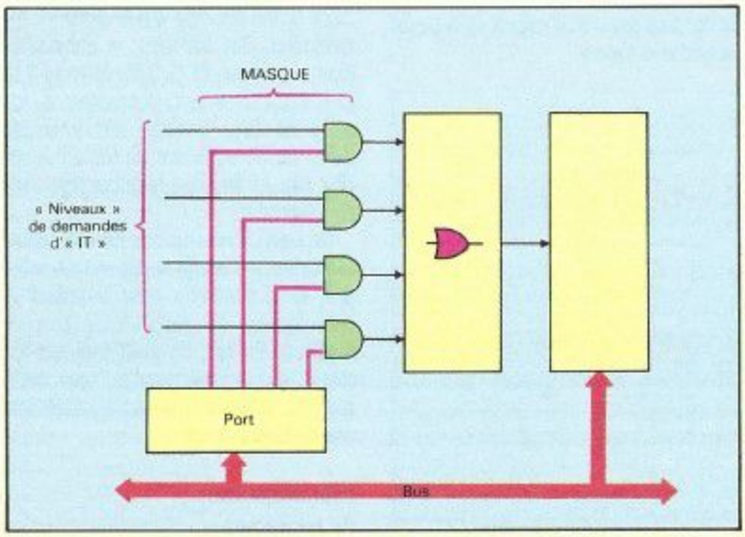


Fig. 15. – Les demandes d'interruptions sont en général hiérarchisées en « niveaux ». Un ensemble de portes commandées par un registre (port) en sortie permet de ne prendre en compte qu'un sous-ensemble choisi des demandes en instance.

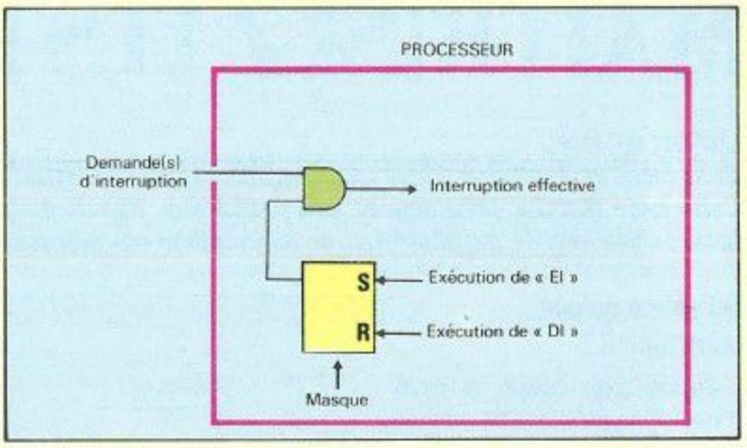


Fig. 14a. – Le processeur dispose en général d'un dispositif interne pour interdire les interruptions, et d'instructions spécialisées pour les autoriser ou les bloquer.

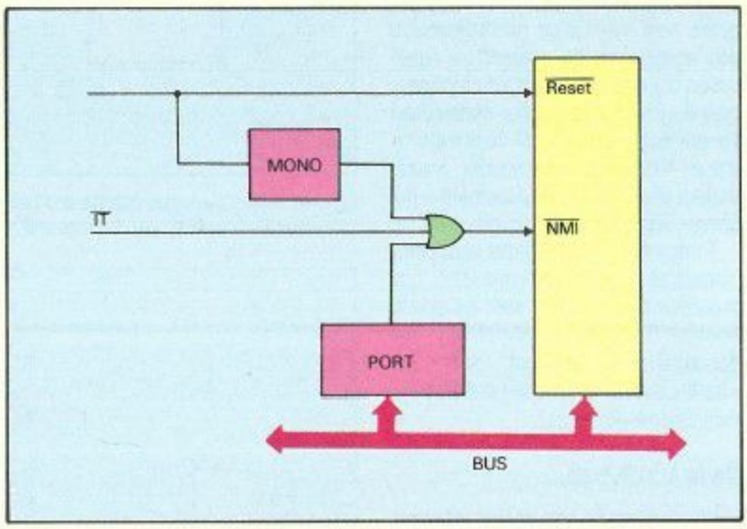


Fig. 14b. – Comment transformer une interruption « non masquable » en interruption ordinaire... prenant pour exemple un 68xx. Le signal d'interruption est « barré » par un OU (il s'agit de signaux actifs au niveau bas), commandé par un port.
Un monostable commandé par RESET interdira ces interruptions le temps nécessaire pour que le logiciel « prenne le contrôle » du montage.

les » d'ordinateurs, disposent donc d'un masque **global** contre un **ensemble** d'interruptions qui se concentrent sur une seule entrée en « OU ».

Plus un ordinateur est équipé en organes d'entrées/sorties, plus nombreuses sont les sources de signaux candidats à interrompre son processeur. A un instant donné, le logiciel a d'excellentes raisons d'autoriser certaines d'entre elles à se manifester et, au contraire, d'interdire à d'autres d'interférer avec son déroulement.

Techniquement, on masque ces sources de manière **sélective** grâce à un registre monté en sortie du processeur et dont chaque bit commande un **niveau** d'interruption **(fig. 15).** L'usage veut que l'on appelle **masque** ce registre-là ; en cas d'ambiguïté, on précise :

- que l'on masque **les** interruptions collectivement (par un barrage interne le plus souvent), ou
- que l'on masque **telle** interruption, grâce à un circuit généralement distinct du processeur proprement dit.

Il s'agit en somme de désigner la première ou la seconde ligne de défense...

Pour les sorties, surtout

Quand on lit un ouvrage d'initiation et qu'il explique les interruptions, l'exemple choisi est habituellement celui d'un organe d'entrée tel qu'un récepteur branché sur une ligne de transmission. Il est expliqué qu'à chaque fois qu'un caractère est reçu, le circuit récepteur fait une demande d'interruption (signal « j'ai reçu »). Le processeur est censé exécuter un sous-programme pour copier ce caractère en mémoire, libérant ainsi le récepteur pour une nouvelle opération **(fig. 16).**

L'exemple est bien choisi au plan pédagogique, car il est simple ; bien plus simple que le sens de transmission inverse !

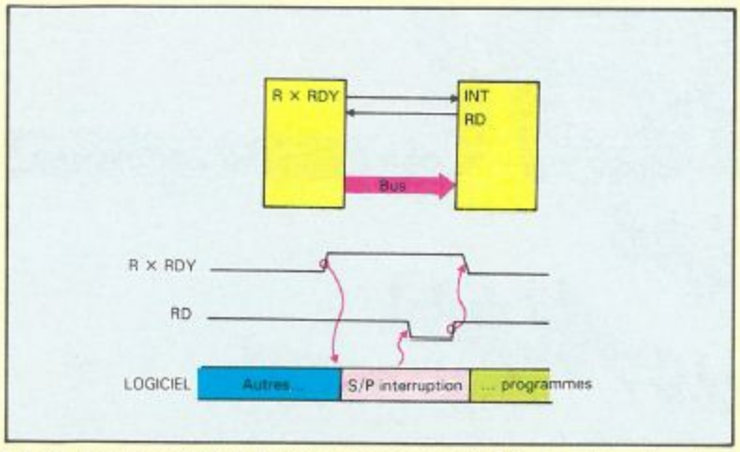


Fig. 16. – Déroulement typique d'un dialogue processeur/périphérique sur interruption : ici, un récepteur asynchrone (UART) signale l'arrivée d'un caractère en levant un signal caractéristique (RxRDY). Celui-ci active un sous-programme d'interruption qui « lit » (RD) le caractère via le bus : ce qui annule la demande d'interruption.

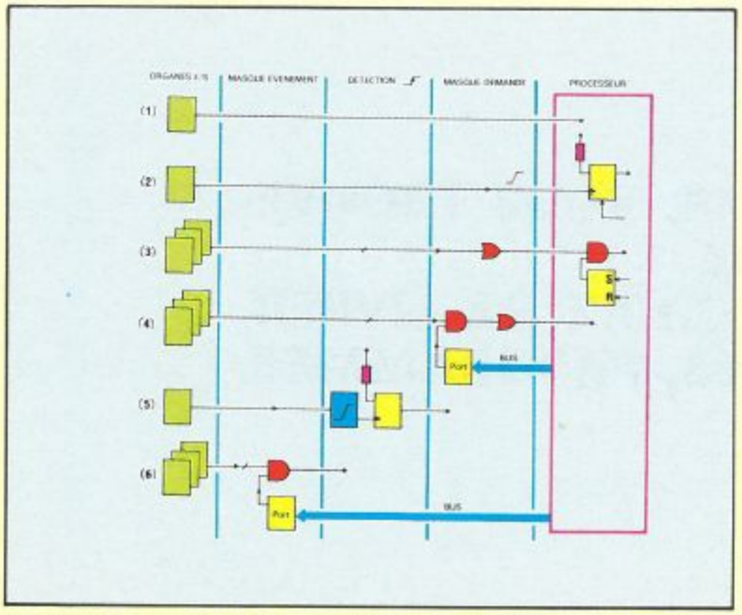


Fig. 19. – Cette figure représente (séparément) les différents dispositifs que l'on trouve dans les montages d'interruption. Dans un système donné, ils sont en général combinés et/ou pas tous présents.
(1) Liaison d'interruption simple, un état constitue la demande d'interruption.
(2) Grâce à une cellule de détection, le processeur enregistre des transitions ; mémorisées, elles deviennent demandes d'interruption.
(3) Le processeur dispose d'un masque global des demandes d'interruption regroupées sur une ligne unique. Des instructions permettent d'autoriser/interdire les interruptions en question.
(4) Par l'intermédiaire d'un port extérieur (ou d'un circuit spécialisé), le processeur peut sélectivement valider les niveaux d'interruption qui peuvent être pris en compte à un instant donné.
(5) Présence de détecteurs de transitions, qui mémorisent les événements fugitifs pour en faire des demandes d'interruption durables.
(6) Barrage en amont des détecteurs de transition, pour que le processeur puisse choisir les instants où les événements sont pris en charge, et les périodes où il sera « aveugle ».
Certains processeurs (ex. : 8085) intègrent plusieurs des fonctions décrites comme « extérieures » : masques individuels, détecteur de transition...

En effet, l'état de repos d'un récepteur, c'est « je n'ai rien reçu » ; il n'est donc pas demandeur d'interruption s'il n'y a pas lieu d'exécuter des instructions pour lui. Parfait !

Dans l'autre sens – l'émission s'il s'agit du sens inverse pour une ligne de transmission –, le signal d'interruption signifie « je suis prêt à émettre le caractère suivant ». De deux choses l'une :

- ou bien le logiciel a effectivement un caractère en attente, et le signal en question est le bienvenu ;
- ou bien, le logiciel n'a plus rien à dire, et ce signal n'a pas lieu d'être pris en compte !

L'état de repos des organes de sortie est en général une demande d'interruption (au moins latente) ; c'est dans ce cas surtout qu'il y a lieu d'employer le masque.

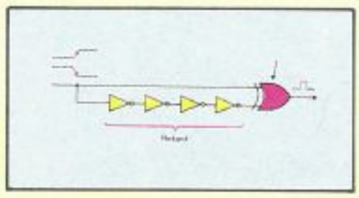


Fig. 17 a. – Le même signal d'entrée arrive dans un comparateur OU-EXCLUSIF directement, d'une part, et avec un retard de traversée de plusieurs portes, d'autre part. Il apparaît en sortie une brève impulsion de durée égale à ce retard : les deux entrées du OUX sont en effet fugitivement différentes.

Etat ou événement ?

Continuons de cheminer du processeur vers les organes d'entrées/sorties.

En toute rigueur, seuls importent au logiciel les **événements,** c'est-à-dire des **changements** qui interviennent dans son environnement :

- « C'est l'heure d'allumer le moteur-fusée du second étage », dit le temporisateur d'Ariane.
- « J'ai fini de convertir telle variable analogique », dit le convertisseur.
- « On a frappé le clavier », dit l'encodeur, et ainsi de suite.

En termes de signaux logiques, un événement c'est une **transition ;** toute transition peut être « normalisée » par un circuit très simple, qui produit une brève impulsion quel que soit le sens de cette transition **(fig. 17).** Ou bien, on peut privilégier un sens déterminé de transition.

Certains événements se traduisent par un signal qui se stabilise naturellement. Sauf restauration, un circuit émetteur « prêt à émettre » le restera tant qu'on ne lui confiera pas un caractère à transmettre. La demande d'interruption est utilisable en tant qu'**état,** c'est-à-dire qu'elle peut être présentée telle quelle devant le masque.

D'autres événements sont par essence **fugitifs.** Par exemple, une cellule photoélectrique détectant le passage d'un objet dans son champ ne restera active que pendant ce passage. Si l'on veut que le logiciel prenne en compte cette sorte d'événement même s'il était « inattentif » au moment précis où il se manifestait, on introduit entre le détecteur et le processeur un étage logique qui **mémorise** la transition voulue **(fig. 18).**

Encore une ligne de défense !

Au point où nous en sommes, on peut imaginer une ligne de défense supplémentaire, destinée à éliminer sélectivement les événements que le logiciel juge inintéressants pendant une certaine période. C'est l'affaire d'un registre ad hoc et de portes montées en « barrage », de telle sorte que l'on a pour les logiques d'interruption plusieurs étages de circuits :

- mise en forme/mémorisation,
- masquage individuel,
- masquage collectif.

Dans les machines que vous rencontrerez, vous ne les trouverez en général pas tous. Mais tous existent dans telle ou telle !

Le schéma de la dernière figure **(fig. 19)** n'a pas la prétention d'être un modèle « universel » des logiques d'interruptions. Cependant, il constitue une sorte de grille pour déchiffrer les spécifications – rarement limpides, hélas ! – avec lesquelles doivent se débrouiller les hommes du logiciel.

Amis programmeurs, un petit coup d'œil aux schémas peut vous faire gagner des heures et des heures d'essais et d'erreurs frustrants... Vous en avez maintenant les moyens !

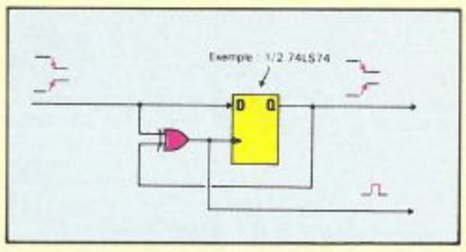


Fig. 17b. – Autre détecteur de transition très usité : l'apparition en entrée d'une valeur différente de celle mémorisée dans la bascule D force une transition via le OUX... qui enregistre dans la bascule la nouvelle valeur.

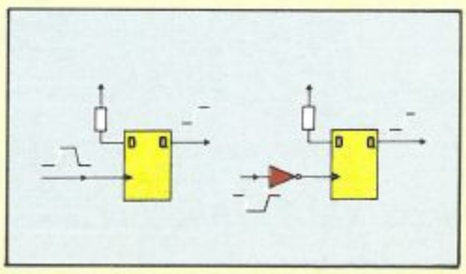


Fig. 18. – Mémorisation d'une transition dans un sens déterminé, par une bascule D commandée par son flanc habituel, ou par l'opposé en passant par un inverseur. Ces montages impliquent une liaison de remise à zéro pour enregistrer une nouvelle transition (non représenté).