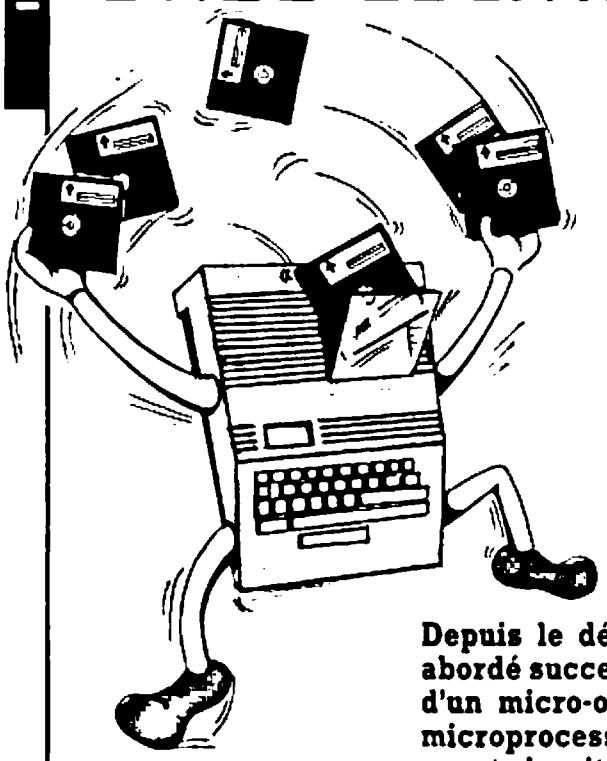
L'ABC DE LA MICRO-INFORMATIQUE



LES CIRCUITS D'INTERFACE

Depuis le début de cette série, nous avons abordé successivement la structure générale d'un micro-ordinateur, la présentation d'un microprocesseur, les divers types de mémoires et circuits programmables. Il nous reste à vous parler d'une famille très étendue, constituée par les circuits d'interface.
Comme nous l'avons vu tout au début de cette série, ces circuits permettent à tout micro-ordinateur de communiquer avec le monde extérieur, que ce dernier soit un opérateur humain ou un appareil particulier tel qu'un lecteur de disquettes, par exemple.
La très grande diversité des circuits d'interface nous a conduit à faire des choix et à nous limiter aux familles principales. Néanmoins, le panorama que nous allons brosser à compter d'aujourd'hui sera assez complet pour couvrir la majorité des applications.

GENERALITES

Supposons que l'on ait à relier un micro-ordinateur à plusieurs équipements tels que terminaux, imprimantes, etc. La première idée qui vient à l'esprit est d'utiliser des liaisons de type parallèle, c'est-à-dire des liaisons comportant un grand nombre de fils et véhiculant, d'un seul coup, un octet. En effet, nous avons vu que l'information traitée dans les micro-ordinateurs était très souvent organisée en octet, et il semble donc logique de la transmettre sous cette forme.
Pratiquement, ce n'est malheureusement pas aussi simple car, d'une part, il va falloir utiliser un grand nombre de fils (8 pour un octet + des fils de contrôle de la liaison), ce qui va très vite devenir coûteux dès que la liaison va devoir dépasser quelques mètres. D'autre part, les signaux utilisés dans les micro-ordinateurs sont des signaux logiques TTL qui doivent respecter des normes très strictes et, en particulier, qui doivent avoir des fronts de montée et de descente très raides. De tels signaux s'accommodent fort mal des capacités parasites des câbles, capacités qui sont d'autant plus importantes que les câbles sont plus longs.
Toutes ces constatations ont conduit les informaticiens, il y a de nombreuses années de cela, à définir un autre type de liaison : la liaison série. Cette dernière est vite devenue un standard, sous une de ses formes particulières tout du moins, et on la rencontre maintenant partout, que ce soit en micro-informatique ou en informatique classique. Nous allons donc commencer notre étude par cette dernière qui, comme vous allez le constater, permet de faire connaissance avec bon nombre de notions nouvelles.

PRINCIPE D'UNE LIAISON SERIE

Le principe de base est très simple et repose sur des circuits logiques connus depuis de nombreuses années : les registres à décalage. Si on regarde la figure 1, on peut y voir un registre à décalage à entrée parallèle et sortie série. Le fonctionnement d'un tel circuit est très simple et se trouve schématisé sur cette même figure. Des données, ici des mots de 8 bits, bien sûr, sont appliquées aux entrées parallèles du circuit, qui les délivre ensuite sous forme série sur la sortie du même nom, au rythme de l'horloge fournie au registre. On retrouve donc sur un seul fil nos 8 bits les uns derrière les autres.
Ce procédé permet donc de faire voyager, sur un seul et unique fil, autant de bits de

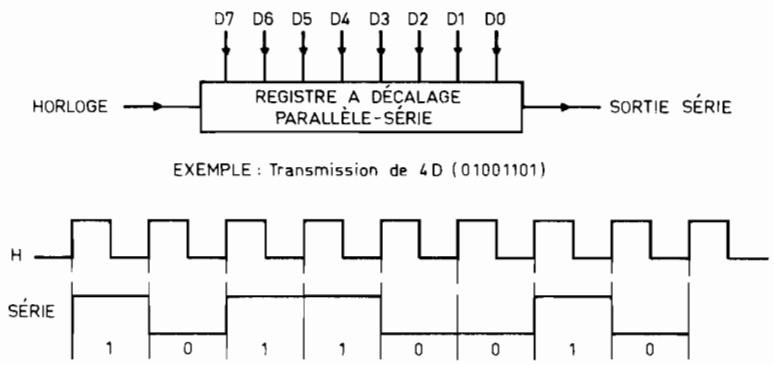


Fig. 1. – Utilisation d'un registre à décalage pour émettre des données sous forme série.

données qu'on le désire ; il suffit de disposer d'un registre à décalage de taille adéquate. Dans le cas qui nous occupe dans le cadre de cette étude, il nous faut faire voyager des mots de 8 bits, ce qui implique l'emploi de registres à décalage parallèle – série à 8 entrées, qui sont des composants très classiques.

Puisque nous savons maintenant envoyer des données sous forme série, reste à voir comment faire l'opération inverse côté réception, c'est-à-dire comment passer à nouveau de parallèle en série. La solution est simple et fait appel à un circuit logique connu : le... registre à décalage. Mais oui, c'est le même type de circuit que celui utilisé à l'émission qui va servir à la réception ; mais, ici, il sera du type registre à entrée série et sorties parallèles. La figure 2 montre son synoptique d'emploi qui se passe de commentaire.

Pour que cela fonctionne, il est bien évident qu'il faut que les horloges appliquées aux deux registres, émetteur et récepteur, aient la même fréquence mais, ce qui est plus grave, c'est qu'il faut aussi que ces horloges soient en phase. En effet, examinez la figure 3 qui montre ce qui peut se passer si les horloges ne sont plus en phase. Le registre récepteur vo transformer les données séries reçues en données parallèles, à des instants liés précisément aux fronts de son horloge. Si ceux-ci sont décalés par rapport à ceux de l'horloge d'émission, le mot de 8 bits reconstitué n'aura rien à voir avec le mot émis. La transmission sero donc inutilisable.

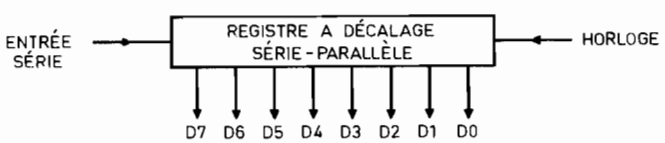
Si l'on sait faire facilement des horloges fonctionnant sur des fréquences suffisamment proches l'une de l'autre en les pilotant par quartz, il n'en est pas de même de la relation de phase. La seule solution pour satisfaire la contrainte vue ci-avant est de transmettre à la fois donnée et horloge de l'émetteur au récepteur pour que notre liaison puisse fonctionner, comme schématisé figure 4. Outre le fait que cela contraint à utiliser un fil de plus que prévu, le problème se complique dès lors que l'on veut faire une liaison bidirectionnelle, comme c'est presque toujours le cas en informatique. En effet, il va falloir véhiculer deux horloges, une dans chaque sens, puisque chaque équipement sera tout à tour émetteur ou récepteur, selon le sens du diologue établi.

Toutes ces considérations n'ont pas fait abandonner ce type de liaison série, que l'on appelle la liaison série synchrone, mais l'ont fait réserver à des applications particulières où la vitesse de transmission devait être élevée et où le surcroît de complexité et de coût entraîné par ces problèmes d'horloge était négligeable par rapport au reste du système. Sur les micro-ordinateurs et les appareils où la notion de vitesse de transfert est moins importante, un autre type de liaison série a été développé.

LA LIAISON SERIE ASYNCHRONE

Le principe adopté pour les liaisons séries asynchrones est analogue à celui décrit ci-avant, à savoir que les données transmises sont converties de parallèle en série et de série en parallèle au moyen de circuits analogues à des registres à décalage. Mais, de plus, pour résoudre les problèmes de synchronisation d'horloge que nous venons d'évoquer, deux informations supplémentaires sont ajoutées à chaque mot de huit bits transmis. Examinons la figure 5 sur laquelle nous avons représenté l'état d'une ligne de transmission série asynchrone. Au repos, c'est-à-dire en l'absence de transmission, la ligne est au niveau logique haut. Avant l'émission du premier bit du mot à transmettre, la ligne va passer au niveau 0 pendant une période de l'horloge de transmission ; ce passage à 0 représente le bit de début de mot, ou bit de « start ». Le mot à transmettre est ensuite émis normalement tel qu'il est fourni par un registre à décalage, comme nous

Fig. 2. – Un registre à décalage est également utilisable pour recevoir des données sous forme série.



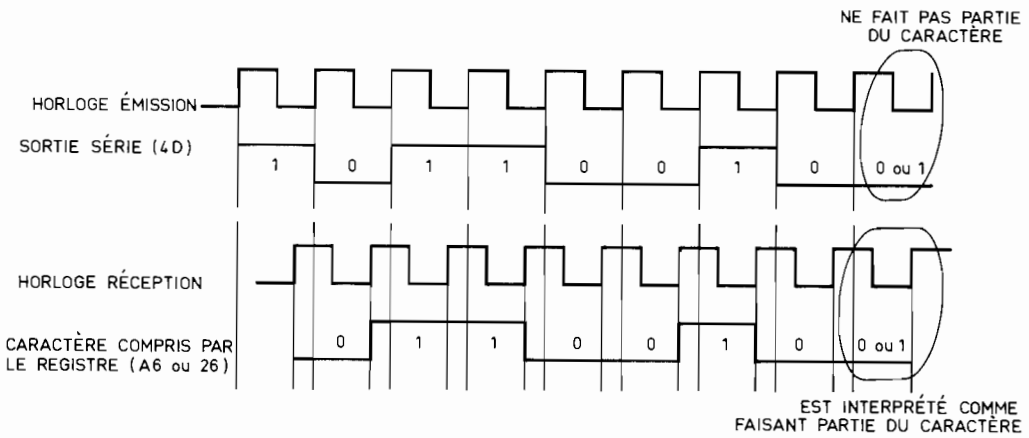


Fig. 3. – Les effets néfastes d'un décalage en phase des horloges d'émission et de réception.

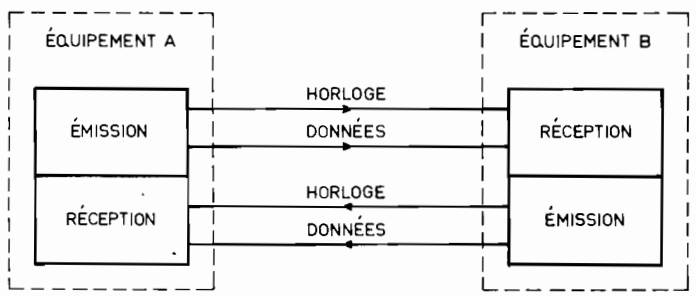


Fig. 4. – La transmission des horloges conduit à utiliser plusieurs fils.

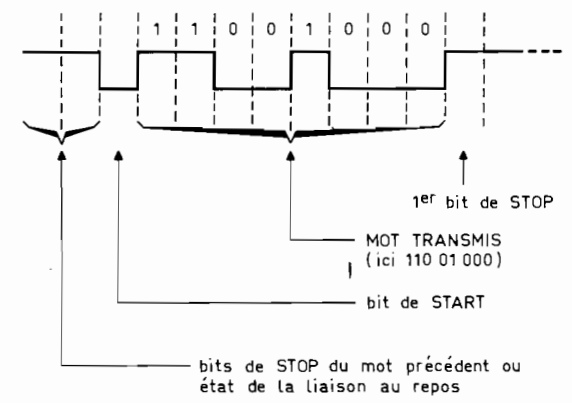


Fig. 5
Aspect d'un caractère transmis en mode asynchrone.

l'avons vu ci-avant. Après le dernier bit de ce mot, la ligne passe à nouveau à l'état haut pendant une ou deux périodes de l'horloge de transmission, afin de constituer le ou les bits de « stop » (un ou deux selon le format choisi, comme nous le verrons ultérieurement).

Toute transmission d'un mot de huit bits par une liaison série asynchrone nécessite donc deux ou trois bits supplémentaires : un bit de start qui précède le mot « utile » et un ou deux bits de stop qui suivent le mot « utile ».

Ces bits particuliers sont exploités dans le circuit de réception qui, vous le concevez bien, ne peut plus être un simple registre à décalage. En effet, ce circuit sait, lors de la réception du bit de start, recaler son horloge locale avec celle utilisée à l'émission, et il peut donc ensuite recevoir parfaitement les bits « utiles » qui suivent. Le synchronisme en phase indispensable dans la liaison série synchrone a donc été rendu inutile ici. Par contre, bien sûr, il faut toujours que les horloges d'émission et de réception soient à la même fréquence. Il n'est cependant pas nécessaire que ces fréquences soient rigoureusement identiques car, du fait du recalage réalisé pour chaque bit de start et, donc, pour chaque caractère reçu, une différence de quelques pour cent est admissible. Comme le moindre oscillateur à quartz fait mieux que ça, même dans le pire des cas, il n'est plus nécessaire de véhiculer d'horloge dans une liaison série asynchrone : chaque équipement possède la sienne propre.

Ce procédé de liaison permet donc de relier facilement deux équipements informatiques quelconques puisqu'il suffit de trois fils : un fil par sens de transmission et une masse. La

seule contrainte à respecter est que chacun des équipements puisse travailler avec la même fréquence d'horloge.

Pour ce faire, et pour éviter que la plus complète pagaille ne règne, des vitesses de transmission ont été définies et normalisées. Comme la seule entité réellement visible sur une liaison de ce type est le mot de 8 bits transmis, que l'on appelle dans ce cas un caractère (nous verrons pourquoi dans un instant), les vitesses ont été définies en premier lieu en caractères par seconde. Les valeurs les plus courantes à l'heure actuelle sont 10, 30, 60, 120, 240, 480, 960 et 1 920 caractères par seconde. Ce ne sont pas les seules normalisées, mais ce sont les plus fréquentes sur le matériel informatique classique (les minitels se distinguent avec une liaison à 120 caractères par seconde dans un sens et à 7,5 caractères par seconde dans l'autre, mais nous verrons pourquoi ultérieurement).

Comme chaque caractère transmis nécessite en fait 10 bits (8 bits « utiles » + 1 start + 1 stop), on parle aussi de vitesses de transmission en bits par seconde ou en bauds dont les valeurs normalisées sont alors les multiples par 10 des valeurs vues ci-avant. Ainsi, une transmission à 960 caractères par seconde s'appelle aussi une transmission à 9 600 bits par seconde ou à 9 600 bauds. Si vous êtes observateur, vous pourrez nous faire remarquer que, si l'on utilise deux bits de stop et non pas un, les caractères transmis comportent 11 bits, et que la règle ci-avant devient donc fausse. C'est exact, mais ce cas de 11 bits n'existe quasiment que pour les liaisons à 10 caractères par seconde, qui sont alors des liaisons à 110 bauds ou 110 bits par seconde. Aux autres vitesses, ce sont toujours 10 bits par caractère qui sont utilisés.

UN CODE PRESQUE UNIVERSEL

Maintenant que nous savons relier deux équipements et qu'ils sont capables d'échanger des informations, il faut nous intéresser à la signification de ces dernières. En effet, si le codage sur 8 bits des chiffres de 0 à 9 et des lettres de A à F ne pose pas de problème en utilisant, par exemple, la notation hexadécimale présentée au début de cette série d'articles, nous ne savons pas coder autre chose. Si vous souhaitez relier un micro-ordinateur qui fait du traitement de texte à une imprimante, il faut pourtant bien pouvoir lui envoyer toutes les lettres, chiffres et signes utilisés habituellement en dactylographie. Un code a donc été mis sur pied pour ce faire, il y a déjà de nombreuses années, code presque quasi universellement adopté pour ce type de liaison. Il a pour nom le code ASCII (prononcez aski), ce qui signiifie American Standard Code for Information Interchange, c'est-à-dire code américain standard pour l'échange d'informations.
Ce code, dont le tableau officiel vous est présenté figure 6, n'utilise que 7 bits et permet donc de coder 128 signes différents. C'est plus que suffisant pour l'application envisagée puisque, outre toutes les lettres, majuscules et minuscules, tous les chiffres et tous les symboles que l'on rencontre sur un clavier de machine à écrire électrique, il permet aussi de coder un certain nombre de caractères dits de contrôle. Ceux-ci sont regroupés dans les basses valeurs de code (de 00 à 1F), reçoivent un nom à deux ou trois lettres et ont des significations particulières, telles que : le retour chariot, le saut de ligne avant et arrière, le saut de page, le retour arrière d'un caractère, etc.
Le tableau du code ASCII appelle quelques autres commentaires.

Bits 7, 6, 5					000	001	010	011	100	101	110	111
Bits 4	3	2	1	Hex 1	Hex 0	0	1	2	3	4	5	6	7
0	0	0	0	0		NUL	DLE	SP	0	@	P	'	p
0	0	0	1	1		SOH	DC1	/t !	1	A	Q	a	q
0	0	1	0	2		STX	DC2	"	2	B	R	b	r
0	0	1	1	3		ETX	DC3	#	3	C	S	c	s
0	1	0	0	4		EOT	DC4	$	4	D	T	d	t
0	1	0	1	5		ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6		ACK	SYN	&	6	F	V	f	v
0	1	1	1	7		BEL	ETB	'	7	G	W	g	w
1	0	0	0	8		BS	CAN	(	8	H	X	h	x
1	0	0	1	9		HT	EM	)	9	I	Y	i	y
1	0	1	0	A		LF	SUB	*	:	J	Z	j	z
1	0	1	1	B		VT	ESC	+	;	K	[	k	{
1	1	0	0	C		FF	FS	,	<	L	\	l	
1	1	0	1	D		CR	GS	-	=	M	]	m	}
1	1	1	0	E		SO	RS	.	>	N	¬↑∧	n	~
1	1	1	1	F		SI	US	/	?	O	←	o	DEL

Fig. 6. – Tableau du code ASCII.

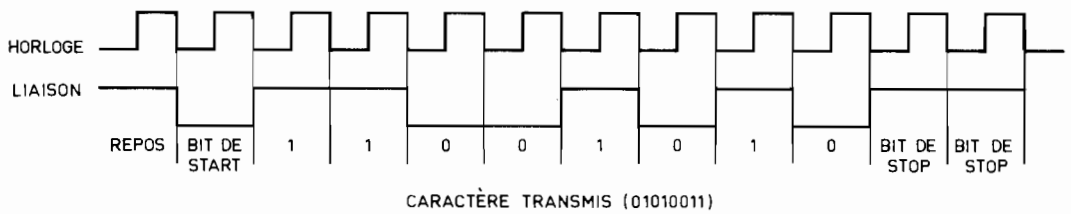


Fig. 7. – Représentation d'un caractère codé en ASCII dans une transmission série asynchrone avec bit de parité.

• Les codes de contrôle dont nous venons de parler sont groupés de 00 à 1F.

• Les chiffres de 0 à 9 sont codés respectivement de 30 à 39.

• Les lettres de A à Z sont codées à partir de 41 avec une incrémentation de 1 par lettre, en respectant l'ordre alphabétique.

• De même, les lettres a à z partent de 61.

• Il y a une différence de code de 20 entre une lettre majuscule et son équivalent minuscule (B vaut 42, alors que b vaut 62).

• Les caractères accentués français ne figurent pas dans ce tableau, pas plus que certains caractères spéciaux tels que les accents circonflexes, le œ collé, le c cédille, etc.

C'est normal, vu l'origine américaine du code, puisque ces caractères-là n'existent pas dans la langue de Lincoln.

Ce manque de caractères français a fait définir à certains constructeurs un code ASCII dit étendu, appellation bien vite reprise par une foule de journalistes soi-disant techniques.

Une telle définition n'a aucun sens à l'heure actuelle. Le code ASCII se limite au tableau de la figure 6 ; toutes les extensions qui peuvent être définies (et qui sont d'ailleurs différentes chez le fabricant d'imprimantes Epson et chez IBM, pour ne citer qu'eux) ne sont que des cas particuliers, qui n'ont aucune valeur de norme.

UN BIT DE TROP

Nous vous avons parlé, jusqu'à maintenant, de transmissions de mots de 8 bits « utiles », ce qui est logique puisque les micro-ordinateurs manipulent des octets, et nous venons de voir que le code ASCII n'utilise que 7 bits. Il y a donc un bit de trop dans notre octet.

Ce bit peut être ou ne pas être utilisé. S'il ne l'est pas, il est mis à n'importe quel niveau (0 ou 1) par l'émetteur, et est ignoré par le récepteur. Par contre, il peut être utilisé pour effectuer un contrôle de qualité de la transmission sous forme de bit de parité.

Cette parité peut être paire ou impaire et se calcule de la façon suivante : en parité paire, le bit de parité est mis à 1 si le nombre de bits à 1 du caractère transmis est pair. Il reste à 0 dans le cas contraire. En parité impaire, le bit de parité est mis à 1 si le nombre de bits à 1 du caractère transmis est impair. Il reste à 0 dans le cas contraire. A titre d'exemple, la figure 7 montre le codage du caractère S en ASCII avec parité impaire. Remarquez que le calcul de la parité ne prend en compte que les bits de la partie « utile » du caractère transmis. Le bit de start, le bit de parité lui-même et le ou les bits de stop sont ignorés par le calcul.

L'utilisation de la parité se passe de la façon suivante : l'émetteur du caractère calcule la parité et ajoute celle-ci en tant que 8e bit à chaque caractère émis. Le récepteur calcule, lui aussi, la parité sur chaque caractère reçu et compare ce qu'il trouve avec la parité qu'il a reçue. Si elles sont différentes, il y a nécessairement eu une erreur de transmission.

Il est évident, en revanche, que, si les parités sont identiques, cela ne signifie pas nécessairement que la transmission s'est bien passée. En effet, deux bits peuvent avoir changé d'état simultanément, ce qui ne fait pas générer d'erreur de parité. Un tel contrôle est cependant acceptable car on considère qu'en cas d'absence quasi permanente d'erreur de parité tout se passe bien.

C'est valable dans 99 des cas, car les erreurs qui se compensent sont tout de même assez rares.

CONCLUSION

Nous en resterons là pour aujourd'hui, car il y a beaucoup à dire à propos de ces liaisons séries asynchrones. Nous avons cependant bien progressé depuis le début de cet article, puisque nous savons maintenant connecter deux équipements informatiques quelconques, les faire dialoguer et effectuer un contrôle sommaire de la qualité de la liaison.

C. TAVERNIER