MANUEL

micro-ordinateur

Commodore

64

commodore COMPUTER

COMMODORE 64

Publié par
COMMODORE INTERNATIONAL, LTD.
Computer Systems Division
950 Rittenhouse Road
Norristown, PA 19403
U.S.A.

Imprimé en Allemagne.

TABLE DES MATIERES

Annexes **Page**

INTRODUCTION

Félicitations pour votre achat de l'un des meilleurs ordinateurs du monde. Vous êtes à présent le fier propriétaire du COMMODORE 64. Commodore est connue comme une société informatique amicale et il fait partie de cette amitié de vous faciliter la lecture, l'utilisation et la compréhension des manuels d'instruction. Le MANUEL D'UTILISATION COMMODORE 64 est conçu pour vous fournir tous les renseignements dont vous avez besoin pour installer correctement votre équipement, vous familiariser avec le fonctionnement du COMMODORE 64 et permettre un départ simple et agréable dans l'apprentissage de la confection de vos propres programmes. Pour ceux qui ne souhaitent pas être tourmentés par l'apprentissage de la programmation, nous avons placé tous les renseignements dont vous avez besoin pour utiliser les programmes Commodore ou d'autres programmes préconfectionnés et/ou cassettes de jeux (software de tiers) dès le début. Ceci signifie que vous n'avez pas besoin de parcourir la totalité du manuel pour commencer à utiliser l'appareil.

A présent, examinons certaines des caractéristiques excitantes offertes par votre COMMODORE 64. Tout d'abord, lorsque l'on en viendra aux GRAPHIQUES, vous disposerez du synthétiseur d'images le plus avancé de l'industrie des micros-ordinateurs. Nous l'appelons SPRITE GRAPHICS et il vous permet de concevoir vos propres images dans 4 couleurs différentes, tout comme celles que vous connaissez des jeux vidéo. Et de plus le SPRITE EDITOR vous permet d'animer simultanément jusqu'à 8 différents niveaux d'image. Vous pouvez déplacer vos créations où que ce soit sur l'écran et même passer une image devant ou derrière une autre. Votre COMMODORE 64 dispose même d'une détection de collision automatique qui donne à l'ordinateur l'instruction d'entreprendre la manoeuvre que vous souhaitez lorsque les sylphes se heurtent les uns contre les autres.

Ensuite, le COMMODORE 64 possède des effects musicaux et sonores incorporés qui peuvent rivaliser avec nombre de synthétiseurs musicaux bien connus. Cette partie de votre ordinateur vous fournit trois voix indépendantes, chacune sur une gamme de type piano complète à 9 octaves. D'autre part, vous disposez de 4 formes de signaux différentes (dent de scie, triangle, impulsion variable et bruit), d'un générateur ADSR programmable (montée, descente; appui, relâchement), d'un générateur d'enveloppe, de filtres passe-haut, passe-bas et passe-bande programmables pour chaque voix et de commandes de volume et de résonnance variables. Le COMMODORE 64 vous permet de raccorder votre sortie

audio à la plupart des systèmes d'amplification de haute qualité si vous souhaitez jouer votre musique sous forme d'une reproduction sonore professionnelle.

Pendant que nous sommes sur le sujet de la connexion du COMMODORE 64 sur d'autres équipements . . . votre système peut être étendu en ajoutant des accessoires intitulés périphériques au fur et à mesure de la croissance de vos besoins de calcul. Certaines de vos options comprennent des appareils,tels qu'un enregistreur DATASSETTE* ou jusqu'à 5 unités de stockage et d'entraînement de disques VIC 1541 pour les programmes que vous constituez et/ou passez. Si vous disposez déjà d'un entraîneur de disques VIC 1540, votre revendeur peut l'adapter pour qu'il soit utilisé avec le COMMODORE 64. Vous pouvez ajouter une imprimante à matrices VIC pour vous fournir des exemplaires imprimés de vos programmes, lettres, factures, etc. Finalement, si vous êtes l'une de ces personnes intéressées par la grande variété du software d'application disponible en CP/M**, le COMMODORE 64 peut être équipé d'un microprocessor enfichable Z-80.

Le fait que ce manuel d'utilisation contribuera à développer votre compréhension des ordinateurs, est tout aussi important que le matérial disponible. Il ne vous dira pas tout ce qui est connu sur les ordinateurs, mais il vous renverra à une grande variété de publications pour des informations plus détaillées sur les sujets présentés. Commodore vous souhaite une grande satisfaction dans l'utilisation de votre nouveau COMMODORE 64. Et pour avoir de la satisfaction, pensez-y: la programmation n'est pas le genre de chose que vous pouvez apprendre en un jour. Ayez de la patience en progressant dans la lecture du manuel d'utilisation. Avant de commencer, prenez quelques minutes pour remplir et envoyer la carte de propriétaire/d'enregistrement qui est jointe à votre ordinateur. Elle vous assurera que votre COMMODORE 64 est correctement enregistré. Bienvenue dans un tout nouveau monde de distraction!

REMARQUE:

De nombreux programmes sont en cours de développement pendant que ce manuel est sous presse. Veuillez consulter votre revendeur Commodore local qui vous tiendra au courant de l'abondance des programmes d'application existants pour le COMMODORE 64 dans le monde entier.

---

* DATASSETTE est une marque déposée de Commodore Business Machine Inc.

** CP/M est une marque déposée de Digital Research Inc. Spécifications soumises à modification.

CHAPITRE 1

MISE EN PLACE

DEBALLAGE ET BRANCHEMENT DU COMMODORE 64

Les instructions suivantes vous indiquent pas par pas comment raccorder le COMMODORE 64 à votre appareil de télévision, au système ou au moniteur et à vous assurer que tout fonctionne correctement. Avant de connecter quoi que ce soit sur l'ordinateur, contrôlez le contenu de la boîte du COMMODORE 64. En dehors de ce manuel, vous trouverez les articles suivants:

1. COMMODORE 64
2. Alimentation (coffret noir avec le cordon d'alimentation)
3. Câble d'antenne

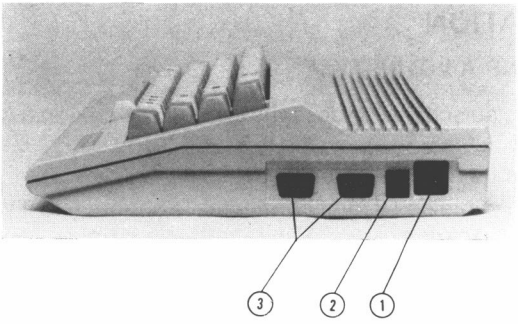
Si l'un de ces articles manque, contrôlez encore une fois sans tarder le contenu avec votre revendeur pour le compléter. En premier lieu, examinons la disposition des différentes connexions sur l'ordinateur et leur fonctionnement.

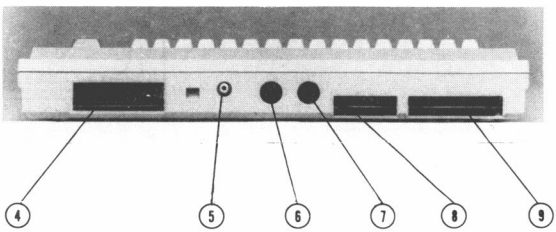
CONNEXIONS DU PANNEAU LATERAL

1. **Prise de courant.** La fiche du câble qui connecte l'ordinateur avec l'alimentation est branchée ici.
2. **Interrupteur d'alimentation.**
3. **Ports de jeu.** Chaque connecteur de jeu peut recevoir une manette, un contrôleur de jeu ou un crayon lumineux.

CONNEXIONS ARRIERES

4. **Fente pour modules.** La fente rectangulaire à gauche est destinée aux programmes ou jeux enfichables.
5. **Connecteur TV.** Possibilité de connecter l'ordinateur par l'entrée d'antenne (75 Ohm). Le signal contient le son et l'image.
6. **Sortie audio et vidéo.** Ce connecteur fournit directement le son qui peut être raccordé à un système audio de haute qualité et un signal vidéo qui peut être amené à un moniteur.
7. **Port sériel.** Vous pouvez raccorder une imprimante ou une station de disquettes simple directement au COMMODORE 64 par l'intermédiaire de ce connecteur.





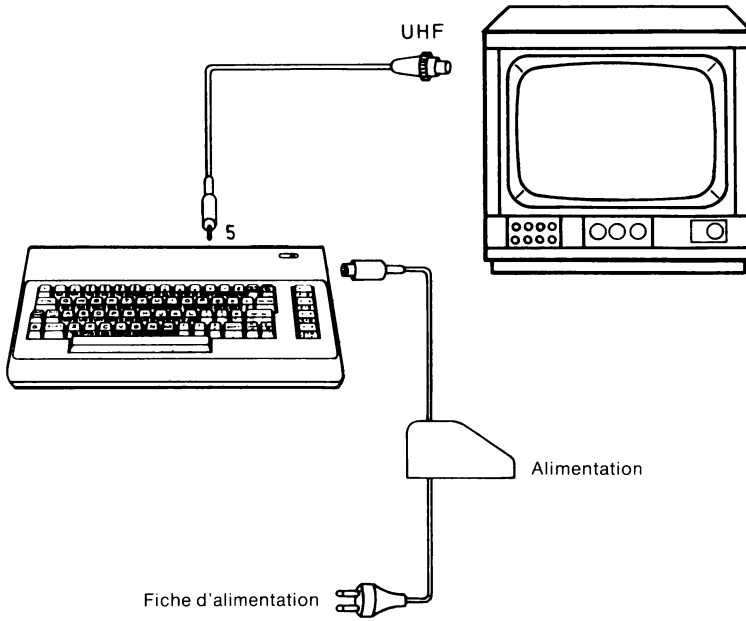
8. **Interface cassette.** Un enregistreur DATASSETTE peut être fixé à l'ordinateur de manière que vous puissiez sauvegarder les informations entrées pour les utiliser plus tard.
9. **Port utilisateur.** Entrée/sortie librement programmable et connexion pour modules enfichables tel que l'interface RS 232.

INSTALLATION

CONNEXIONS A VOTRE TV

Référez-vous au schéma ci-dessous pour la connexion à votre TV.

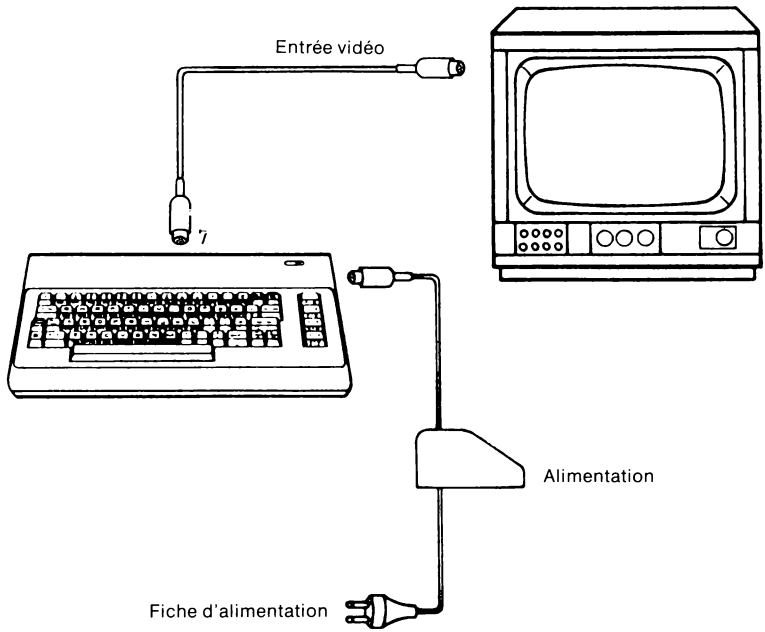
1. **Utilisation de l'entrée antenne UHF**
   A l'aide du câble de TV joint connectez le connecteur TV (no 5) au dos du COMMODORE 64 avec l'entrée antenne UHF de votre TV. Enlevez l'antenne avant. Fixez votre appareil au canal 36.



2. **Sortie vidéo**
   L'image optimale est obtenue par un moniteur. A ce fait utilisez un câble vidéo avec d'un côté une fiche de diode à 5 broches (DIN 41524) pour la connexion no 6 du COMMODORE 64 et de l'autre côté une fiche vidéo (DIN 45322) pour votre moniteur. Au cas ou votre TV dispose d'une prise vidéo, vous pouvez l'utiliser également comme moniteur, veillez seulement à ce que le connecteur vidéo de votre TV est défini comme entrée. Si on utilise un recorder vidéo cela se fait normalement par un potentiel auxiliaire de 12 V qui se branche au pin 1 de la fiche vidéo de la TV.

Votre COMMODORE 64 n'a pas ce potentiel. Faites donc installer par un spécialiste TV un commutateur adéquat. Il existe des TV ou il suffit d'introduire un pont de fil dans la fiche vidéo, du fait qu'il y a des téléviseurs qui ont prévu ce potentiel auxiliaire de 12 V dans le pin 5 de la prise vidéo. Utilisez donc cette entrée très prudemment. Le potentiel auxiliaire ne doit en aucun cas toucher les sorties de votre COMMODORE 64. L'ordinateur serait détruit immédiatement. Confiez ce travail seulement à un spécialiste.

D'habitude vous devez enlever le câble vidéo si vous recevez des émissions de TV normales.



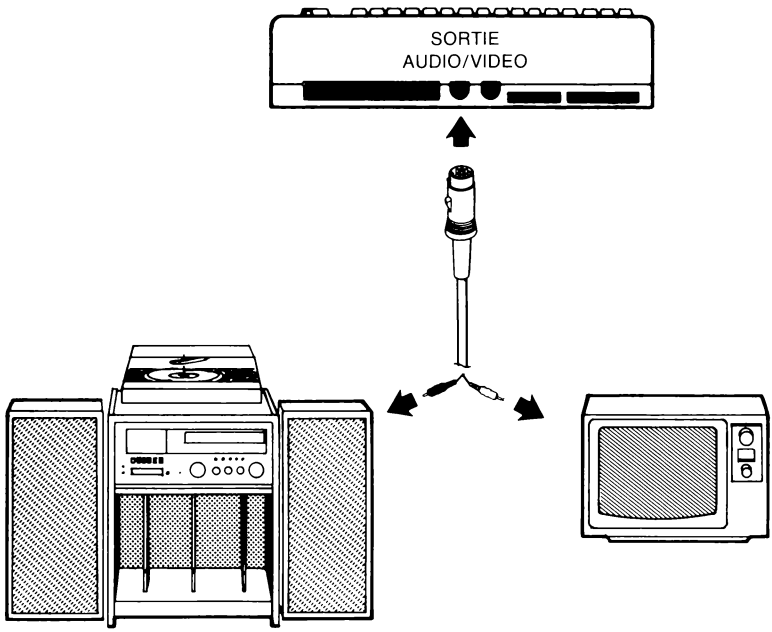
3. Branchement au secteur

Connectez votre alimentation avec le COMMODORE 64 (prise 1) et un branchement au secteur de 220 V/50 Hz.

4. Connexions optionnelles

Vu que le COMMODORE 64 fournit un canal audio de haute fidélité, vous pouvez souhaiter l'utiliser par l'intermédiaire d'un amplificateur de qualité.

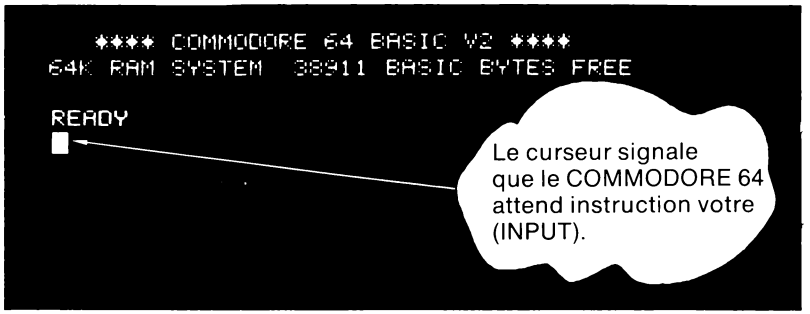
Le signal de son est pris également de la prise 6 (voir annexe I pour l'occupation des contacts de fiches).



Comme vous le savez certainement, vous avez la possibilité de brancher des périphériques comme la station de disquettes (Floppy Disk) VC 1541 ou l'imprimante VC 1525.

UTILISATION DU COMMODORE 64

1. Mettre l'ordinateur sous tension en utilisant l'interrupteur à bascule situé sur le côté droit.
2. Après quelques instants, les informations suivantes sont affichées sur l'écran TV:



3. Si votre télévision possède un bouton de réglage fin manuel, réglez l'image jusqu'à ce que vous obteniez une image claire.
4. Vous pouvez également souhaiter régler les commandes de couleur et de contraste de l'appareil de télévision pour obtenir la meilleure représentation. Vous pouvez utiliser la procédure de réglage de couleur décrite ci-dessous pour obtenir un bon réglage. Lorsque vous obtenez une image pour la première fois, l'écran doit apparaître généralement bleu foncé avec une bordure et les caractères bleu clair.

Si vous n'obtenez pas les résultats escomptés, contrôlez encore une fois les câbles et connexions. Le tableau ci-après vous aidera à localiser les problèmes.

TABLEAU DE DEPANNAGE

Symptôme	Cause	Remède
Voyant indicateur pas allumé	L'ordinateur n'est pas sous tension	Veiller à ce que l'interrupteur d'alimentation soit sur la position «ON»
	Le câble d'alimentation n'est pas branché	Contrôler la prise de courant pour savoir si le câble d'alimentation est bien enfoncé ou branché.
	La boîte d'alimentation n'est pas branchée	Contrôler le raccord et la prise de courant au mur
	Fusible défectueux dans l'ordinateur	Porter l'ordinateur chez le revendeur autorisé pour faire échanger le fusible
Affichage sur écran de visualisation	Sélectionné un mauvais canal sur la TV	Essayer un autre canal pour l'image (3 ou 4)
	Couplage incorrect	Le calculateur est couplé aux bornes de l'antenne VHF
	Câble vidéo non branché	Contrôler la connexion du câble de sortie de l'appareil de télévision
	Ordinateur réglé sur le mauvais canal	Régler l'ordinateur sur le même canal que celui de l'appareil de télévision (3 ou 4)
Grille aléatoire sur l'appareil de télévision, la cartouche étant en place	La cartouche n'est pas correctement insérée	Reinsérer la cartouche après avoir coupé l'alimentation
Image sans couleur	Mauvais réglage de l'appareil de TV	Modifier le réglage de l'appareil de télévision
Image d'une mauvaise couleur	Mauvais réglage de la couleur sur l'appareil TV	Régler les commandes de couleur/de contraste/de luminosité sur l'appareil TV
Image avec excès de bruit de fond	Volume sonore de la TV trop élevé	Ajuster le volume sonore de la TV
Image en ordre mais pas de son	Volume sonore de la TV trop faible	Ajuster le volume sonore de la TV
	Sortie auxiliaire mal branchée	Raccorder la prise de son à l'entrée auxiliaire de l'amplificateur et mettre sur l'entrée auxiliaire

CURSEUR

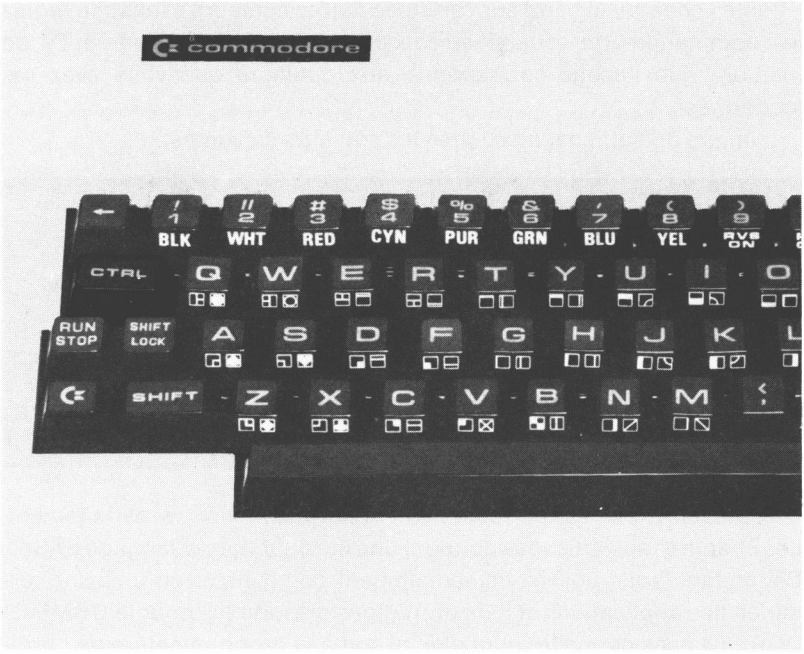
Le carré scintillant à côté de READY est intitulé le curseur et affiche sur l'écran la position et les lettres que vous tapez. Au fur et à mesure que vous tapez, le curseur avance d'un espace lorsque la position originale du curseur est remplacée par le caractère que vous tapez. Essayez de taper sur le clavier et observez comment les caractères que vous tapez sont affichés sur l'écran de la télévision.

REGLAGE DE LA COULEUR

Il existe une façon simple d'obtenir un échantillon des couleurs sur l'appareil de télévision de manière que vous puissiez aisément le régler. Même si vous ne vous êtes pas encore familiarisé avec le fonctionnement de l'ordinateur, il vous suffit de suivre les instructions et vous constaterez combien il est facile d'utiliser le COMMODORE 64. Tout d'abord, regardez sur le côté gauche du clavier et cherchez la touche CTRL. Il s'agit de l'abréviation de ConTRoL et elle est utilisée en liaison avec d'autres touches pour ordonner à l'ordinateur d'accomplir une tâche spécifique.

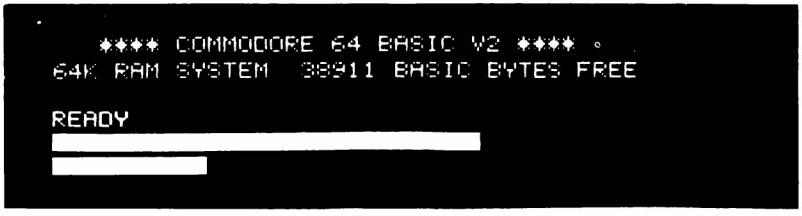
Pour utiliser une fonction de commande, vous maintenez enfoncée la touche CTRL tout en appuyant sur une seconde touche. Essayez ceci: Maintenez la touche CTRL enfoncée tout en appuyant sur la touche 9. Ensuite, relâchez les deux touches. Rien d'apparent ne doit se présenter mais, si vous touchez à présent n'importe quelle autre touche, l'écran indiquera le caractère affiché de façon inverse et non de façon normale – comme le message d'ouverture ou quoi que ce soit que vous ayez tapé auparavant.

Maintenez la touche SPACE BAR enfoncée. Que se passe-t-il? Si vous avez correctement effectué la procédure ci-dessus, vous constaterez qu'une barre bleu clair se déplace à travers l'écran et ensuite descend à la ligne suivante tant que la touche SPACE BAR est pressée.



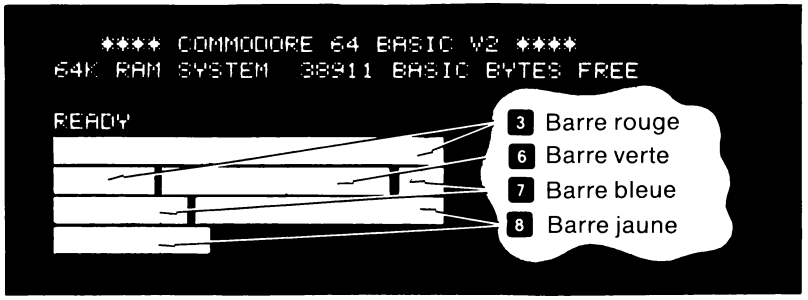
Après cela, maintenez **CTRL** enfoncée tout en appuyant sur l'une des autres touches numériques. Chacune d'elles possède sa propre couleur. Tout ce qui est affiché à partir de ce point le sera dans cette couleur. Par exemple, maintenez **CTRL** enfoncée et appuyez sur la touche **8** et relâchez les deux. A présent, maintenez **SPACE BAR** enfoncée.

Observez l'image. La barre est maintenant jaune! Par analogie, vous pouvez amener la barre sur toute autre couleur indiquée sur les touches numériques en maintenant **CTRL** enfoncée et en appuyant sur la touche appropriée.



Faites changer la barre sur certaines autres couleurs et ensuite réglez les commandes de couleur et de contraste de votre appareil TV de manière que l'image corresponde aux couleurs que vous avez sélectionnées.

L'image doit afficher à peu près les couleurs suivantes:



A présent, tout est correctement réglé et fonctionne parfaitement. Les chapitres suivants vous donnent une introduction au langage BASIC. Cependant, vous pouvez immédiatement commencer en utilisant certaines des applications et jeux prérédigés disponibles pour le COMMODORE 64 sans connaître quoi que ce soit à la programmation de l'ordinateur.

Chacun de ces programmes contient des informations détaillées sur leur utilisation. Il est néanmoins conseillé de lire les premiers chapitres de ce manuel pour se familiariser avec les bases de fonctionnement de votre nouveau système.

CHAPITRE 2

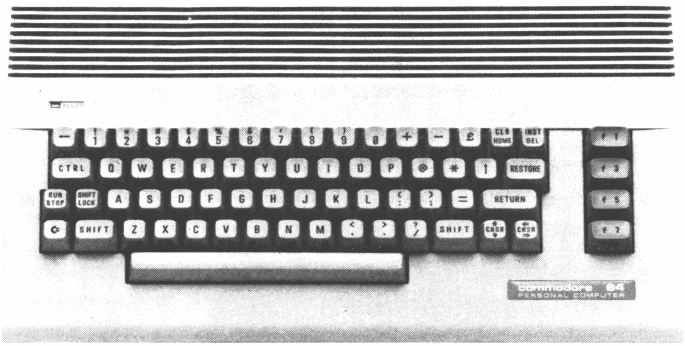
MISE EN SERVICE

- Clavier
- Chargement et sauvegarde de programmes
- La commande PRINT
- Calculs

CLAVIER

Dès que vous avez tout installé et réglé, prenez quelques minutes pour vous familiariser avec le clavier, étant donné qu'il est pour vous le moyen le plus important de communication avec le COMMODORE 64.

Dans l'ensemble le clavier ressemble beaucoup à celui d'une machine à écrire standardisée. Cependant, il existe un certain nombre de nouvelles touches qui commandent les fonctions spécialisées. Vous trouverez ci-dessous une brève description des différentes touches et la façon dont elles fonctionnent. Le fonctionnement détaillé de chaque touche sera expliqué plus tard.



RETURN

La touche RETURN signale à l'ordinateur de faire attention aux informations que vous frappez et de mettre ces informations en mémoire.

SHIFT

La touche SHIFT fonctionne comme celle d'une machine à écrire standard. De nombreuses touches sont capables d'afficher deux lettres ou symboles et deux caractères graphiques. Dans le mode «case supérieure/inférieure», la touche SHIFT vous fournit les caractères standards de la case supérieure. Dans le mode «case supérieure/graphique», la touche SHIFT affiche le caractère graphique représenté à droite sur la touche. Dans le cas de touches de fonctions spéciales, la touche SHIFT vous fournit la fonction marquée sur la partie supérieure de la touche.

EDITION

Personne n'est parfait et le COMMODORE 64 en tient compte. Un certain nombre de touches d'édition vous permet de corriger les fautes de frappe et de déplacer les informations sur l'écran.

CRSR

Il existe deux touches repérées CRSR (CuRSoR), l'une ayant des flèches vers le haut et vers le bas, l'autre ayant des flèches vers la gauche et vers la droite . . . Vous pouvez utiliser cette touche pour déplacer le curseur vers le haut et vers le bas ou vers la gauche et vers la droite. Dans le mode normal, les touches CRSR vous permettent de déplacer le curseur vers le bas et vers la droite. L'utilisation de la touche SHIFT et des touches CRSR permet au curseur d'être déplacé soit vers le haut, soit vers la gauche. Les touches du curseur possèdent un dispositif de répétition spécial qui fait bouger le curseur jusqu'à ce que vous relâchiez la touche.

INST/DEL

Si vous frappez la touche INST/DEL, le curseur se déplace d'un espace en effaçant (DELeting) le caractère que vous avez précédemment tapé. Si vous vous trouvez au centre d'une ligne, le caractère à gauche est supprimé et les caractères à droite se déplacent automatiquement pour combler l'espace.

La touche INST/DEL SHIFTed (décalée) vous permet d'insérer INSerT des informations sur une ligne. Par exemple, si vous avez décelé une faute de frappe au début d'une ligne – par exemple vous avez oublié la partie d'un mot – vous pouvez utiliser la touche CRSR pour revenir sur la faute et ensuite appuyer sur INST/DEL pour insérer un espace. Après quoi, il suffit de frapper la lettre qui manque.

CLR/HOME

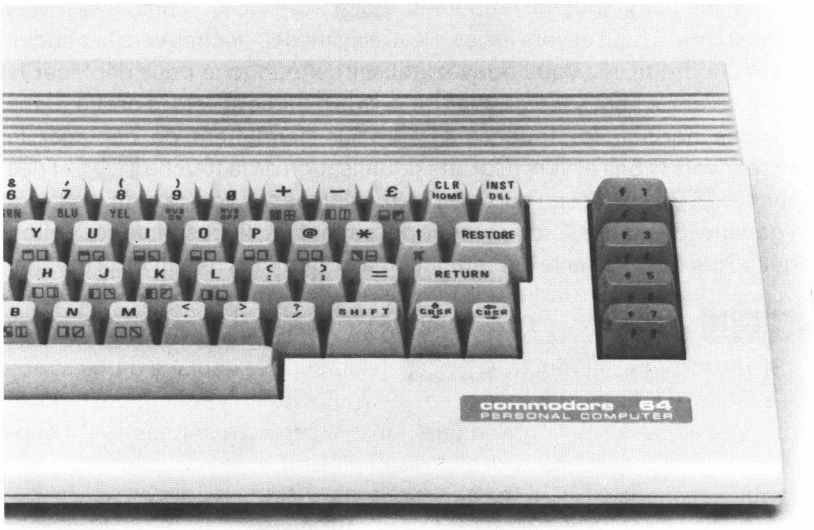
La touche CLR/HOME positionne le curseur sur la position «HOME» (le départ) de l'écran qui est le coin supérieur gauche. La touche CLR/HOME décalée efface l'écran et place le curseur sur la position de départ.

RESTORE

La touche RESTORE fonctionne comme son nom l'implique. Elle rétablit l'état normal dans l'ordinateur, celui qu'il avait avant que vous l'ayez modifié par un programme ou certaines commandes. Des informations complémentaires vous seront fournies à ce sujet dans les chapitres ultérieures.

TOUCHES DE FONCTION

Les quatre touches de fonction situées sur le côté droit du clavier peuvent être programmés pour traiter une variété de fonctions. Elles peuvent être définies de nombreuses façons accomplir des tâches répétitives.



CTRL

La touche **CTRL** qui est l'abréviation de ConTRoL vous permet de régler les couleurs et d'effectuer d'autres fonctions spécialisées. Maintenir la touche **CTRL** enfoncée tout en appuyant sur un certain nombre d'autres touches pour obtenir une fonction de commande. Vous avez eu la possibilité d'essayer la touche **CTRL** lorsque vous avez modifié les couleurs du texte pour créer des barres de couleurs différentes pendant la procédure d'installation.

RUN/STOP

Normalement, en appuyant sur la touche **RUN/STOP**, on arrête l'exécution d'un programme BASIC. Elle signale à l'ordinateur d'arrêter de faire quelque chose.

L'utilisation de la touche **RUN/STOP** dans le mode décalé vous permet de charger automatiquement un programme à partir d'une bande.

TOUCHE COMMODORE

La touche COMMODORE effectue un certain nombre de fonctions. En premier lieu, elle vous permet de passer du mode texte au mode graphique et vice versa. Lorsque l'ordinateur est mis en marche pour la première fois, il se trouve dans le mode graphique/case supérieure, c'est-à-dire que tout ce que vous tapez se trouve dans les lettres de la case supérieure. Comme mentionné l'utilisation de la touche SHIFT dans ce mode fait apparaître le graphique sur le côté droit des touches.

Si vous maintenez enfoncées le touche ainsi que la touche SHIFT, l'affichage passe sur le mode case supérieure et inférieure. A présent, si vous maintenez la touche enfoncée et une autre touche quelconque portant un symbole graphique, le graphique du côté gauche de la touche est représenté.

Pour revenir dans le mode case supérieure/graphique, maintenir la touche enfoncée et appuyer à nouveau sur la touche SHIFT.

La seconde fonction de la touche est de vous permettre d'accéder à un second jeu de 8 couleurs de texte. En maintenant la touche enfoncée et l'une des touches numériques, n'importe quel texte frappé apparaîtra dans l'autre couleur disponible sur la touche que vous avez pressée. Le chapitre 5 fournit une liste des couleurs de texte disponibles pour chaque touche.

RETOUR A LA NORMALE

A présent que vous avez obtenu une vue d'ensemble sur clavier, nous allons explorer certaines des nombreuses performances du COMMODORE 64.

Si vous avez toujours les barres de couleurs sur l'ecran depuis le réglage de votre appareil de télévision, maintenez les touches SHIFT et CLR/HOME enfoncées. L'écran doit s'effacer et le curseur doit se positionner sur le point «de départ» (coin supérieur gauche de l'écran).

A présent maintenez simultanément enfoncées les touches et 7. Ceci amène la couleur du texte sur le bleu clair. Un pas supplémentaire est nécesaire pour que tout revienne à la normale. Maintenez enfoncées les touches CTRL et 0 (zéro et non la lettre «O»!). Ceci ramène le mode d'affichage sur la normale. Rappelez-vous, nous avons inversé le terme REVERSE avec CTRL 9 pour créer les barres de couleurs (les barres de couleurs étaient effectivement des espaces d'une représentation inversée. Si nous étions dans le mode de texte normal pendant le test des couleurs, le curseur se serait déplacé mais en laissant des espaces blancs.

CONSEIL:

Maintenant que vous l'avez fait de façon compliquée, voilà la façon simple de réinitialiser la machine sur l'affichage normal. Appuyer simultanément sur

RUN/STOP et RESTORE

Ceci efface l'écran et ramène tout à la normale. Si un programme se trouve dans l'ordinateur, il ne sera pas touché. C'est une séquence à retenir, en particulier si vous programmez beaucoup.

Si vous souhaitez réinitialiser la machine comme si elle avait été coupée et ensuite remise sous tension, frappez: SYS 64738 et appuyez sur la touche RETURN. Soyez prudent si vous utilisez cette commande! Elle efface tous les programmes ou informations qui résident dans l'ordinateur.

CHARGEMENT ET SAUVEGARDE DE PROGRAMMES

Une des caractéristiques les plus importantes du COMMODORE 64 est d'enregistrer sur cassette ou disquette, des programmes que vous pouvez a nouveau changer par la suite.

Cette possibilité vous permet de conserver les programmes que vous rédigez pour vous en servir plus tard ou d'acheter des programmes prérédigés pour les utiliser avec le COMMODORE 64.

Veiller l'unité d'entraînement de disque, ainsi que l'unité DATASSETTE soient correctement fixées.

Chargement de programmes préconditionnés

Pour ceux qui s'intéressent à l'utilisation de programmes préprogrammés disponibles sur cartouches, cassettes ou disques. Voici tout ce que vous avez à faire:

1. **MODULES ENFICHABLES:** L'ordinateur COMMODORE 64 possède une gamme de programmes et de jeux sur cassettes. Les programmes offrent une grande variété d'applications commerciales et personnelles et les jeux sont semblables à des jeux commerciaux réels – pas des imitations. Pour charger ces jeux, mettre en premier lieu votre appareil TV en marche. Ensuite COUPER l'alimentation de votre COMMODORE 64. **VOUS DEVEZ COUPER L'ALIMENTATION DE VOTRE COMMODORE 64 AVANT D'INTRODUIRE OU D'ENLEVER LES CARTOUCHES SINON VOUS DETRUISEZ LA CARTOUCHE!** En troisième lieu, introduire la cartouche. Maintenant mettez votre COMMODORE 64 sous tension. Finalement, appuyez sur la touche START appropriée, tel que mentionné sur la feuille d'instruction qui est jointe à chaque jeu.

2. **Cassettes:** Utiliser votre enregistreur DATASSETTE et les cassettes audio ordinaires qui font partie de votre programme préconditionné. Veiller à ce que la bande soit complètement rebobinée jusqu'au début de la première face. Ensuite, il suffit de frapper LOAD. L'ordinateur répond par PRESS PLAY ON TAPE si bien que vous répondez en appuyant sur «play on» de votre unité DATASSETTE. A ce point, l'écran de l'ordinateur s'efface jusqu'à ce que le programme soit trouvé. L'ordinateur indique FOUND (NOM DU PROGRAMME) sur l'écran. Appuyez alors sur la touche C=. Ceci charge effectivement le programme dans l'ordinateur. Appuyer simplement sur la touche RUN/STOP si vous souhaitez interrompre le chargement.

3. **DISQUETTES:** En utilisant votre entraîneur de disquettes, introduire soigneusement la disquette préprogrammé de telle manière que l'étiquette de la disquette soit tournée vers le haut et dans votre direction. Rechercher une petite encoche sur la disquette (elle peut être recouverte par un bout de ruban). Si vous chargez la disquette correctement, l'encoche sera sur le côté gauche. Une fois la disquette en place, fermez le couvercle en appuyant sur le levier. A présent, frapper LOAD «NOM DU PROGRAMME», 8 et appuyez sur la touche RETURN. La disquette fait un bruit et votre écran indique:

```
SEARCHING FOR PROGRAM NAME
LOADING

READY
```

Lorsque READY apparaît et que le RETURN est allumé, il suffit d'appuyer sur RUN et votre software préprogrammé est prêt à être utilisé.

Chargement de programmes à partir de la bande

Le rechargement d'un programme à partir d'une bande ou d'un disque est tout aussi simple. Pour la bande, rebobiner la bande jusqu' au début et frapper:

```
LOAD "PROGRAM NAME"
```

Si vous ne vous rappelez pas le nom du programme, il suffit de frapper LOAD et le premier programme sur la bande est mémorisé.

oir appuyé sur la touche RETURN, l'ordinateur répond par:

```
PRESS PLAY ON TAPE
```

Après avoir appuyé sur la touche «play» (restitution), l'écran s'efface en rétablissant la couleur de fond de l'écran lorsque l'ordinateur cherche le programme. Lorsque le programme est trouvé, l'écran indique:

```
FOUND PROGRAM NAME
```

Pour charger effectivement le programme, presser la touche C=. Pour interrompre la procédure de chargement, presser la touche RUN/STOP. Si vous frappez la touche COMMODORE, l'écran rétablit la couleur du cadre pendant que le programme est chargé. Après l'achèvement de la procédure de chargement, l'écran revient à l'état normal et READY réapparaît rapidement.

Chargement de programmes à partir du disque

Le chargement d'un programme à partir du disque suit la même procédure. Frapper:

```
LOAD "PROGRAM NAME",8
```

Après avoir appuyé sur la touche RETURN, le disque commence à tourner à toute vitesse et l'affichage indique:

```
SEARCHING FOR PROGRAM NAME
LOADING

READY
```

Remarque:

Lorsque vous chargez un nouveau programme dans la mémoire de l'ordinateur, toutes les instructions qui se trouvaient auparavant dans l'ordinateur sont effacées. Veillez à sauvegarder le programme avec

lequel vous avez travaillé jusqu'à présent avant d'en charger un autre. Une fois un programme chargé, vous pouvez le faire passer, lister ou réaliser des modifications et sauvegarder la nouvelle version.

Sauvegarde de programmes sur une bande

Après avoir entré un programme, si vous souhaitez le sauvegarder sur une bande, frappez:

```
SAVE "PROGRAM NAME"
```

«LE NOM DU PROGRAMME» peut être une combinaison jusqu'à 16 caractères. Après avoir appuyé sur la touche RETURN, l'ordinateur répond par:

```
PRESS PLAY AND RECORD ON TAPE
```

Enfoncer en même temps les deux touches «RECORD et PLAY» sur l'unité DATASSETTE. L'écran s'efface en prenant la couleur du cadre.

Après l'enregistrement du programme sur la bande, READY réapparaît rapidement, indiquant que vous pouvez commencer à travailler sur un autre programme ou arrêter l'ordinateur pour un certain temps.

Sauvegarde de programmes sur un disque

La sauvegarde d'un programme sur un disque est même plus simple, Frapper:

```
SAVE "PROGRAM NAME",8
```

Le 8 est le code du disque, si bien qu'il vous suffit de faire savoir à l'ordinateur que vous souhaitez que le programme soit sauvegardé sur disque.

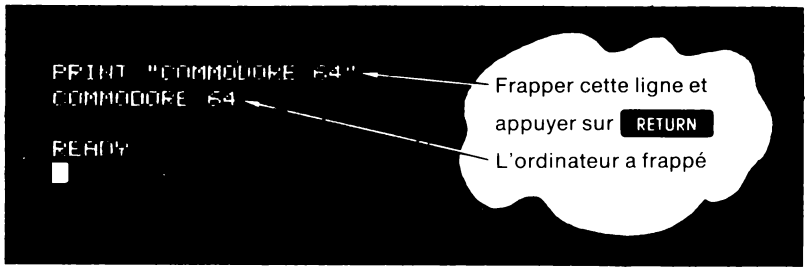
Après avoir appuyé sur la touche RETURN, le disque commence à tourner et l'ordinateur répond par:

```
SAVING "PROGRAM  NAME"
OK
READY
```

LA COMMANDE PRINT ET CALCULS

Après avoir effectué un certain nombre des opérations les plus difficiles, dont vous avez besoin pour conserver les programmes que vous souhaitez, commençons à réaliser certains programmes que vous pouvez sauvegarder.

Essayez de frapper ce qui suit exactement comme indiqué:



Si vous faites une faute de frappe, utilisez la touche INST/DEL pour effacer le caractère immédiatement à gauche du curseur. Vous pouvez supprimer autant de caractères que nécessaire.

Voyons ce qui s'est passé avec l'exemple ci-dessus. En premier lieu, vous avez donné l'instruction à l'ordinateur d'afficher (PRINT) ce qui se trouvait à l'intérieur des guillemets. En appuyant sur la touche RETURN, vous indiquez à l'ordinateur de faire selon vos instructions et COMMODORE 64 est affiché sur l'écran.

Lorsque vous utilisez l'instruction PRINT sous cette forme, tout ce qui est entre les guillemets est affiché exactement comme vous l'avez frappé. Si l'ordinateur répond par:

?SYNTAX ERROR

demandez-vous si vous avez commis une faute de frappe ou oublié les guillemets.

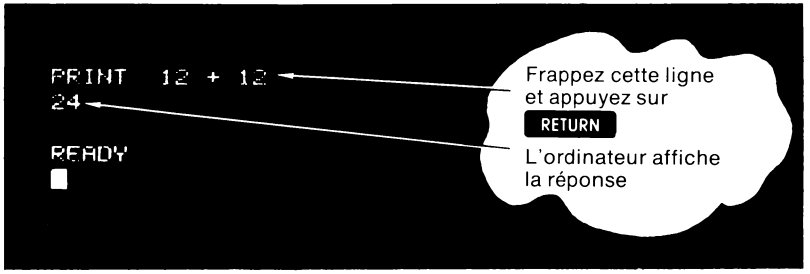
L'ordinateur est précis et suppose que les instructions lui soient données dans une forme déterminée. Mais ne vous inquiétez pas; il vous suffit de frapper les instructions telles que nous vous les présentons dans

les exemples et le COMMODORE 64 vous donnera entière satisfaction.

Rappelez-vous que vous ne pouvez détruire l'ordinateur en frappant les touches et que la meilleure façon d'apprendre le BASIC est de et de voir ce qui se passe.

PRINT est l'une des instructions les plus utiles et les plus performantes du langage BASIC. Avec elle, vous pouvez représenter presque tout ce que vous souhaitez, y compris des graphiques et des résultats de calculs.

Par exemple, essayez ce qui suit. Effacez l'écran en maintenant la touche SHIFT et la touche CLR/HOME enfoncées et frappez ce qui suit (veillez à utiliser la touche 1 pour un et non la lettre «1»):



Ce que vous avez découvert est que le COMMODORE 64 est un calculateur dans sa forme de base. Le résultat «24» a été calculé et affiché automatiquement. En fait, vous pouvez également réaliser des soustractions, multiplications, divisions, élévations à une puissance et des fonctions mathématiques avancées, telles que le calcul de racines carrées, etc. Vous n'êtes pas limité à un simple calcul sur une ligne, mais vous en saurez plus ultérieurement.

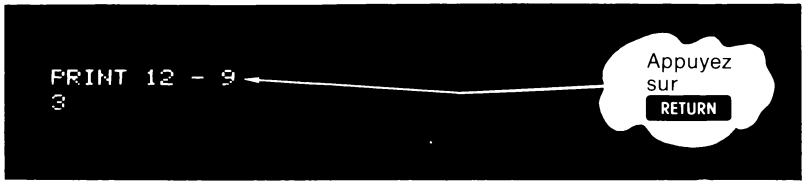
Remarquez que, dans la forme ci-dessus, PRINT se comporte différemment du premier exemple. Dans ce cas, une valeur ou le résultat d'un calcul est affiché et non le message exact que vous avez entré étant donné que les guillemets ont été omis.

Addition:

Le signe plus (+) signale une addition: Nous ordonnons à l'ordinateur d'afficher le résultat de 12 ajouté à 12. D'autres opérations arithmétiques présentent une forme similaire à celle de l'addition. Il faut toujours se rappeler de frapper RETURN après avoir frappé PRINT et le calcul.

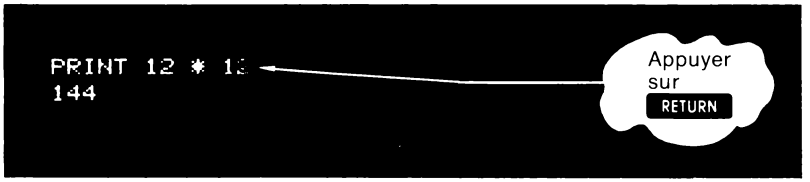
Soustraction

Pour soustraire, utiliser l'essayer le signe conventionnel moins (−): Frappez



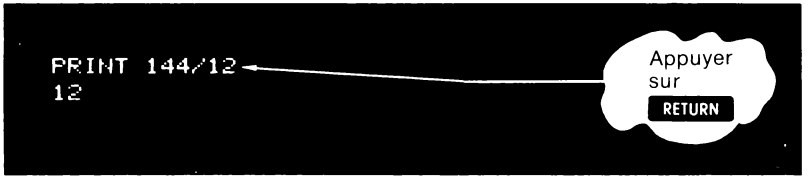
Multiplication

Si vous souhaitez multiplier 12 par 12, utilisez l'astérisque (*) pour représenter la multiplication. Vous devez frapper:



Division

La division utilise la (/) familière. Par exemple, pour diviser 144 par 12, frappez:



Elevation à une puissance

De façon analogue, vous pouvez aisément élever un nombre à une puissance (c'est-à-dire multiplier un nombre par lui-même un nombre déterminé de fois). La «↑» (flèche dirigée vers le haut) signifie élevation à la puissance.

Ce qui revient à frapper:

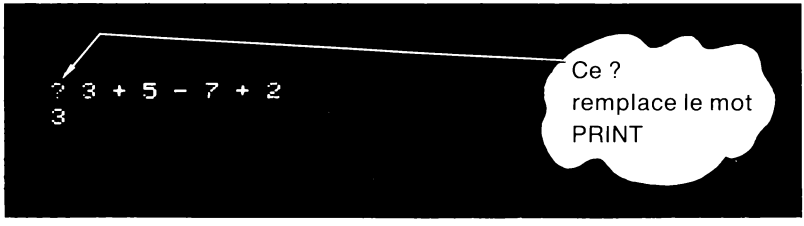
```
PRINT 12 * 12 * 12 * 12 * 12
248832
```

CONSEIL:

Le langage BASIC dispose d'un certain nombre de possibilités raccourcies pour réaliser les opérations. Une telle possibilité consiste à abréger des commandes BASIC (ou mots-clés). Par exemple, un ? peut être utilisé à la place de PRINT. Au fur et à mesure que nous avancerons, nous vous présenterons de nombreuses commandes; l'annexe D fournit les abréviations de chacune d'elle et ce qui est affiché sur l'écran lorsque vous tapez la forme abrégée.

Le dernier exemple illustre un autre détail important: Plusieurs calculs peuvent être réalisés sur la même ligne et ils peuvent être de types mixtes.

Vous pouvez calculer ce problème:



Jusqu'è ce point, nous n'avons utilisé que de petits nombres et des exemples simples. Cependant, le COMMODORE 64 est en mesure d'effectuer des calculs plus complexes. Vous pouvez, par exemple, additionner plusieurs nombres importants. Essayez cet exemple, mais ne faites pas appel aux virgules, sinon vous allez commettre une erreur:

```
? 123.45 + 345.78 + 7895.687
8364.917
```

Cela paraît fantastique, mais maintenant essayez ceci:

```
? 12123123.45 + 345.78 + 7895.687
12131364.9
```

Si vous prenez le temps de réaliser l'addition à la main, vous obtenez un autre résultat.

Qu'est-ce qui se passe ? Bien que l'ordinateur ait une grande puissance, il existe une limite aux nombres qu'il peut traiter. Le COMMODORE 64 peut traiter des nombres comportant 10 chiffres. Cependant, lorsqu'un nombre est représenté, seuls 9 chiffres sont affichés.

Ainsi dans notre exemple, le résultat a été «arrondi» pour ne pas dépasser 9 chiffres. Le COMMODORE 64 arrondit à la valeur supérieure lorsque le chiffre suivant est 5 ou plus, il arrondit à la valeur inférieure lorsque le chiffre suivant est 4 ou moins.

Les nombres compris entre 0,01 et 99,999,999 sont affichés en utilisant la notation standard. Les nombres à l'extérieur de ces limites sont affichés en utilisant la notation scientifique.

La notation scientifique est seulement un moyen d'exprimer un nombre très petit ou très grand sous forme d'une puissance de 10.

Si vous frappez:

```
? 12300000000000000000
1.23E+17
```

Ce qui revient à 1.23*1017 et qui est utilisé pour conserver l'ordre des choses. Il existe des limites aux nombres que l'ordinateur peut traiter, même en notation scientifique. Ces limites sont les suivantes:

Plus grand nombre: 1.70141183E+38
Plus petit nombre: 2.93873588E−39

PRIORITES

Si vous essayez de réaliser certains calculs mixtes différents des exemples auparavant présentés, vous pouvez ne pas obtenir les résultats que vous escomptez. La raison en est que l'ordinateur réalise les calculs dans un certain ordre.

Dans ce calcul:

$$20 + 8/2$$

vous ne pouvez dire si la réponse doit être 24 ou 14 jusqu'à ce que vous sachiez l'ordre suivant lequel réaliser les calculs. Si vous ajoutez 20 à 8 divisé par 2 (ou 4), le résultat est 24. Mais, si vous ajoutez 20 à 8 et ensuite divisé par 2, la réponse est 14. Essayez cet exemple et vérifiez le résultat que vous obtenez.

La raison pour laquelle vous avez obtenu 24 est due au fait que le COMMODORE 64 réalise les calculs de gauche à droite conformément à l'ordre suivant:

Premièrement: −	Signe moins indiquant les nombres négatifs
Deuxièmement: ↑	Puissance de gauche à droite
Troisièmement: */	Multiplications et divisions de gauche à droite
Quartrièmement: +−	Additions et soustractions de gauche à droite

En suivant les opérations selon l'ordre de priorité, vous constaterez que dans l'exemple ci-dessus la division a été réalisée en premier lieu et ensuite l'addition pour obtenir le résultat de 24.

Préparez certains problèmes vous-même et observez si vous pouvez suivre la procédure et prévoir les résultats conformément aux règles établies ci-dessus.

Il existe également une façon aisée de modifier la procédure de priorité en utilisant des parenthèses pour sortir les opérations que vous souhaitez réaliser en premier. Par exemple, si vous souhaitez diviser 35 par 5 + 2, vous frappez:

Vous obtiendrez 35 divisé par 5 avec 2 ajouté à la réponse, ce qui n'est pas du tout ce que vous entendiez. Pour obtenir ce que vous souhaitez réellement, essayez ceci:

A présent, l'ordinateur évalue en premier lieu ce qui est contenu dans la parenthèse. S'il existe des parenthèses à l'intérieur des parenthèses, les plus à l'intérieur sont évaluées en premier lieu.

Lorsqu'il existe plusieurs parenthèses sur une ligne, comme par exemple:

```
? (12 + 9) * (6 + 1)
147
```

l'ordinateur procéde aux évaluations de gauche à droite. Ici 21 devrait être multipiié par 7 et donner comme résultat 147.

COMBINAISONS

Même si nous avons consacré pas mal de temps à des domaines qui peuvent ne pas sembler très importants, les détails présentés ici prendront toute leur importance lorsque vous commencerez à programmer et s'avèreront d'une valeur inestimable. Pour vous donner une idée des rapports, considérez ce qui suit: Comment faut-il combiner les deux types d'instructions d'affichage que nous avons examinés, de manière à afficher sur l'écran quelque chose de plus significatif?

Nous savons qu'en mettant quelque chose entre guillemets, les informations apparaissent sur l'écran exactement telles qu'elles ont été entrées et qu'un utilisant des opérateurs mathématiques, des calculs peuvent être effectués. Alors pourquoi ne pas combiner les deux types d'instructions PRINT comme ceci:

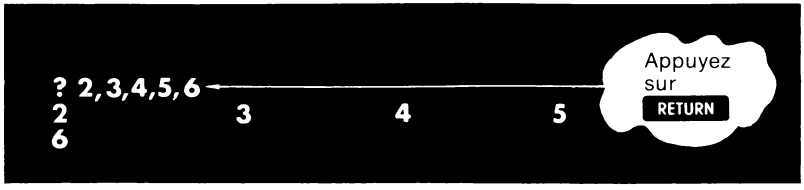
LE POINT VIRGULE SIGNIFIE SANS ESPACE

```
? "5 * 9 = "; 5 * 9
5 * 9 = 45
```

Même si ceci peut sembler un peu superflu, nous avons simplement utilisé les deux types d'instructions d'affichage ensemble. La première partie affiche «5*9=» exactement telle qu'elle a été frappée. La seconde partie effectue le travail proprement dit et affiche le résultat avec le point-virgule séparant la partie message de l'instruction du calcul effectif.

Vous devez toujours séparer les parties d'une instruction d'affichage mixte par une certaine ponctuation pour qu'elle fonctionne correctement. Essayez une virgule à la place du point-virgule et observez ce qui se passe.

Curieusement, le point-virgule amène la partie du texte de l'instruction à être éditée immédiatement après la partie précédente sans espace. La virgule a un effet quelque peu différent. Même s'il s'agit d'un séparateur acceptable, elle espace les choses un peu plus. Si vous frappez:



les nombres seront affichés sur la largeur de l'écran et en-dessous sur la ligne suivante.

L'image du COMMODORE 64 est divisée en 4 zones de 10 colonnes chacune. La virgule tabule chaque résultat dans la zone suivante disponible. Vu que nous avons demandé que plus d'informations s'affichent sur une ligne (nous avons essayé de placer 5 zones de 10 colonnes sur une ligne), la dernière position s'est reportée à la ligne suivante.

La différence fondamentale entre la virgule et le point-virgule dans le formatage des instructions PRINT peut être un avantage lors de la création d'images plus complexes: Elle nous permet de créer très facilement des résultats plus sophistiqués.

CHAPITRE 3

INITIATION A LA PROGRAMMATION BASIC

- GOTO
- Conseils d'édition
- IF . . . THEN
- Boucles FOR . . . NEXT

LA PROCHAINE ETAPE

Jusqu'à présent, nous avons réalisé certaines opérations simples en entrant une seule ligne d'instruction dans l'ordinateur. Après avoir appuyé sur la touche RETURN, l'opération déterminée fut immédiatement accomplie. Ceci est intitulé le mode IMMEDIAT ou CALCULATEUR.

Mais, pour être plus efficace, nous devons être en mesure d'utiliser l'ordinateur avec plus d'une ligne d'instruction. Un certain nombre d'instructions combinées entre elles est intitulé un PROGRAMME et vous permet d'exploiter la pleine puissance du COMMODORE 64.
Pour voir avec quelle facilité vous rédigez votre premier programme sur le COMMODORE 64, essayez ceci:

Effacez l'écran en maintenant la touche SHIFT enfoncée, ensuite appuyez sur la touche CLR/HOME.

Frappez NEW et appuyez sur la touche RETURN. (Ceci efface tous les chiffres qui pourraient avoir été laissés dans l'ordinateur à la suite de vos essais).

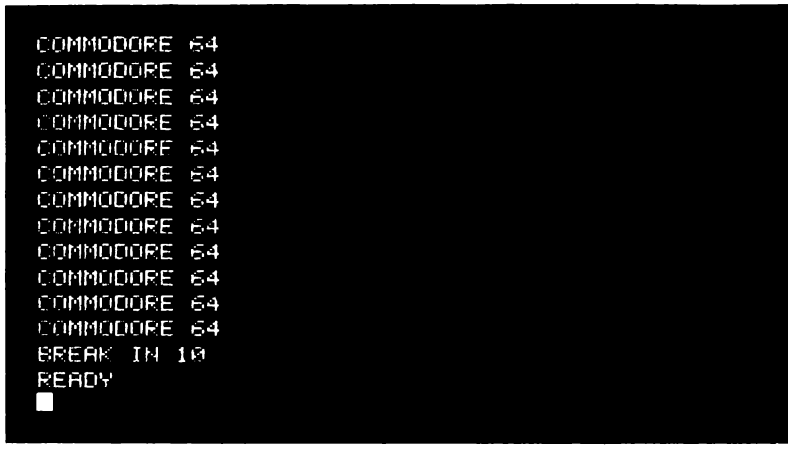
Maintenant frappez exactement ce qui suit: (n'oubliez pas d'appuyer sur la touche RETURN après chaque ligne).

```
10 ?"COMMODORE 64"
20 GOTO 10
```

A présent, frappez RUN et appuyez sur la touche RETURN – observez ce qui se passe. Votre écran s'anime avec COMMODORE 64. Après que vous avez fini de regarder l'image, appuyez sur RUN/STOP pour arrêter le programme. Un nombre important de concepts a été introduit dans ce bref programme, qui constitue la base de toute programmation.

Notez que nous avons fait précéder chaque instruction d'un numéro. Il s'agit du numéro de la ligne qui indique à l'ordinateur dans quel ordre traiter chaque instruction. Ces numéros représentent également une référence dans le cas où le programme a besoin de revenir sur une ligne particulière. Les numéros de ligne peuvent être un nombre entier (integer) quelconque dont la valeur est comprise entre 0 et 63,999.

```
10 PRINT "COMMODORE 64"
↑    ↑
└    └── Instruction
└──────── Numéro de ligne
```



Une bonne procédure de programmation consiste à numéroter les lignes par pas de 10 – au cas où vous auriez besoin d'introduire certaines instructions par après. En dehors de PRINT, notre programme utilise également une autre commande BASIC, GOTO. Celle-ci ordonne à l'ordinateur de passer directement à une ligne particulière et de la traiter, pour ensuite poursuivre à partir de ce point.

```
10 PRINT "COMMODORE 64"
20 GOTO 10
```

Dans notre exemple, le programme affiche le message sur le ligne 10, passe à la ligne suivante (20), qui lui donne pour instruction de repasser à ligne 10 et d'afficher le message encore une fois. Ensuite, le cycle se répète. Vu que nous n'avons pas indiqué à l'ordinateur une voie pour sortir de cette boucle, le programme continue infiniment jusqu'à ce que nous l'arrêtions physiquement avec la touche RUN/STOP.

Après que vous avez arrêté le programme, frappez: LIST. Notre programme sera affiché, intact, parce qu'il est toujours présent dans la mémoire de l'ordinateur. Notez également que l'ordinateur convertit ? en PRINT pour vous. Le programme peut maintenant être modifié, sauvegardé ou repassé.

Une autre différence importante entre frapper quelque chose dans le mode immédiat et écrire un programme concerne le fait qu'une instruction immédiate exécutée et effacée de l'écran elle est perdue. Cependant, vous pouvez toujours revenir sur un programme en frappant seulement LIST.

D'autre part, lorsqu'il s'agit d'abréviations, n'oubliez pas que l'ordinateur peut sortir de la ligne si vous en utilisez de trop.

CONSEILS D'EDITION

Si vous faites une faute sur une ligne, vous disposez d'un certain nombre d'options d'édition.

1. Vous pouvez frapper encore une fois une ligne à un moment voulu et l'ordinateur substitue automatiquement la nouvelle ligne à l'ancienne.
2. Une ligne non souhaitée peut être effacée en frappant simplement le numéro de ligne et RETURN.
3. Vous pouvez également aisément éditer une ligne existante en utilisant les touches curseur et édition.

Supposons que vous ayez fait une faute de frappe dans une ligne de l'exemple. Pour la corriger sans retaper la totalité de la ligne, essayez cette procédure:

Frappez LIST, ensuite en utilisant les touches SHIFT et CRSR conjointement, déplacez le curseur vers le haut jusqu'à ce qu'il soit positionné sur la ligne qui a besoin d'être modifiée.

A présent, utilisez la touche curseur droite pour déplacer le curseur sur le caractère que vous souhaitez modifier, frappez la modification sur l'ancien caractère. Puis appuyez sur la touche RETURN et la ligne corrigée remplace l'ancienne.

Si vous avez besoin de plus d'espace sur la ligne, positionnez le curseur là où l'espace est nécessaire, appuyez simultanément sur les touches SHIFT et INST/DEL et un espace est ouvert. A présent, il suffit de frapper l'information supplémentaire dans l'espace concerné et d'appuyer sur la touche RETURN. Par analogie, vous pouvez supprimer des caractères non désirés en plaçant le curseur sur la droite du caractère non désiré et en appuyant sur la touche INST/DEL.

Pour vérifier si les modifications sont entrées, frappez à nouveau LIST et le programme corrigé est affiché! D'ailleurs, il n'est pas nécessaire que les lignes soient entrées par ordre numérique. L'ordinateur les place automatiquement dans la séquence adéquate.

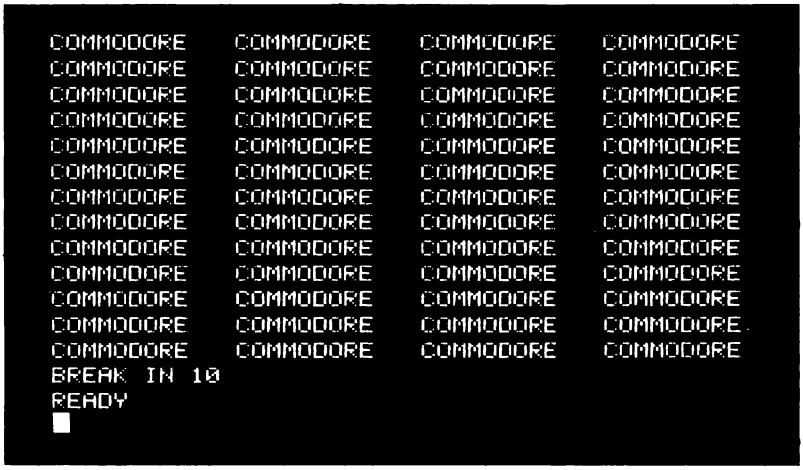
Essayez d'éditer notre programme-type en modifiant la ligne 10 et en ajoutant un point-virgule à la fin de la ligne, comme indiqué page 35. Ensuite, repassez le programme.

10 PRINT"COMMODORE"

N'oubliez pas de déplacer le curseur au-delà de la ligne 20 avant de passer le programme.

VARIABLES

Les variables sont certaines des facilités les plus utilisées de tout langage de programmation étant donné que les variables peuvent représenter beaucoup plus d'informations dans l'ordinateur. La compréhension de la façon dont fonctionnent les variables rend le calcul plus aisé et nous permet de réaliser des exploits qui sans elle ne seraient pas possibles.



Imaginez un nombre de cases à l'intérieur de l'ordinateur qui peuvent chacune contenir un certain nombre ou une chaîne de caractères de texte. Chacune de ces cases doit être étiquetée par un nom que nous choisissons. Ce nom est appelé une variable et représente l'information dans la case correspondante.

Par exemple, si nous disons:

```
10 X% = 15
20 X = 23,5
30 X$ = "LA SOMME DE X% + X ="
```

L'ordinateur va représenter les variables comme suit:

```
X%    15
X     23,5
X$    LA SOMME DE X% + X =
```

Un nom de variable représente la case ou l'emplacement dans la mémoire où la valeur effective de la variable est mémorisée. Comme vous pouvez le constater, nous pouvons assigner soit un nombre entier, soit

un nombre à virgule flottante, soit une chaîne de textes à une variable. Le symbole % suivant un nom de variable indique que la variable représente un nombre entier. Vous trouverez ci-dessous des noms de variables entiers valides:

A%
X%
A1%
NM%

Le «$» suivant le nom de la variable indique que la variable représente une chaîne de textes. Vous trouverez ci-dessous des exemples de variables de chaîne.

A$
X$
MI$

Les variables à virgule flottante suivent le même format, avec l'indicateur de type:

A1
X
Y
MI

En assignant un nom à une variable il faut tenir compte d'un certain nombre de conditions: En premier lieu, un nom de variable peut avoir un ou deux caractères. Le premier caractère doit être un caractère alphabétique de A à Z; le second caractère peut être soit alphabétique, soit numérique (dans la gamme 0 à 9). Un troisième caractère peut être incorporé pour indiquer le type de variable (nombre entier ou chaîne de caractères), 4 ou $.

Vous pouvez utiliser des noms de variables possédant plus de 2 caractères alphabétiques, mais seuls les deux premiers sont identifiés par l'ordinateur. Ainsi PA et PARTNO sont identiques et se réfèrent à la même case de variable.

La dernière règle des noms de variables est simple: Ils ne peuvent contenir aucun mot-clé BASIC (mots réservés), tel que GOTO, RUN, etc. Reportez-vous à l'annexe D où vous trouverez une liste complète des mots BASIC réservés. Pour voir la façon dont les variables peuvent être mises en œuvre, frappez le programme complet qui a été introduit précédemment et faites le passer. N'oubliez-pas d'appuyer sur la touche RETURN après chaque ligne du programme.

```
READY.

10 X% = 15
20 X = 23.5
30 X$ = "LA SOMME DE X% + X = "
40 PRINT "X% = "; X%, "X = "; X
50 PRINT X$; X% + X

READY.
```

Si vous avez fait tout ce qui est indiqué, vous devez obtenir le résultat suivant affiché sur l'écran.

```
RUN
X% =  15    X =  23.5
LA SOMME DE  X% + X =  38.5
READY
```

Nous avons combiné toutes les astuces apprises concernant le formatage de l'image comme vous pouvez le constater et affiché la somme des deux variables.

Aux lignes 10 et 20, nous avons assigné une valeur entière à X% et assigné une valeur à virgule flottante à X. Ceci place le nombre associé avec la variable dans sa case. A la ligne 30, nous avons assigné une chaîne de textes à X$. La ligne 40 combine les deux types d'instruction PRINT pour afficher un message et la valeur effective de X% et X. La ligne 50 affiche la chaîne de textes assignée à X$ et la somme de X% et X.

Notez que même si X est utilisé comme une part de chaque nom de variable, les identificateurs % et $ rendent X%, X et X$ uniques, représentant ainsi 3 variables distinctes. Mais les variables sont encore beaucoup plus intéressantes.

Si vous changez leur valeur, la nouvelle valeur remplace la valeur originale dans la même case. Ceci vous permet d'écrire une instruction comme suit:

X = X + 1

Ce ne serait jamais accepté dans l'algèbre normale, mais il s'agit d'un des concepts les plus utilisés dans la programmation. Il signifie: prendre la valeur courante de X, ajouter un et placer la nouvelle somme dans la case représentant X.

IF . . . THEN

Capables de modifier aisément la valeur des variables, nous pouvons à présent essayer un programme comme suit:

```
NEW
10 CT = 0
20 ?"COMMODORE 64"
30 CT = CT + 1
40 IF CT < 5 THEN 20
50 END
RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Ce que nous avons fait est d'introduire deux commandes BASIC et d'assurer un certain contrôle sur notre petit programme d'édition introduit au début de ce chapitre. IF . . . THEN ajoute une certaine logique au programme. Il signifie IF (si) une condition est confirmée THEN (alors) effectuez quelque chose. IF (si) la condition n'est plus confirmée, THEN (alors) passez à la ligne suivante du programme.

Un certain nombre de conditions peut être défini en utilisant l'instruction IF . . . THEN.

SYMBOLE	*SIGNIFICATION*
<	Inférieur à
>	Supérieur à
=	Egal à
< >	Différent de
> =	Supérieur ou égal à
< =	Inférieur ou égal à

L'utilisation de l'une de ces conditions est facile tout en offrant des possibilités étonnantes.

```
10 CT = 0
20 ?"COMMODORE 64"
30 CT = CT + 1
40 IF CT < 5 THEN 20
50 END
```

Dans l'exemple du programme, nous avons introduit une «boucle» qui imposait un certain nombre de restrictions en affirmant: IF (si) une valeur est inférieure à un certain nombre, THEN (alors) effectuez quelque chose.

La ligne 10 définit CT (CounT = comptage) égal à 0. La ligne 20 affiche notre message. La ligne 30 ajoute un à la variable CT. Cette ligne compte combien de fois nous avons exécuté la boucle. Chaque fois que la boucle est exécutée, CT augmente d'une unité.

La ligne 40 est notre ligne de contrôle. Si CT est inférieur à 5, signifiant que nous avons exécuté la boucle moins de 5 fois, le programme revient à la ligne 20 et procède à une réédition. Lorsque CT devient égal à 5 – indiquant que 5 COMMODORE 64 ont été affichés – le programme passe à la ligne 50 qui signale la fin du programme.

Essayez le programme pour voir de quoi il s'agit. En modifiant la limite CT de la ligne 40, vous pouvez faire afficher un nombre quelconque de lignes. IF . . . THEN a une multitude d'autres utilisations que nous examinerons dans de futurs exemples.

BOUCLES FOR . . . NEXT

Il y a une méthode plus simple et préférable pour réaliser ce que nous avons fait dans l'exemple précédent, soit en utilisant une boucle FOR . . . NEXT. Considérons ce qui suit:

```
NEW

10 FOR CT = 1 TO 5
20 PRINT "COMMODORE 64"
30 NEXT CT

RUN
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
COMMODORE 64
```

Comme vous pouvez le constater, le programme est beaucoup plus restreint et plus direct.

CT commence par 1 la ligne 10. Ensuite, la ligne 20 provoque un certain affichage. A la ligne 30, CT est avancé de 1. L'instruction NEXT de la ligne 30 renvoie automatiquement le programme à la ligne 10 où la partie FOR de l'instruction FOR . . . NEXT est située. Ce processus se poursuit

jusqu'à ce que CT atteigne la limite que vous avez entrée. La variable utilisée dans une boucle FOR . . . NEXT peut être avancée de valeurs inférieures à 1 si nécessaire. Essayez cet exemple:

```
NEW

10 FOR NB = 1 TO 10 STEP .5
20 PRINT NB,
30 NEXT NB

RUN
1             1.5           2             2.5
3             3.5           4             4.5
5             5.5           6             6.5
7             7.5           8             8.5
9             9.5           10
```

Si vous entrez et passez ce programme, vous constaterez des nombres de 1 à 10, espacés de 0,5, affichés sur l'écran. Tout ce que nous avons fait ici est d'afficher les valeurs que NB est supposé prendre lorsqu'il passe dans une boucle. Vous pouvez même spécifier augmente ou diminue. Substituons ce qui suit à la ligne 10:

10 FOR NB = 10 TO 1 STEP −.5

et observez que le contraire se passe quand NB passe de 10 à 1 par ordre décroissant.

CHAPITRE 4

BASIC AVANCÉ

- Introduction
- Animation simple – Boucles imbriquées
- INPUT
- GET
- Nombres aléatoires
- Jeu de devinettes
- Jeu de dés
- Graphiques aléatoires Fonctions CHR$ et ASC

INTRODUCTION

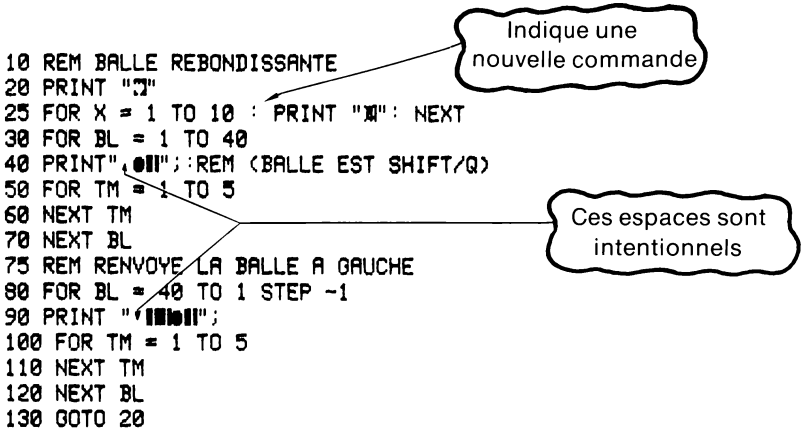
Les quelques chapitres qui suivent ont été rédigés pour les personnes qui se sont relativement familiarisées avec le langage de programmation BASIC et les concepts nécessaires pour rédiger des programmes des plus avancés.

Pour ceux d'entre vous qui commencent tout juste à apprendre à programmer , vous pouvez trouver certaines informations un peu trop techniques pour les comprendre complètement. Mais soyez attentif, étant donné que pour ces deux chapitres amusants GRAPHIQUES DE SYLPHES et CREATIONS SONORES, nous avons élaboré certains exemples simples qui sont rédigés pour les nouveaux utilisateurs. Les exemples vous fourniront une bonne idée de la façon dont utiliser les possibilités graphiques et sonores sophistiquées disponibles sur votre COMMODORE 64.

Si vous avez décidé de vouloir savoir plus sur la rédaction des programmes en langage BASIC, nous avons prévu un lexique (annexe N) à la fin de ce manuel. Si vous vous êtes déjà familiarisé avec la programmation BASIC, ces chapitres vous faciliteront l'apprentissage des techniques de programmation BASIC avancé. Vous trouverez des informations plus détaillées dans le manuel de référence du programmeur COMMODORE 64, disponible auprès de votre revendeur local Commodore.

ANIMATION SIMPLE

Essayons certaines des possibilités graphiques du COMMODORE 64 en prenant tout ce que nous avons vu jusqu'à présent et en y ajoutant quelques nouveaux concepts. Si vous êtes ambitieux, frappez le programme suivant et regardez ce qui se passe. Vous noterez que dans le contexte des instructions d'affichage nous pouvons également donner des ordres au curseur et à l'écran. Si vous trouvez quelque chose comme CRSR LEFT dans une liste de programme, maintenez la touche SHIFT enfoncée et appuyez sur la touche CRSR LEFT/RIGHT. L'écran affiche la représentation graphique d'un curseur gauche (deux barres verticales inversées). De la même façon, l'action de SHIFT et CLR/HOME affiche un coeur inversé.




```
10 REM BALLE REBONDISSANTE
20 PRINT "■"
25 FOR X = 1 TO 10 : PRINT "■": NEXT
30 FOR BL = 1 TO 40
40 PRINT"  ■■";:REM (BALLE EST SHIFT/Q)
50 FOR TM = 1 TO 5
60 NEXT TM
70 NEXT BL
75 REM RENVOYE LA BALLE A GAUCHE
80 FOR BL = 40 TO 1 STEP -1
90 PRINT "■■■■";
100 FOR TM = 1 TO 5
110 NEXT TM
120 NEXT BL
130 GOTO 20
```

CONSEIL:

Tous les mots de ce texte seront achevés sur une ligne. Cependant, tant que vous n'avez pas pressé la touche RETURN, votre COMMODORE 64 passera automatiquement à la ligne suivante, même au milieu d'un mot. Le programme représentera une balle rebondissante, se déplaçant de gauche à droite et vice versa à travers l'écran. Si nous regardons bien le programme (reproduit page 44), vous pouvez constater la façon dont cet exploit est réalisé.

La ligne 10 est une REMarque qui indique ce que fait le programme; elle n'a pas d'effet sur le programme proprement dit.

La ligne 20 efface toutes les informations de l'écran. La ligne 25 affiche 10 instructions de descente du curseur. Ceci positionne la balle au centre

```
    10 REM BALLE REBONDISSANTE
    20 PRINT "■"
    25 FOR X = 1 TO 10 : PRINT "■": NEXT
    30 FOR BL = 1 TO 40
    40 PRINT" ●■■";:REM (BALLE EST SHIFT/Q)
    50 FOR TM = 1 TO 5
    60 NEXT TM
    70 NEXT BL
    75 REM RENVOYE LA BALLE A GAUCHE
    80 FOR BL = 40 TO 1 STEP -1
    90 PRINT " ■■●■■";
   100 FOR TM = 1 TO 5
   110 NEXT TM
   120 NEXT BL
   130 GOTO 20
```

de l'écran. Si la ligne 25 avait été éliminée, la balle se serait déplacée sur la ligne supérieure de l'écran.

La ligne 30 détermine une boucle de déplacement de la balle sur les 40 colonnes de gauche à droite.

La ligne 40 accomplit un énorme travail. En premier lieu, elle prévoit un espace pour effacer les positions antérieures de la balle, ensuite elle affiche la balle et finalement elle effectue un déplacement à gauche du curseur pour tout préparer afin d'effacer à nouveau l'actuelle position de la balle. La boucle établie lignes 50 et 60 ralentit légèrement la balle ou retarde le programme. Sans elle, la balle se déplacerait trop rapidement pour l'observer. La ligne 70 achève la boucle déterminée ligne 30 qui affiche les balles sur l'écran. Chaque fois que la boucle est exécutée, la balle se déplace d'un autre espace vers la droite. Comme vous le noterez à partir de l'illustration, nous avons créé une boucle à l'intérieur d'une boucle.

Ceci est parfaitement acceptable. La seule fois que vous aurez des problèmes ce sera quand le boucles se croisent. Il est utile dans la rédaction de programmes, que vous contrôlez vous-même, comme illustré ici, si la logique d'une boucle est correcte. Pour vérifier ce qui arrive si vous croisez une boucle, inversez les instructions des lignes 60 et 70. Vous obtiendrez une erreur, étant donné que l'ordinateur sera confus et ne peut afficher ce qui se passe.

Le lignes 80 à 120 renversent uniquement les étapes de le première partie du programme et déplacent la balle de droite à gauche. La ligne 90 est légèrement différente de la ligne 40, étant donné que la balle se déplace dans la direction opposée (nous avons à effacer la balle vers la droite et à la déplacer vers la gauche).

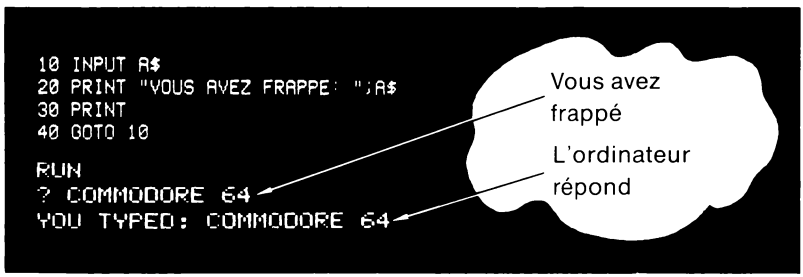
Et quand tout ceci est fait, le programme revient à la ligne 20 pour recommencer tout le processus. Joli, n'est-ce-pas!
Pour varier le programme, éditez la ligne 40 pour lire:

Pour obtenir le zéro, maintenez la touche SHIFT enfoncée et frappez la lettre «Q».

Passez le programme et observez ce qui se passe maintenant. Etant donné que nous n'avons pas tenu compte de la commande du curseur, chaque balle reste sur l'écran jusqu'à ce qu'elle soit effacée par la balle se déplaçant de droite à gauche dans la seconde partie du programme.

INPUT

Jusqu'à présent, chaque chose à l'intérieur d'un programme a été définie avant de passer. Une fois que le programme a démarré, rien ne pouvait être modifié. INPUT nous permet d'introduire de nouvelles informations dans un programme lorsqu'il est passé et fait que ces nouvelles informations agissent sur celui-ci. Pour obtenir une idée sur la façon dont INPUT fonctionne, frappez NEW RETURN et entrez ce court programme:



Ce qui arrive lorsque vous passez ce programme est simple. Un point d'interrogation apparaît indiquant que l'ordinateur attend que vous frappiez quelque chose. Entrez n'importe quel caractère ou groupe de caractères moyennant le clavier et pressez la touche RETURN. L'ordinateur répond alors par «YOU TYPED» (vous avez frappé) suivi par l'information que vous avez entrée.

Ceci peut sembler très élémentaire, mais imaginez ce que vous pouvez faire faire à l'ordinateur avec n'importe quelle information que vous entrez.

Vous pouvez entrer (INPUT) soit des variables numériques, soit des chaînes de variables et même suggérer à l'utilisateur de faire quelque

chose en raison de l'instruction INPUT sous forme d'un message. Le format d'INPUT est le suivant:

La suggestion doit être inférieure à 40 caractères. Ou seulement:

INPUT VARIABLE

REMARQUE: Pour sortir de ce programme, maintenez les touches RUN/STOP et RESTORE enfoncées.

Le programme suivant est non seulement utile mais donne une démonstration d'un grand nombre des points qui ont été présentés jusqu'à présent, y compris la nouvelle instruction d'entrée (INPUT).

```
5 PRINT"□"
10 PRINT"EDITION EN FAHRENHEIT OU CELSIUS (F/C)":INPUT A$
20 IF A$=""THEN10
30 IF A$="C" THEN100
40 IF A$="F" THEN50
45 GOTO 10
50 INPUT "ENTREE DE CELSIUS: ";C
60 F = (C*9)/5+32
70 PRINTC;"DEGRES CELSIUS = ";F;"DEGRES FAHRENHEIT"
80 PRINT
90 GOTO10
100 INPUT "ENTREE DE FAHRENHEIT: ";F
110 C = (F-32)*5/9
120 PRINTF;"DEGRES FAHRENHEIT = ";C;"DEGRES CELSIUS"
130 PRINT
140 GOTO10
```

Pas d'espace ici

Ne pas oublier d'appuyer sur RETURN

Si vous entrez et passez ce programme, vous pouvez voir l'instruction INPUT en action.

La ligne 10 utilise l'instruction INPUT (entrée) non seulement pour se procurer des informations, mais également pour afficher notre suggestion. Notez aussi que nous pouvons demander soit un nombre, soit une chaîne (en utilisant une variable numérique ou de chaîne).

Les lignes 20, 30 et 40 effectuent certains contrôles sur ce qui est frappé. A la ligne 20, si rien n'est entré (seule la touche RETURN est enfoncée), le programme revient à la ligne 10 et redemande à nouveau une entrée. A la ligne 30, si F est frappé, vous savez que l'utilisateur souhaite convertir une température de degrés Fahrenheit en degrés Celsius, si bien que le programme se branche sur la partie qui réalise cette conversion.

La ligne 40 réalise un contrôle supplémentaire. Nous savons qu'il n'y a que deux choix valables afin l'utilisateur puisse entrer. Pour passer à la ligne 40, l'utilisateur doit avoir frappé certains caractères autres que F. A présent, un contrôle est effectué pour vérifier si ce caractère est un C; dans le cas contraire, le programme demande à nouveau une entrée.

Ceci peut apparaître comme trop minutieux, mais il s'agit d'une bonne procédure de programmation.

Un utilisateur non familiarisé avec le programme peut être très frustré si celui fait quelque chose d'étrange parce qu'une erreur a été commise en entrant les informations. Après avoir déterminé quel type de conversion réaliser, le programme fait le calcul et affiche la température entrée et la température convertie. Le calcul est purement mathématique parce qu'il se sert de la formule comme pour la conversion de la température. Après la fin du calcul et l'affichage de la réponse, le programme revient à zéro et redémarre.

Après le passage du programme, l'écran se présente comme suit:

```
CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ?F
ENTER DEGREES FAHRENHEIT: 32
32 DEG. FAHRENHEIT = 0 DEG. CELSIUS

CONVERT FROM FAHRENHEIT OR CELSIUS (F/C): ?
```

Après avoir passé le programme, veillez à le sauvegarder sur disque ou bande. Ce programme, de même que d'autres présentés dans ce manuel, peuvent constituer la base de votre bibliothèque de programmes.

GET

GET vous permet d'entrer un seul caractère à la fois à partir du clavier sans appuyer sur la touche RETURN. Ceci accélère effectivement l'entrée des données dans de nombreuses applications. Quelle que soit la touche enfoncée, elle est assignée à la variable que vous spécifiez par GET. La routine suivante illustre la façon dont GET fonctionne:

```
NEW

1 PRINT "{CLR/HOME}"
10 GET A$: IF A$ = " " THEN 10
20 PRINT A$;
30 GOTO 10
```

Si vous passez (RUN) le programme, l'écran est effacé et chaque fois que vous appuyez sur une touche, la ligne 20 l'affiche sur l'écran et ensuite vous passez (GET) à un autre caractère. Il est important de noter que le caractère entré ne sera pas affiché à moins que vous ne l'affichiez (PRINT) exprès sur l'écran, comme nous l'avons fait ici. La seconde instruction sur la ligne 10 est également importante. GET agit continuellement, même si aucune touche n'est enfoncée (contrairement à INPUT qui attend une réponse), si bien que la seconde partie de cette ligne contrôle sans cesse le clavier jusqu'à ce qu'une touche soit frappée. Voir ce qui se passe si la seconde partie de la ligne 10 est éliminée.

Pour arrêter ce programme, vous pouvez appuyer sur les touches RUN/STOP et RESTORE.

La première partie du programme de conversion de température peut être facilement réécrite pour utiliser GET. Chargez (LOAD) le programme de conversion de température et modifiez les lignes 10, 20 et 40 comme indiqué ci-dessous:

```
10 PRINT"EDITION EN FAHRENHEIT OU CELSIUS (F/C)"
20 GETA$:IF A$="" THEN20
40 IF A$<>"F" THEN20
45 :
```

Cette modification rend le fonctionnement du programme plus souple étant donné que rien n'arrive avant que l'utilisateur ne frappe sur le clavier l'une des réponses souhaitées pour sélectionner le type de conversion.

Une fois que cette modification est réalisée, veillez à sauvegarder la nouvelle version du programme.

NOMBRES ALEATOIRES

Le COMMODORE 64 contient un certain nombre de fonctions qui sont utilisées pour réaliser des opérations spéciales. Les fonctions peuvent être considérées comme des programmes intégrés inclus dans le BASIC. Mais, plutôt que de frapper sur le clavier un certain nombre d'instruction chaque fois que vous avez besoin de réaliser un calcul spécialisé, il vous suffit de frapper la commande de la fonction souhaitée et l'ordinateur fait le reste.

De nombreuses fois, lors de la désignation d'un jeu ou d'un programme éducatif, vous avez besoin de générer un nombre aléatoire pour simuler par exemple un jet de dé. Vous pouvez certainement rédiger un programme qui générerait ces nombres, mais une manière plus facile consiste à appeler la fonction nombre aléatoire (RaNDom).

Pour voir ce que la fonction RND effectue effectivement, essayez ce court programme:

```
10 FOR X = 1 TO 10
20 PRINT RND(1),
30 NEXT
```

Après avoir passé le programme, vous constaterez l'affichage suivant.

```
 .789280697        .664673958
 .256373663        .0123442287
 .682952381        3.90587279E-04
 .402343724       .879300926
 .158209063       .245596701
```

Vos nombres ne correspondent-ils pas? Parfait. S'ils correspondaient, nous serions tous troublés, étant donné qu'ils doivent être complètement aléatoires!

Essayez de passer le programme encore quelques fois pour vérifier que les résultats sont toujours différents. Même si les nombres ne suivent pas un schéma quel qu'il soit, vous devez commencer à noter que certains points restent identiques chaque fois que le programme est passé. Tout d'abord, les résultats sont toujours compris entre 0 et 1, mais jamais égaux à 0 ou à 1. Ce qui n'arrivera certainement jamais si nous souhaitons simuler le jet aléatoire d'un dé, vu que nous cherchons des nombres compris entre 1 et 6.

L'autre caractéristique importante à retenir est que nous travaillons avec des nombres réels (avec des décimales). Ceci peut également être un problème, vu que des chiffres entiers sont fréquemment employés.

Il y a un grand nombre de moyens simples pour produire des nombres à partir de la fonction RND dans la plage souhaitée.

Remplacez la ligne 20 par celle qui suit et passez à nouveau le programme:

```
20 PRINT 6*RND(1),

RUN


3.60563664          4.53660853
5.47238963          8.40850227
3.19265054          4.39547668
3.16331095          5.50620749
9.32527884          4.17090293
```

Ceci a résolu le problème du manque de résultats supérieurs à 1, mais nous sommes toujours confrontés à la partie décimale du résultat. A présent, une autre fonction peut être appelée. La fonction INTeger (entier) convertit les nombres réels en valeurs entières.

Une fois de plus, remplacez la ligne 20 par celle qui suit et passez le programme pour voir l'effet de la modification.

Ceci a arrangé pas mal de choses tout en nous permettant de nous rapprocher de notre objectif d'origine, à savoir de générer des nombres aléatoires compris entre 1 et 6. Si vous examinez attentivement ce que nous avons généré cette dernière fois, vous constaterez que les résultats se situent seulement entre 0 et 6.

A la dernière étape, ajoutez un 1 à l'instruction comme suit:

```
20 PRINT INT(6*RND(1)),

RUN


2             3             1             0
2             4             5             5
0             1
```

A présent, nous avons obtenu les résultats souhaités. D'une façon générale, vous pouvez placer un nombre, une variable ou n'importe quelle expression BASIC entre parenthèses d'une fonction INT. Suivant la plage

souhaitée, il vous suffit de multiplier la limite supérieure par la fonction RND. Par exemple, pour générer des nombres aléatoires entre 1 et 25, vous devez frapper:

20 PRINT INT(25*RND(1)+1

La formule générale de génération d'un jeu de nombres aléatoires dans une certaine limite est:

NUMBER = INT((UPPER LIMIT-LOWER LIMIT)*RND(1)+LOWER LIMIT

JEU DE DEVINETTES

Vu que nous avons passé un certain temps pour comprendre les nombres aléatoires, pourquoi ne pas utiliser ces informations? Le jeu suivant n'illustre non seulement les nombres aléatoires, mais il assure une introduction à certaines théories de programmation complémentaires.

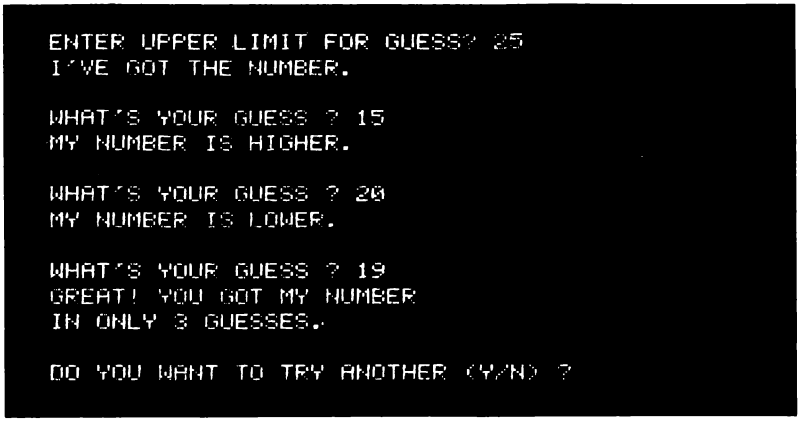
En passant ce programme, un nombre aléatoire NM sera généré.

```
1 REM JEU DE DEVINETTES
2 PRINT "□"
5 INPUT "IMPORTANCE MAXIMALE DU CHIFFRE ";LI
10 NM = INT(LI*RND(1))+1
15 CN =0
20 PRINT "COMMENCONS"
30 INPUT "VOTRE CHOIX"; GU
35 CN = CN + 1
40 IF GU > NM THEN PRINT "MON CHIFFRE EST PLUS PETIT":PRINT:GOTO 30
50 IF GU < NM THEN PRINT "MON CHIFFRE EST PLUS GRAND":PRINT:GOTO 30
60 IF GU = NM THEN PRINT "CA Y EST, VOUS AVEZ DEVINE LE CHIFFRE"
65 PRINT "EN SEULEMENT "; CN ; "ESSAIS.":PRINT
70 PRINT "UNE AUTRE CHANCE (O/N)?";
80 GET AN$: IF AN$="" THEN 80
90 IF AN$= "O" THEN 2
100 IF AN$ <>"N" THEN 80

READY.
```

Vous pouvez spécifier l'importance maximale du nombre au démarrage du programme. Ensuite, c'est à vous de deviner ce nombre.

L'exemple d'un passage suit avec une explication.



Les instructions IF/THEN comparent votre suggestion au nombre généré. Suivant votre suggestion, le programme vous indique si votre suggestion était supérieure ou inférieure au nombre aléatoire généré.

A partir de la formule donnée pour la détermination de la limite des nombres aléatoires, vérifiez si vous pouvez ajouter quelques lignes au programme qui permettent à l'utilisateur de spécifier également la limite inférieure des nombres générés.

Chaque fois que vous faites une suggestion, CN est avancé de 1 pour compter le nombre des suggestions. En utilisant le programme, essayez de vous servir de votre bon sens pour devinier le chiffre en peu d'essais.

Quand vous obtenez la bonne réponse, le programme affiche le message «GREAT! YOU GOT MY NUMBER» (félicitations, vous avez trouvé mon nombre) ensemble avec le nombre d'essais qui avaient été nécessaires. Vous pouvez alors recommencer la procédure. Rappelez-vous que le programme génère à chaque fois un nouveau nombre aléatoire.

CONSEILS DE PROGRAMMATION:

Deux points sont utilisés dans les colonnes 40 et 50 pour séparer plusieurs instructions sur la même ligne. Ceci permet non seulement des économies de frappe, mais conserve dans les longs programmes davantage d'espace dans la mémoire.

Notez également que dans les instructions IF/THEN sur les deux lignes correspondantes nous avons donné pour instruction à l'ordinateur d'afficher (PRINT) quelque chose, au lieu d'aller immédiatement sur un certain autre point du programme.

Ce dernier point illustre la raison pour laquelle l'on utilise des numéros de ligne par incrément de 10: Après que le programme a été rédigé, nous avons décidé d'ajouter la partie comptage. En ajoutant seulement ces nouvelles lignes à la fin du programme, numérotées pour être intégrées dans les bonnes lignes existantes, le programme a été aisément modifié.

JEU DE DÉS

Le programme suivant simule le jet de deux dés. Vous pouvez l'utiliser tel qu'il est ou en tant que partie d'un jeu plus important.

```
5 PRINT "TENTEZ VOTRE CHANCE"
10 PRINT "DE ROUGE  = ";INT(6*RND(1))+1
20 PRINT "DE BLANC  = ";INT(6*RND(1))+1
30 PRINT "PRESSEZ SPACE POUR D'AUTRES JETS":PRINT
40 GET A$: IF A$ = "" THEN 40
50 IF A$ = CHR$(32) THEN 10
```

Voulez-vous tenter votre chance?

Avec tout ce que vous avez appris sur les nombres aléatoires et le langage BASIC, voyons si vous pouvez suivre ce qui se passe.

GRAPHIQUES ALEATOIRES

Comme note finale au sujet des nombres aléatoires et comme introduction à la conception de graphiques, prenez un moment pour introduire et passer ce simple petit programme:

```
10 PRINT "{CLR/HOME}"
20 PRINT CHR$(205.5 + RND(1));
40 GOTO 20
```

Comme vous pouvez l'avoir supposé, la ligne 20 est ici la ligne clé. Une autre fonction, CHR$ (chaîne de caractères) vous fournit un caractère basé sur un numéro de code standard de 0 à 255. Chaque caractère que

le COMMODORE 64 peut afficher est codé de cette façon (voir annexe F). Pour trouver rapidement le code d'un caractère quelconque, il suffit de frapper:

PRINT ASC ("X")

où X est le caractère que vous contrôlez (celui-ci peut être tout caractère affichable, y compris les graphiques). La réponse est le code du caractère que vous avez frappé. Comme vous l'avez probablement remarqué «ASC» est une autre fonction qui fournit le code normalisé «ASCII» pour les caractères que vous avez frappés. Vous pouvez à présent afficher ce caractère en frappant:

PRINT CHR$(Y)

Si vous essayez de frapper:

PRINT CHR$(205); CHR$(206)

vous obtenez les deux caractères graphiques du côté droit sur les touches M et N. Il s'agit des deux caractères que le programme utilise pour le labyrinthe.

En utilisant la formule 205.5 + RND (1), l'ordinateur prélève un nombre aléatoire compris entre 205.5 et 206.5. Il y a 50% de chance que le nombre soit supérieur ou inférieur à 206. CHR$ ignore toutes les valeurs fractionnaires si bien que la moitié du temps, le caractère portant le code 205 est affiché et le reste du temps le code 206 est affiché.

Si vous souhaitez procéder à des expériences avec ce programme, essayez de changer 205.5 en ajoutant ou soustrayant un certain nombre de dixièmes de celui-ci. Ceci donnera à l'un ou l'autre caractère une plus grande probabilité d'être selectionné.

CHAPITRE 5

COMMANDES GRAPHIQUES ET COULEURS AVANCEES

- Couleurs et graphiques
- Couleurs d'affichage (PRINT)
- Codes de couleurs CHR$
- PEEK et POKE
- Encore un jeu

COULEURS ET GRAPHIQUES

Jusqu'à présent, nous avons exploré certaines des possibilités de calculs sophistiqués du COMMODORE 64. Mais l'une de ses caractéristiques les plus fascinantes est son exceptionnelle capacité de produire des couleurs et des graphiques.

Vous avez vu un court exemple de graphiques dans les programmes «balle rebondissante» et «labyrinthe». Mais ceci n'a pu qu'effleurer les performances dont vous disposez. Un certain nombre de nouveaux concepts seront introduits dans cette section pour expliquer la programmation des graphiques et des couleurs et vous indiquer comment vous pouvez créer vos propres jeux et animations avancés.

Etant donné que nous nous sommes concentrés sur les possibilités de calcul de la machine, tous les affichages que nous avons générés étaient jusqu'à présent dans une couleur unique (texte en bleu clair sur fond bleu foncé et cadre bleu clair).

Dans ce chapitre, nous allons voir la façon dont on peut ajouter des couleurs aux programmes et commander la totalité de ces symboles graphiques étranges du clavier.

AFFICHAGE DE COULEURS

Comme vous l'avez découvert si vous avez essayé le réglage des couleurs de chapitre 1, vous pouvez modifier les couleurs du texte en maintenant simplement la touche CTRL enfoncée et en appuyant sur l'une des touches de couleurs. Ceci fonctionne parfaitement dans le mode immédiat, mais que se passe-t-il si vous souhaitez intégrer des changements de couleurs dans vos programmes?

Quand nous avons présenté le programme «balle rebondissante», vous avez vu la façon dont des commandes du clavier, comme le déplacement du curseur, pouvaient être incorporées dans les instructions PRINT. De façon analogue, vous pouvez également ajouter des changements des couleurs du texte à vos programmes.

Vous disposez d'une gamme complète de 16 couleurs de texte. Les couleurs suivantes sont disponibles en utilisant la touche CTRL et une touche numérique:

1	2	3	4	5	6	7	8
noir	blanc	rouge	turquoise	pourpre	vert	bleu	jaune

Si vous maintenez la touche C= enfoncée ensemble avec la touche numérique appropriée, ces 8 couleurs supplémentaires peuvent être utilisées:

1	2	3	4	5	6	7	8
orange	brun	rouge clair	gris 1	gris 2	vert clair	bleu clair	gris 3

Frappez NEW et essayez le suivant. Maintenez la touche CTRL enfoncée et simultanément appuyez sur la touche 1. Ensuite, appuyez sur la touche R sans maintenir la touche CTRL enfoncée. A présent, tout en maintenant la touche CTRL enfoncée, appuyez simultanément sur la touche 2. Relâchez la touche CTRL et appuyez sur la touche A. Avancez dans les chiffres en alternant avec les lettres et frappez le mot COULEURS comme suit:

```
10 PRINT" C O U L E U R E S"
          ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
```


```
RUN
COULEURS
```

Tout comme les commandes du curseur qui sont affichées sous forme de caractères graphiques à l'intérieur des guillemets des instructions d'affichage les commandes de couleurs sont également représentées sous forme de caractères graphiques. Dans l'exemple précédent, lorsque vous maintenez CTRL enfoncée tout en appuyant sur 3, un «£» a été affiché. CTRL 7 a affiché une «←». Chaque commande de couleur affiche son code de couleur graphique unique lorsqu'il est utilisé de cette façon. Le tableau ci-dessous fournit les représentations graphiques de chaque commande de couleur qui peut être affichée.

CLAVIER	COULEUR	AFFICHAGE	CLAVIER	COULEUR	AFFICHAGE
CTRL 1	NOIR		C= 1	ORANGE	
CTRL 2	BLANC	E	C= 2	BRUN	
CTRL 3	ROUGE	£	C= 3	ROUGE CLAIR	
CTRL 4	TURQUOISE		C= 4	GRIS 1	
CTRL 5	POURPRE		C= 5	GRIS 2	
CTRL 6	VERT	1	C= 6	VERT CLAIR	
CTRL 7	BLEU	←	C= 7	BLEU CLAIR	
CTRL 8	JAUNE	π	C= 8	GRIS 3	

Même si l'instruction PRINT peut paraître un peu étrange sur l'écran, quand vous passez le programme, seul le texte est affiché. Et il modifiera

automatiquement les couleurs conformément aux commandes de couleurs que vous avez placées dans l'instruction PRINT.

Essayez vous-même quelques exemples en mélangeant plusieurs nombres de couleur dans une seule instruction PRINT. Rappelez-vous également que vous pouvez utiliser le second jeu de couleurs de texte en employant la touche Commodore et les touches numériques.

CONSEIL:

Vous noterez, après avoir passé un programme avec des modifications de couleurs ou de modes (inversé), que l'information «READY» et n'importe quel texte supplémentaire que vous tapez est identique à la dernière couleur ou à la dernière modification de mode. Pour revenir à l'affichage normal, rappelez-vous qu'il faut appuyer sur: RUN/STOP et RESTORE.

CODES DE COULEURS CHR$

Jetez un coup d'oeil sur l'annexe F, puis revenez à cette section.

Vous pouvez avoir noté, en observant la liste des codes CHR$ de l'annexe F, que chaque couleur (de même que la plupart des autres commandes du clavier, telles que le déplacement du curseur) possède un code unique. Ces codes peuvent être directement affichés pour obtenir les mêmes résultats qu'en appuyant sur la touche CTRL et la touche appropriée dans l'instruction PRINT.

Par exemple, essayez ceci:

```
10 PRINT CHR$(147) : REM CLR/HOME
20 PRINT CHR$(30);"CHR$(30) ME COLORE?"

READY.
```

Le texte doit à présent être vert. Dans de nombreux cas, l'utilisation de la fonction CHR$ sera beaucoup plus facile, en particulier si vous souhaitez expérimenter en changeant les couleurs. Vous trouverez à la page suivante un autre moyen pour obtenir un arc-en-ciel. Vu qu'il y a un certain nombre de lignes qui sont similaires (4-10), utilisez les touches d'édition pour sauvegarder une grande partie de ce qui est frappé. Voir les remarques après la liste pour rafraîchir votre mémoire sur les procédures d'édition.

```
1 REM BARRES DE COULEURS AUTOMATIQUES
5 PRINTCHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "     "; : REM BARRE REVERS
20 CL = INT(16*RND(1))+1
```

```
30 ON CL GOTO 40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190
40 PRINTCHR$(5):: GOTO 10
50 PRINTCHR$(28):: GOTO 10
60 PRINTCHR$(30):: GOTO 10
70 PRINTCHR$(31):: GOTO 10
80 PRINTCHR$(144):: GOTO 10
90 PRINTCHR$(156):: GOTO 10
100 PRINTCHR$(158):: GOTO 10
110 PRINTCHR$(159):: GOTO 10
120 PRINTCHR$(129):: GOTO 10
130 PRINTCHR$(149):: GOTO 10
140 PRINTCHR$(150):: GOTO 10
150 PRINTCHR$(151):: GOTO 10
160 PRINTCHR$(152):: GOTO 10
170 PRINTCHR$(153):: GOTO 10
180 PRINTCHR$(154):: GOTO 10
190 PRINTCHR$(155):: GOTO 10
```

Frappez les lignes 5 à 40 normalement. Votre écran doit afficher l'ordre suivant:

```
1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "      ";:REM  REVERSE BARS
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
40 PRINT CHR$(5):: GOTO 10
```

REMARQUES D'EDITION

Utilisez la touche CRSR-UP pour positionner le curseur sur la ligne 40. Ensuite, frappez 5 au-dessus du 4 de 40. Après quoi, utilisez la touche CRSR-RIGHT pour déplacer le curseur sur 5 dans les parenthèses CHR$. Appuyez sur les touches SHIFT INST/DEL pour ouvrir un espace et frappez «28». Maintenant, enfoncez seulement la touche RETURN avec le curseur se trouvant n'importe où sur la ligne. L'affichage doit à présent être le suivant:

```
NEW
1 REM AUTOMATIC COLOR BARS
5 PRINT CHR$(147) : REM CHR$(147)= CLR/HOME
10 PRINT CHR$(18); "      " ;:REM REVERSE BAR
20 CL = INT(8*RND(1))+1
30 ON CL GOTO 40,50,60,70,80,90,100,110
50 PRINT CHR$(28):: GOTO 10
```

Ne vous inquiétez pas. La ligne 40 est toujours présente. Listez le programme et vous le constaterez. En utilisant la même procédure, continuez à modifier la dernière ligne par un nouveau numéro de ligne et le code CHR$ jusqu'à ce que toutes les lignes restantes aient été entrées. Voyez, nous vous avons dit que les touches d'impression seraient utiles. A titre de contrôle final, lister la totalité du programme pour s'assurer que toutes les lignes ont été correctement entrées avant de passer le programme.

Vous trouverez ci-dessous une brève explication de ce qui se passe.

Vous avez probablement saisi la plus grande partie du programme des barres de couleurs jusqu'à présent, sauf une certaine nouvelle instruction étrange ligne 30. Mais voyons rapidement ce que le programme est actuellement en train de faire. La ligne 5 affiche le code CHR$ pour CLR/HOME.

La ligne 10 inverse la frappe et laisse 5 espaces qui à leur tour doivent être une barre, vu qu'ils sont inversé. La première fois, dans le programme, la barre sera bleu clair, la couleur normale du texte.

La ligne 20 utilise notre cheval de bataille, la fonction aléatoire pour sélectionner une couleur aléatoire entre 1 et 8. La ligne 30 contient une variation de l'instruction IF . . . THEN qui est intitulée ON . . . GOTO. ON . . . GOTO permet au programme de choisir où aller à partir d'une liste de numéros de ligne. Si la variable (dans ce cas CL) possède une valeur de 1, le premier numéro de ligne est celui choisi (ici 40). Si la valeur est 2, le second numéro de ligne de la liste est utilisé, etc.

Les lignes 40-110 convertissent uniquement nos couleurs clés aléatoires dans le code CHR$ approprié pour la couleur concernée et ramènent le programme à la ligne 10 pour afficher une section de la barre dans cette couleur. Ensuite, tout le processus recommence.

Regardez, si vous arrivez à produire 16 nombres aléatoires, élargissez ON . . . GOTO pour les traiter et ajoutez les codes CHR$ restants pour afficher les 8 couleurs restantes.

PEEK et POKE

Non, nous n'allons pas parler de piquer l'ordinateur, mais nous serons en mesure de jeter un coup d'oeil à l'intérieur de la machine et de «percer à jour» certaines choses en elle.

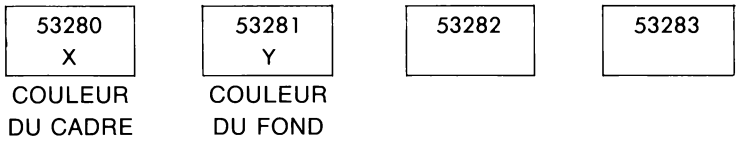
Tout comme les variables qui peuvent être considérées comme des «cases» à l'intérieur de la machine où vous avez placé vos informations, vous pouvez également imaginer certaines cases définies à l'intérieur de

l'ordinateur qui représentent des emplacements spécifiques de la mémoire.

Le COMMODORE 64 examine ces emplacements de la mémoire pour voir quels doivent être le fond de l'écran ou la couleur du cadre, quels sont les caractères qui doivent être affichés sur l'écran – et où – et une multitude d'autres tâches.

En mettant (POKEing) une valeur différente dans l'emplacement respectif de la mémoire, nous pouvons modifier des couleurs, définir et déplacer des objets et même créer de la musique.

Ces emplacements de la mémoire peuvent être représentés de la façon suivante:



Page 60, nous avons seulement représenté 4 emplacements, dont deux commandent les couleurs de l'écran et du fond.

Essayez de frapper ceci:

POKE 53281,7 RETURN

La couleur du fond de l'écran passe au jaune, étant donné que nous avons mis la valeur «7» pour le jaune dans l'emplacement qui commande la couleur du fond de l'écran.

Essayez de POKEing (mettre) différentes valeurs dans l'emplacement de la couleur du fond et observez les résultats que vous obtenez. Vous pouvez mettre n'importe quelle valeur entre 0 et 255, mais seules les valeurs de 0 à 15 fonctionnent.

Les valeurs effectives à mettre pour chaque couleur sont les suivantes.

0	noir	8	orange
1	blanc	9	brun
2	rouge	10	rouge clair
3	turquoise	11	gris 1
4	pourpre	12	gris 2
5	vert	13	vert clair
6	bleu	14	bleu clair
7	jaune	15	gris 3

Pouvez-vous imaginer un moyen d'afficher les différentes combinaisons de cadre et de fond? Les indications ci-dessous peuvent vous quelque aider peu.

```
NEW

10 FOR BA = 0 TO 15
20 FOR BO  = 0 TO 15
30 POKE 53280, BA
40 POKE 53281, BO
50 FOR X = 1 TO 2000: NEXT X
60 NEXT BO: NEXT BA

RUN
```

Deux boucles simples ont été déterminées pour POKE (mettre) différentes valeurs de manière à modifier les couleurs du fond et du cadre. La boucle DELAY (retard) de la ligne 5 ralentit un peu les choses.

Pour les curieux, essayez:

? PEEK (53280) AND 15

Vous devriez obtenir une valeur de 15. Il s'agit de la dernière valeur donnée au cadre et c'est évident parce que les deux couleurs du fond et du cadre sont grises (valeur 15) après le passage du programme. En entrant AND 15, vous éliminez toutes les autres valeurs, mises à part 1-15, en raison de la façon dont les codes de couleur sont mémorisés dans l'ordinateur. Normalement, vous devriez vous attendre à trouver la même valeur que celle qui a été POKEd (mise) en dernier lieu sur l'emplacement. En règle générale, PEEK nous permet d'examiner un emplacement spécifique et de constater quelle valeur s'y trouve actuellement. Pouvez-vous imaginer un supplément d'une ligne au programme qui afficherait la valeur du fond et du cadre quand le programme est passé? par exemple le suivant:

25 PRINT CHR$(147);"CADRE =";PEEK 53280) AND 15, "FOND = "; PEEK (53281) AND 15

GRAPHIQUES DE L'ECRAN

Dans tous les affichages d'informations que vous avez réalisés jusqu'à présent, l'ordinateur a normalement traité les informations d'une façon séquentielle: Un caractère est affiché après l'autre en commençant par l'actuelle position du curseur (sauf si vous avez demandé une nouvelle ligne ou utilisé le «,» dans le formatage PRINT).

Pour afficher (PRINT) des données dans un position particulière, vous pouvez commencer à partir d'un emplacement connu sur l'écran et

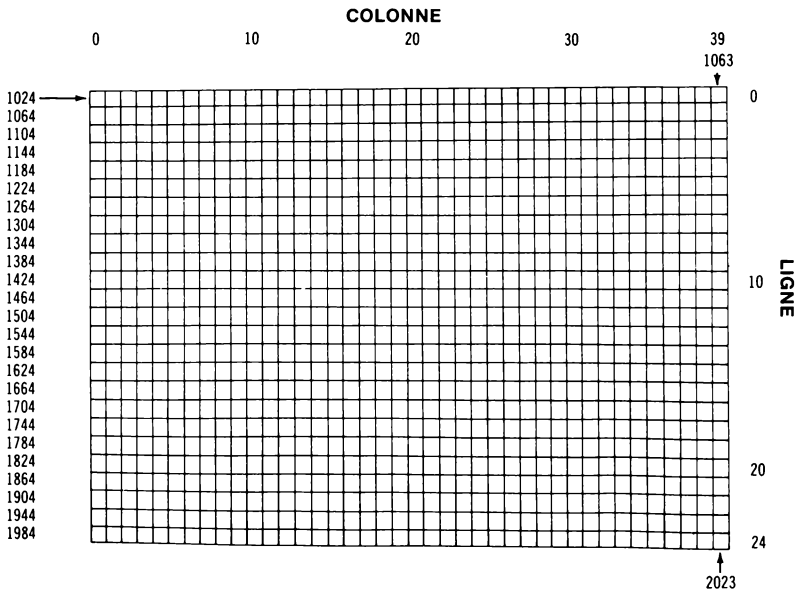
afficher (PRINT) le nombre adéquat de commandes du curseur pour formater l'image. Mais ceci nécessite plusieurs pas du programme et demande du temps.

Mais, tout comme il existe certaines positions dans la mémoire du COMMODORE 64 pour commander les couleurs, il existe également des emplacements que vous pouvez utiliser pour commander directement chaque point sur l'écran.

MEMOIRE DE L'ECRAN

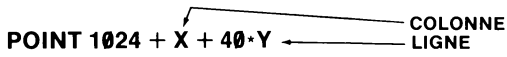
Vu que l'écran de l'ordinateur est capable de retenir 1000 caractères (colonnes de 25 lignes), 1000 emplacements de la mémoire sont réservés pour traiter ce qui est placé sur l'écran. La disposition de l'écran pourrait être considérée comme une grille, chaque carré représentant un emplacement de la mémoire.

Et vu que chaque emplacement de la mémoire peut contenir un nombre de 0 à 255, il existe 256 valeurs possibles pour chaque emplacement de la mémoire. Ces valeurs représentent les différents caractères que le COMMODORE 64 peut afficher (voir annexe E). En mettant (POKE) la valeur d'un caractère dans l'emplacement approprié de la mémoire de l'écran, ce caractère sera affiché dans la bonne position.



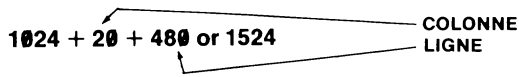
La mémoire de l'écran du COMMODORE 64 commence normalement sur l'emplacement de la mémoire 1024 et s'achève sur l'emplacement 2023. L'emplacement 1024 est le coin supérieur gauche de l'écran. L'emplacement 1025 est la position du caractère suivant à droite du précédent, etc. le long de la ligne. L'emplacement 1063 est la position la plus à droite de la première ligne. L'emplacement suivant le dernier caractère sur une ligne est le premier caractère de la ligne immédiatement dessous.

A présent, disons que vous commandez le rebondissement d'une balle sur l'écran. La balle est au centre de l'écran, colonne 20, ligne 12. La formule de calcul de la position de la mémoire sur l'écran est la suivante:



où X est la colonne et Y est la ligne.

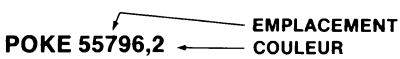
En conséquence, l'emplacement de la mémoire de la balle est:



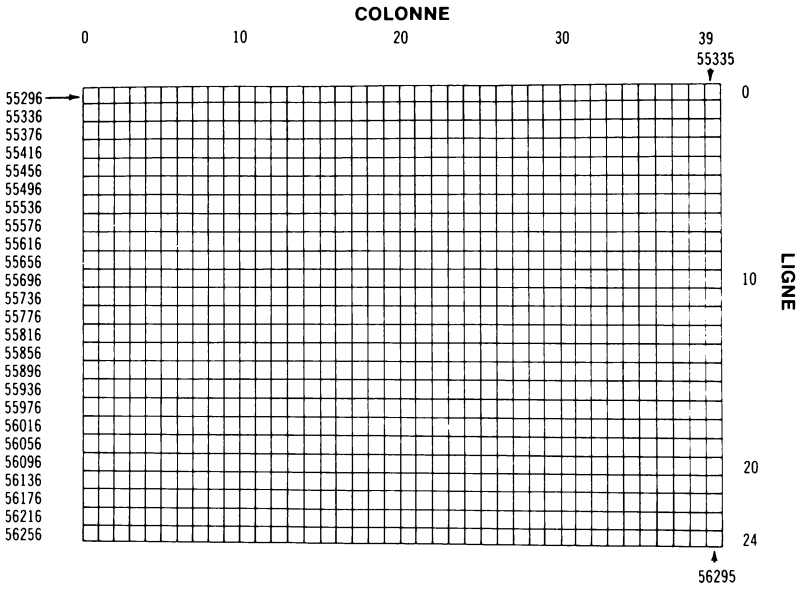
Effacez l'écran avec SHIFT et CLR/HOME et frappez:

MEMOIRE DES COULEURS

Une balle apparaît au centre de l'écran! Vous avez placé un caractère directement dans la mémoire de l'écran sans utiliser l'instruction PRINT. La balle qui apparaît était blanche. Cependant, il existe un moyen pour modifier la couleur d'un objet sur l'écran en modifiant une autre partie de la mémoire. Frappez:



La couleur de la balle passe au rouge. Pour chaque point sur l'écran du COMMODORE 64, il existe deux emplacements de la mémoire, un pour le code caractère et l'autre pour le code couleur. La carte de la mémoire des couleurs commence à l'emplacement 55296 (coin supérieur gauche) et se poursuit sur 1000 emplacements. Les mêmes codes de couleurs de 0 à 15 que ceux utilisés pour modifier les couleurs du cadre et du fond peuvent être employés ici pour directement modifier les couleurs des caractères. La formule que nous avons utilisée pour le calcul des emplacements de la mémoire de l'écran peut être modifiée pour mettre les



emplacements à la disposition des codes de couleurs POKE. La nouvelle formule est la suivante:

EMPLACEMENT DU CODE DE COULEURS = 55296 + X + 40*Y

ENCORE UN JEU

Voici un programme de rebondissements de balles révisé qui est affiché directement sur l'écran avec les POKEs plutôt que d'utiliser les commandes du curseur dans les instructions PRINT. Comme vous le constaterez après avoir passé le programme, il est beaucoup plus souple que le programme antérieur et conduit à programmer une animation plus sophistiquée.

```
NEW

10 PRINT "{CLR/HOME}"
20 POKE 53280,7 : POKE 53281,13
30 X = 1 : Y = 1
40 DX = 1 : DY = 1
50 POKE 1024 + X + 40*Y,81
60 FOR T = 1 TO 10 : NEXT
70 POKE 1024 + X + 40*Y,32
80 X = X + DX
90 IF X = 0 OR X = 39 THEN DX = -DX
```

```
100 Y = Y + DY
110 IF Y = 0 OR Y = 24 THEN DY = -DY
120 GOTO 50
```

La ligne 10 efface l'écran et la ligne 20 règle le fond sur le vert clair avec un cadre jaune.

Les variables X et Y de la ligne 30 retiennent l'actuelle position de la colonne et de la ligne de la balle. Les variables DX et DY de la ligne 40 sont les directions horizontales et verticales du déplacement de la balle. Lorsque + 1 est ajouté à la valeur X, la balle est déplacée vers la droite; lorsque −1 est ajouté, la balle se déplace vers la gauche. + 1 ajouté à Y déplace la balle sur la ligne inférieure − 1 ajouté à Y déplace la balle sur la ligne supérieure.

La ligne 50 place la balle sur l'écran dans l'actuelle position du curseur. La ligne 60 est la boucle de retard déjà familière, laissant la balle sur l'écran juste assez longtemps pour la voir.

La ligne 70 efface la balle en plaçant un espace (code 32) à l'endroit où la balle était sur l'écran.

La ligne 80 ajoute le facteur de direction à X. La ligne 90 fait un test pour vérifier si la balle a atteint l'une des limites latérales en inversant la direction s'il y a lieu.

Les lignes 100 et 110 effectuent la même chose sur les limites supérieures et inférieures. La ligne 120 renvoie le programme à l'affichage et répète les mouvements de la balle.

En modifiant le code des lignes 50 à 81 en un autre code de caractère, vous pouvez modifier la balle en tout autre caractère. Si vous modifiez DX ou DY en 0, la balle rebondit tout droit et non en diagonale.

Vous pouvez également rendre le programme un peu plus intelligent. Jusqu'à présent, la seule chose que vous avez contrôlée était la valeur de X et Y dues aux rebondissements sur l'écran.

Ajoutez la ligne suivante au programme.

```
21 FOR L = 1 TO 10
25 POKE 1024 + INT(RND(1)*1000), 166
27 NEXT L
115 IF PEEK(1024 + X + 40*Y) = 166 THEN DX = -DX:
    GOTO 80
```

Les lignes 21 à 27 placent 10 blocs sur l'écran dans des positions aléatoires. La ligne 115 (PEEK) vérifie si la balle est prête à rebondir sur un bloc et change la direction de la balle si c'est le cas.

CHAPITRE 6

GRAPHIQUES DE SYLPHES

- Introduction aux sylphes
- Création de sylphes
- Arithmétique binaire

INTRODUCTION AUX SYLPHES

Nous avons vu dans les précédents chapitres traitant des graphiques que les symboles graphiques pouvaient être utilisés dans des instructions PRINT pour créer une animation et ajouter des apparences de tableau à nos affichages.

Une façon a également été présentée de «POKE» (mettre) des codes de caractères dans des emplacements spécifiques de la mémoire de l'écran. Ceci place alors les caractères appropriés directement sur l'écran sur le coin droit. Dans ces deux cas, la création d'une animation nécessite un grand travail étant donné que des objets doivent être créés à partir de symboles graphiques existants. Le déplacement des objets nécessite un certain nombre d'instructions de programmation pour poursuivre l'objet et le déplacer sur un nouveau point. Et en raison de la restriction de l'utilisation de symboles graphiques, la forme et la résolution de l'objet peuvent ne pas être aussi bonnes qu'exigées.

L'utilisation de sylphes dans les séquences animées élimine un grand nombre de ces problèmes. Un sylphe est un objet programmable à haute résolution qui peut être constitué sous presque n'importe quelle forme – par l'intermédiaire de commandes BASIC. L'objet peut être aisément déplacé autour de l'écran en indiquant simplement à l'ordinateur la position à laquelle le sylphe doit être déplacé. L'ordinateur s'occupera du reste.

Et les sylphes sont encore beaucoup plus performants. Leur couleur peut être modifiée; vous pouvez déterminer si un objet entre en collision avec un autre; ils peuvent être constitués pour passer devant ou derrière l'un de l'autre et leur dimension peut être aisément agrandie, surtout au début.

L'inconvénient de toutes ces fonctions est minime. Cependant, l'utilisation de sylphes nécessite la connaissance de certains détails supplémentaires sur la façon dont fonctionne le COMMODORE 64 et la façon dont les chiffres sont traités par l'ordinateur. Ce n'est pas aussi difficile que l'on pourrait le croire. Il suffit de suivre les exemples et vous créerez très rapidement vos propres sylphes pour réaliser une animation.

CREATION DE SYLPHES

Les sylphes sont commandés par un synthétiseur d'image séparé du COMMODORE 64 le Vidéo interface Chip. Le synthétiseur d'image agit sur l'affichage vidéo. Il accomplit le travail difficile de création et de suivi des caractères et graphiques, de création de couleurs et du mouvement.

Ce circuit d'affichage possède 46 positions différentes ON/OFF qui agissent comme des emplacements de mémoire internes. Chacun de ces emplacements se subdivise en une série de 8 blocs. Et chaque bloc peut être soit «ON», soit «OFF». Nous en parlerons plus en détail plus tard. En POKEing (mettant) la valeur décimale appropriée sur le bon emplacement de la mémoire, vous pouvez commander la formation et le déplacement de vos créations de sylphes.

D'autre part, pour accéder à de nombreux emplacements de synthétisation de l'image, nous ferons également appel à une certaine partie de la mémoire centrale du COMMODORE 64 pour mémoriser les informations (données) qui définissent les sylphes. Finalement, huit emplacements de mémoire immédiatement après la mémoire de l'écran sont utilisés pour dire exactement à l'ordinateur de quelle zone de la mémoire chaque sylphe obtient ses données.

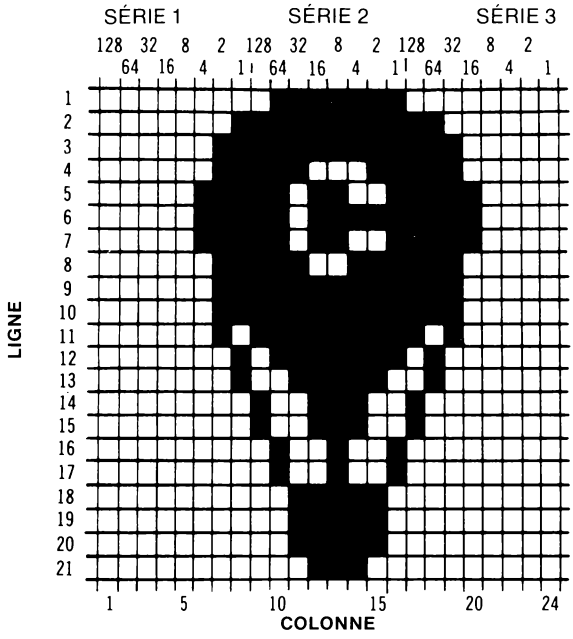
Au fur et à mesure que nous explorerons certains exemples, la procédure deviendra de plus en plus claire et vous en comprendrez le sens.

Passons maintenant à la création de certains graphiques de sylphes. Un objet de sylphe comprend 24 points de large sur 21 points de long. Jusqu'à 8 sylphes peuvent être commandés simultanément. Les sylphes sont affichés dans un mode spécial à haute résolution qui transforme l'écran en une zone d'une largeur de 320 points et d'une hauteur de 200 points.

Admettons que vous souhaitiez créer un ballon et le voir voler dans le ciel. Le ballon peut être conçu dans la grille de 24 x 21 (page 70).

La prochaine étape consiste à convertir la conception graphique en données que l'ordinateur peut utiliser. Prenez un bloc-note ou du papier millimétrique et constituez un exemple de grille de 21 espaces verticaux et 24 espaces horizontaux. En haut inscrivez 128, 64, 32, 16, 8, 4, 2, 1 trois fois (comme indiqué) pour chacun des 24 carrés. Numérotez en descendant le côté gauche de la grille de 1-21 sur chaque ligne. Inscrivez le mot DATA à la fin de chaque ligne. A présent, remplissez la grille d'un dessin quelconque ou utilisez le ballon précité. Il est plus facile d'ébaucher d'abord les contours pour ensuite revenir en arrière pour remplir la grille. A présent, imaginez tous les carrés que vous avez remplis comme étant «ON» et substituez un 1 à chaque carré rempli. Pour ceux qui ne sont pas remplis, ils sont «OFF», mettez alors un zéro.

En commençant au début de la ligne, vous avez besoin de convertir les points en trois blocs de données que l'ordinateur peut lire. Chaque ensemble de 8 carrés est égal à un bloc de données appelé un octet dans votre ballon. En partant de la gauche, les 8 premiers carrés sont blancs, ou 0, si bien que la valeur de cette série de nombres est 0.



La série centrale a l'aspect suivant (à nouveau un 1 indique un point, 0 un espace):

128	64	32	16	8	4	2	1
0	1	1	1	1	1	1	1

0 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 127

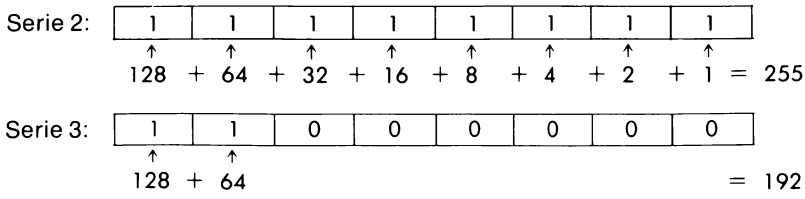
La troisième série de la première ligne contient également des blancs, si bien qu'elle est également égale à 0. En conséquence, les données de la première ligne sont:

DATA 0, 127, 0

Les séries qui constituent la ligne 2 sont calculées comme suit:

Serie 1:	0	0	0	0	0	0	0	1

1 = 1



Pour la ligne 2, les données seraient les suivantes:

DATA 1, 255, 192

De la même façon, les trois séries qui constituent chaque ligne restante seraient converties dans leur valeur décimale. Prenez le temps pour effectuer le reste de la conversion de cet exemple.

A présent que vous avez les données de votre objet, comment peut-il être utilisé?

Frappez le programme suivant et observez ce qui se passe.

```
1 REM UP, UP, AND AWAY!
5 PRINT" CLR/HOME "
10 V=53248:REM ADRESSE DE BASE DU VIC
11 POKEV+21,4:REM ACTIVATION DU SYLPHE 2
12 POKE2042,13:REM DONNEES POUR SYLPHE 2DEPUIS BLK 13
20 FORN=0TO62:READQ:POKE832+N,Q:NEXT
30 FORX=0TO200
40 POKEV+4,X:REM NOUVEAU COORDONNE X
50 POKEV+5,X:REM NOUVEAU COORDONNE Y
60 NEXTX
70 GOTO30
200 DATA0,127,0,1,255,192,3,255,224,3,231,224
210 DATA7,217,240,7,223,240,7,217,240,3,231,224
220 DATA3,255,224,3,255,224,2,255,160,1,127,64
230 DATA1,62,64,0,156,128,0,156,128,0,73,0,0,73,0,0
240 DATA62,0,0,62,0,0,62,0,0,28,0
```

Obtient cette information de DATA*

Information lue à partir de Q*

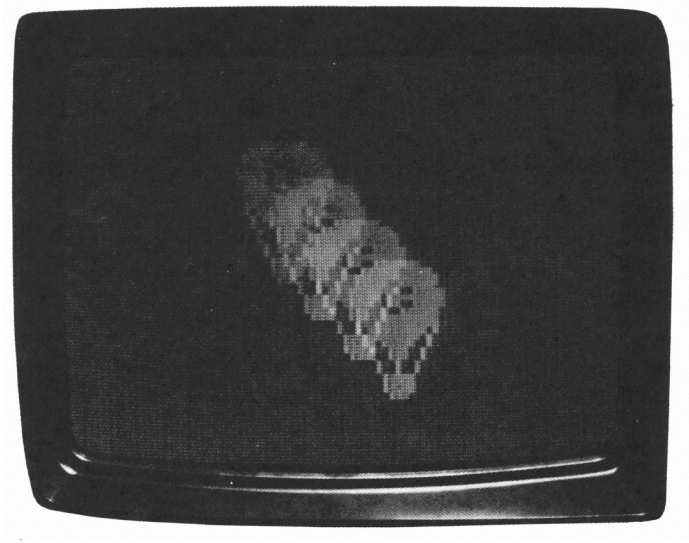
* Pour plus de détails sur READ et DATA se reporter au chapitre 8.

Si vous avez tout tapé correctement, votre ballon vole doucement dans le ciel (page 72).

Pour comprendre ce qui se passe, vous avez en premier lieu besoin de savoir quels sont les emplacements synthétisant l'image qui commandent les fonctions dont vous avez besoin. Ces emplacements intitulés registres peuvent être illustrés comme suit:

Registre	*Description*
0	Coordonnée X du sylphe 0
1	Coordonnée Y du sylphe 0
2 - 15	Appariés tel que 0 et 1 pour les sylphes 1 - 7

16	Bit le plus significatif – coordonée X
21	**Apparition du sylphe: 1 = apparaît, 0 = disparaît**
29	Dilatation du sylphe dans la direction X
23	Dilatation du sylphe dans la direction Y
39-46	Couleur du sylphe 0-7



En plus de cette information, vous avez besoin de savoir à partir de quelle section de 64 blocs chaque série de 8 blocs de mémoire obtient des sylphes dans ses données (1 série n'est pas utilisée).

Ces données sont traitées par 8 emplacements immédiatement après la mémoire de l'écran:

	2040	41	42	43	44	45	46	2047
SYLPHE	0	1	2	3	4	5	6	7

A présent, penchons-nous sur la procédure exacte qui fait bouger les objets pour ensuite écrire un programme.

Seules peu de choses sont nécessaires pour effectivement créer et déplacer un objet.

1. Faites apparaître le bon sylphe sur l'écran par «POKEing» (insertion) sur l'emplacement 21 qui fait apparaître le sylphe.
2. Réglez le pointeur de sylphe (emplacement 2040-7) sur l'emplacement d'où les données du sylphe doivent être lues.
3. POKE (mettre) les données effectives dans la mémoire.

4. Par l'intermédiaire d'une boucle, remettez à jour les coordonnées X et Y pour déplacer le sylphe.
5. Vous pouvez en option dilater l'objet, changer les couleurs ou effectuer une variété de fonctions spéciales. Utilisez l'emplacement 29 pour dilater votre sylphe dans la direction X et l'emplacement 23 dans la direction Y.

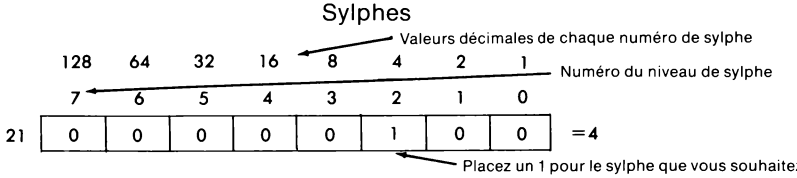
Il n'existe que peu de points du programme qui peuvent ne pas vous être familiers après les discussions tenues jusqu'à présent.

A la ligne 10
V = 53248

règle V sur l'emplacement de départ de la mémoire de la puce vidéo. De cette façon, nous augmentons seulement V par le numéro de la mémoire pour obtenir l'emplacement effectif de la mémoire. Les numéros du registre sont ceux indiqués sur la carte des registres.

A la ligne 11
POKE V + 21,4

fait apparaître le sylphe 2 en plaçant un 4 dans ce qui est appelé le registre de libération (21) pour faire apparaître le sylphe 2. Imaginez le comme suit:



Chaque niveau de sylphe est représenté dans la section 21 de la mémoire des sylphes et 4 apparaît comme étant le niveau de sylphe 2. Si vous utilisez le niveau 3, vous placez un 1 dans le sylphe 3 qui possède une valeur de 8. En fait, si vous utilisez les deux sylphes 2 et 3, vous placez un 1 aussi bien dans 4 que 8. Vous additionnez alors les nombres tout comme vous l'avez fait avec les données sur votre papier millimétré. Ainsi, les sylphes 2 et 3 sont représentés par V + 21,12.

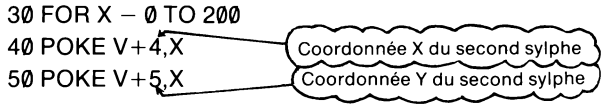
A la ligne 12
POKE 2042,13

ordonne à l'ordinateur de prendre les données du sylphe 2 (position 2042) à partir de la treizième zone de la mémoire. Vous savez, à partir de la synthétisation de votre sylphe, qu'il occupe jusqu'à 63 sections dans la

mémoire. Vous pouvez ne pas l'avoir réalisé mais les nombres que vous placez en haut de votre grille équivalent à ce que l'on appelle trois octets de l'ordinateur. En conséquence, avec les 21 lignes de votre grille multipliés par les 3 octets de chaque ligne, chaque sylphe occupe 63 octets de la mémoire.

20 FOR N = 0 TO 62: READ Q: POKE 832 + N,Q: NEXT

Cette ligne assure la création effective des sylphes. Les 63 octets de données qui représentent le sylphe que vous avez créé sont lus dans la boucle et POKEd (mis) dans le troisième bloc de la mémoire. Ceci commence par l'emplacement 832.



Si vous rappelez de l'école que la coordonnée X représente un déplacement horizontal des objets sur l'écran et que la coordonnée Y un déplacement vertical du sylphe sur l'écran, il en résulte que, lorsque les valeurs de X passent ligne 30 de 0 à 200 (un chiffre à la fois), le sylphe se déplace à travers l'écran vers le bas à droite d'un espace pour chaque chiffre. Les chiffres sont lus par l'ordinateur suffisamment rapidement pour faire apparaître le mouvement comme étant continu au lieu d'être séquentiel. Si vous avez besoin de plus de détails, jetez un coup d'oeil sur la carte de registres, annexe O. Si vous mettez en mouvement des objets multiples, il est impossible pour une seule section de la mémoire de remettre à jour les emplacements de tous ces objets. Par conséquent, chaque sylphe possède son propre jeu de 2 sections de mémoire pour se déplacer sur l'écran. La ligne 70 redémarre le cycle après un passage sur l'écran. Le reste du programme concerne les données du ballon. Elles se présentent vraiment d'une autre manière sur l'écran n'est-ce-pas?

A présent, essayez d'ajouter la ligne suivante:

25 POKE V+23,4: POKE V+29,4: REM EXPAND

et repassez le programme. Le ballon s'est dilaté à deux fois sa dimension d'origine! Ce que nous avons fait est simple. En POKEing (mettant) 4 (à nouveau pour indiquer le sylphe 2) dans les sections de mémoire 23 et 24 le sylphe 2 a été dilaté dans les directions X et Y. Il est important de noter que le sylphe démarre dans le coin supérieur gauche de l'objet. Lors de la dilatation d'un objet dans une direction ou l'autre, le point de départ reste le même. Pour rendre les choses plus intéressantes, introduisez les modifications suivantes:

```
11 POKE V+21,12
12 POKE 2042,13: POKE 2043,13
30 FOR X = 1 to 190
45 POKE V+6,X
55 POKE V+7,190-X
```

Un second sylphe (numéro 3) a été représenté en POKEing (mettant) 12 dans l'emplacement de la mémoire, ce qui fait apparaître le sylphe (V+21). Le 12 fait apparaître les sylphes 3 et 2 (00001100 = 12).

Les lignes supplémentaires 45 et 55 font faire un mouvement circulaire au sylphe 3 en POKEing (mettant) les valeurs dans les positions des coordonnées X et Y du sylphe 3 (V+6 et V+7).

Souhaitez-vous rendre le ciel plus animé? Essayez d'effectuer ces additions:

```
11 POKE V+21,28
12 POKE 2042,13: POKE 2043,13: POKE 2044,13
25 POKE V+23,12: POKE V+29,12
48 POKE V+8,X
58 POKE V+9,100
```

Ligne 11, cette fois, fait apparaître, un autre sylphe (4) en POKEing (mettant) 28 dans l'emplacement approprié «ON» de la section mémoire des sylphes. Maintenant, les sylphes 2 - 4 sont apparus (00011100 = 28). La ligne 12 indique que le sylphe 4 obtient ses données à partir de la même zone de la mémoire (13ème zone de la section 63) que les autres sylphes en POKEing (mettant) 2044,13.

Ligne 25, les sylphes 2 et 3 sont dilatés en POKEing (mettant) 12 (sylphes 2 et 3 présents) dans les emplacements de mémoire des directions X et Y dilatés (V+23 et V+29).

La ligne 48 déplace le sylphe 3 le long de l'axe X. La ligne 58 positionne le sylphe 3 à mi-course en bas de l'écran sur l'emplacement 100. Etant donné que cette valeur ne change pas, contrairement à ce qu'elle a fait auparavant avec X = 0 sur 200, le sylphe 3 ne se déplace qu'en direction horizontale.

D'AUTRES REMARQUES AU SUJET DES SYLPHES

A présent que nous avons fait des expériences avec les sylphes, quelques mots supplémentaires sont nécessaires. En premier lieu, vous pouvez modifier la couleur d'un sylphe par l'un quelconque des 16 codes de couleurs standards (0 - 15) que nous avons utilisés pour changer la couleur des caractères. Vous les trouverez dans le chapitre 5 ou dans l'annexe G.

Par exemple, pour amener le sylphe 1 sur le vert clair, frappez: POKE V + 40, 13 (veillez à régler V = 53248). Vous pouvez avoir noté, en utilisant les exemples relatifs aux programmes de sylphes, que l'objet n'est jamais déplacé sur le bord droit de l'écran. Ceci est dû au fait que l'écran comporte 320 points de large et que le registre de direction X ne peut contenir qu'une valeur de 255. Comment alors pouvez-vous réussir à faire passer un objet à travers tout l'écran?

Il existe un emplacement dans la carte de la mémoire qui n'a pas été mentionné jusqu'à présent. L'emplacement 16 (de la carte) contrôle le soi-disant bit le plus significatif MOST SIGNIFICANT BIT (MSB) de l'emplacement de direction X de sylphes. En effet, ceci vous permet de déplacer le sylphe en direction horizontale entre 256 et 320.

Le bit le plus significatif du registre X fonctionne comme suit: Après que le sylphe a été déplacé sur l'emplacement X 255, introduisez une valeur dans l'emplacement de la mémoire 16 représentant le sylphe que vous souhaitez déplacer. Par exemple, pour obtenir que 2 se déplace sur les emplacements horizontaux 256 - 230, POKE (mettre) la valeur du sylphe 2 qui est (4) dans l'emplacement de la mémoire 16:

POKE V+16,4

A présent, commencez le déplacement dans le registre de direction X usuel du sylphe 2 (qui est sur l'emplacement 4 de la carte) en commençant à nouveau par 1. Vu que vous ne vous déplacez que de 64 autres espaces, les emplacements X ne doivent cette fois n'être compris qu'entre 0 et 63. Tout ce concept est mieux illustré par une version du programme original du sylphe 1:

```
10 V= 53248 : POKE V+21,4 : POKE 2042,13
20 FOR N = 0 TO 62 : READ Q : POKE 832+N,Q : NEXT
25 POKE V+5, 100
30 FOR X = 0 TO 255
40 POKE V+4,X
50 NEXT
60 POKE V+16,4
70 FOR X = 0 TO 63
```

```
80 POKE V+4, X
90 NEXT
100 POKE V+16,0
110 GOTO 30
```

La ligne 60 détermine le bit le plus significatif du sylphe 2. La ligne 70 démarre le mouvement de l'emplacement X de la direction standard, déplaçant la sylphe 2 sur le reste du chemin à travers l'écran.

La ligne 100 est importante parce qu'elle «coupe» le bit le plus significatif de telle manière que le sylphe puisse recommencer à se déplacer à partir du bord gauche de l'écran.

ARITHMETIQUE BINAIRE

Des informations détaillées sur la façon dont l'ordinateur traite les nombres dépasseraient le cadre de ce manuel d'introduction. Cependant, nous allons vous fournir une base pour la compréhension du processus et vous permettre de vous lancer dans l'animation sophistiquée. Mais, avant d'aller plus en avant, nous devons définir certains termes:

BIT – Il s'agit de la plus petite quantité d'informations qu'un ordinateur puisse mémoriser.

Imaginez un BIT comme un interrupteur qui est soit fermé, soit ouvert (ON ou OFF). Lorsqu'un bit est «fermé» (ON), il a une valeur de 1. Lorsqu'un bit est «ouvert» (OFF), il a une valeur de 0.

Après le BIT, le niveau suivant est l'OCTET (BYTE).

OCTET (BYTE) – Celui-ci est défini comme étant une série de bits. Vu qu'un OCTET est constitué de 8 BITS, vous pouvez effectivement avoir un total de 255 combinaisons différentes de BITS. En d'autres termes, vous pouvez avoir tous les BITS sur «off» si bien que votre OCTET a l'aspect suivant:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0

et sa valeur sera nulle. Si tous les bits sont sur «on», on obtiendra:

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

ce qui équivaut à 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

La prochaine étape est intitulée un REGISTRE.

REGISTRE

Il est défini comme étant un bloc d'octets enchaînés. Mais, dans ce cas, chaque registre n'a en fait la longueur que d'un octet. Une série de registres constitue une carte de registres. Les cartes de registres sont des tableaux identiques à celui que vous avez utilisé pour constituer votre sylphe «ballon». Chaque registre commande une fonction différente, telle que l'apparition du sylphe est en fait intitulée ENABLE REGISTER (registre libération). Pour rendre le sylphe plus long, on utilise l'EXPAND X REGISTER, alors que pour rendre le sylphe plus large, on utilise l'EXPAND Y REGISTER. Rappelez-vous qu'un registre est un octet qui assure une tâche spécifique.

Passons à présent au reste de l'arithmétique binaire.

CONVERSION BINAIRE EN DECIMALE

VALEUR DÉCIMALE								
128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	1	2↑0
0	0	0	0	0	0	1	0	2↑1
0	0	0	0	0	1	0	0	2↑2
0	0	0	0	1	0	0	0	2↑3
0	0	0	1	0	0	0	0	2↑4
0	0	1	0	0	0	0	0	2↑5
0	1	0	0	0	0	0	0	2↑6
1	0	0	0	0	0	0	0	2↑7

En utilisant des combinaisons de 8 bits, vous pouvez obtenir n'importe quelle valeur décimale comprise entre 0 et 255. Commencez-vous à comprendre pourquoi, lorsque nous POKEd (mettions) un caractère ou des valeurs de couleurs dans des emplacements de la mémoire, les valeurs devaient être comprises entre 0 et 255? Chaque emplacement de mémoire peut contenir un octet d'informations.

N'importe quelle combinaison possible de huit 0 et 1 est convertie en une valeur décimale unique comprise entre 0 et 255. Si tous les emplacements contiennent un 1, la valeur de l'octet est égale à 255. Tous les zéros équivalent à une valeur d'octet de 0; «00000011» équivaut à 3, etc. C'est la base pour la création des données qui représente les sylphes et leur manipulation. Considérons par exemple que ce groupement d'octets représente une partie d'un sylphe (0 est un espace, 1 est une zone de couleur):

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
1	1	1	1	1	1	1	1	
128 +	64 +	32 +	16 +	8 +	4 +	2 +	1 +	= 255

Alors nous mettons (POKE) 255 dans l'emplacement approprié de la mémoire pour représenter cette partie de l'objet.

CONSEIL:

Pour vous épargner l'inconvénient de convertir les nombres binaires en valeurs décimales – ce qui est fréquemment nécessaire-le programme suivant réalise le travail pour vous. Nous vous conseillons d'entrer et de sauvegarder le programme pour vous en servir plus tard.

```
5 REM CONVERSION BINAIRE EN DECIMALE
10 INPUT"ENTREE CHIFFRE BINAIRE DE 8 BITS :";A$
12 IFLEN(A$)<>8THENPRINT"8 BITS S.V.P...":GOTO10
15 TL=0:C=0
20 FORX=8TO1 STEP-1:C=C+1
30 TL=TL+VAL(MID$(A$,C,1))*2↑(X-1)
40 NEXTX
45 PRINT
50 PRINTA$;" BINAIRE ";" = ";TL;" DECIMALE"
55 PRINT
60 GOTO10
```

Ce programme prend votre nombre binaire qui a été entré sous la forme d'une chaîne et analyse chaque caractère de la chaîne de gauche à droite (la fonction MID$). La variable C indique sur quel caractère agir lorsque le programme passe dans la boucle. La fonction VAL ligne 30 établit la valeur effective du caractère. Vu que nous traitons caractères numériques, la valeur est la même que le caractère. Par exemple, si le premier caractère de A$ est 1, la valeur sera également 1.

La partie finale de la ligne 30 multiplie la valeur du caractère en cours par la puissance 2. Vu que la première valeur est l'emplacement 2↑7 dans l'exemple, TL devrait en premier lieu être égal à une fois 128 ou 128. Si le bit est zéro, la valeur de cet emplacement devrait également être zéro.

Cette procédure est répétée pour la totalité des 8 caractères étant donné que TL conserve la trace du passage de la valeur décimale du nombre binaire.

CHAPITRE 7

CREATION DE SONS AVEC LE COMMODORE 64

- La structure d'un programme sonore
- Exemple d'un programme sonore
- Réglages importants du son
- Effets sonores

La création de sons à l'aide d'un ordinateur s'oriente à deux champs d'application principaux: jouer de la musique et générer des effets sonores. Nous allons voir maintenant brièvement comment un programme sonore est en général structuré, pour ensuite présenter un programme qui vous permettra de faire des essais.

LA STRUCTURE D'UN PROGRAMME SONORE

La tonalité d'un son est déterminée par quatre éléments différents: **la hauteur du son, le volume du son, le timbre** et **le taux d'appui/relâchement.** Les dernières deux caractéristiques nous permettent de distinguer différents instruments de musique par l'ouie. Vous désirez donc certainement d'influencer ces caractéristiques importantes par votre programme.

Dans ce but, votre COMMODORE 64 possède de nouveau SON propre composant électronique construit à cet effet, appelé le SID (Sound Interface Device). Le SID contient un certain nombre de zones de mémoire pour les paramètres qui permettent de synthétiser le son. Vous savez déjà que votre COMMODORE 64 peut créer trois voix en même temps; voyons d'abord la première de ces trois voix.

L'adresse de base du SID – abrégée analogue au chapitre précédent par la variable SI – est le 54272:

SI = 54272

La hauteur du son est définie dans la physique par la **fréquence.** La fréquence est mémorisée dans le SID par un paramètre qui peut adopter des valeurs entre presque zéro et 65000. Vous avez appris dans le chapitre précédent que des chiffres d'une telle importance ne peuvent être mémorisés dans une seule cellule de mémoire, nous devons donc diviser le paramètre de fréquence en haute fréquence (octet plus élevé) et basse fréquence (octet moins élevé), le plus souvent désigné par Hi-(High)-Byte et Lo-(Low)-Byte.
Ces deux octets occupent les deux premiers registres du SID:

FL = SI **(basse fréquence)**
FH = SI+1 **(haute fréquence)**

Le volume du son dispose de 16 niveaux dans le SID, de zéro (éteint) à 15 (plein volume). Le paramètre correspondant est enregistré dans le registre 24:

L = SI+24 **(volume du son)**

C'était encore facile. Nous voilà **au timbre:** il est en large mesure défini par la forme **d'ondes** qui sont produites par l'instrument de musique respectif. Le COMMODORE 64 vous propose quatre formes fondamentales: triangle, dent de scie, rectangle (impulsion) et bruit. Vous trouverez dans le manuel de programmation les instructions à savoir comment modifier de manière sophistiquée les formes de base, ou comment les influencer par les filtres. Il nous suffit pour l'instant de connaître les formes de base: chacune d'elles est contrôlée par un bit dans le registre 4:

W = SI+4 (forme d'ondes)

Pour sélectionner les formes de base citées ci-dessus, vous écrivez dans ce registre l'un des paramètres suivants: 17, 33, 65 et 129. Si vous choisissez le 65 (rectangle/impulsion), vous devez encore fixer un paramètre supplémentaire entre zéro et 4095 concernant le soi-disant **taux d'impulsions** (le taux de votre rectangle entre «allumé» et «éteint»): les deux octets de ce paramètre sont placés dans les registres 2 et 3:

TL = SI+2 (taux d'impulsions, basse fréquence)
TH = SI+3 (taux d'impulsions, haute fréquence)

Maintenant encore le plus captivant: le **taux d'appui/**relâchement, **l'évolution** du son. Votre COMMODORE 64 fait monter d'abord chaque son jusqu'au volume réglé dans le registre 24, puis le son descend de nouveau légèrement pour enfin conserver son volume ainsi atteint tant que vous le laissez allumé, puis, il cesse de résonner. Comme vous pouvez le constater, quatre paramètres participent à cette «courbe enveloppante» qui sont supervisés par le SID dans deux autres registres:

A = SI+5 (taux d'appui/relâchement)
H = SI+6 (maintien)

Chacun de ces deux registres est épissé en deux parties (voir encore une fois l'introduction dans l'arithmétique binaire dans le chapitre précédent): Le paramètre des quatre bits plus élevés de A règle **la montée** du son, celui des quatre bits moins élevés **la descente:** les petits chiffres signifient rapide, dure; les chiffres élevés lent, doux. Ceci est également valable pour les quatre bits moins élevés de H qui contrôlent **la résonance** du son quand il est éteint. Les quatre bits plus élevés de H déterminent **le volume** du son au niveau duquel il est maintenu. De la valeur la plus élevée résulte le volume du son réglé au préalable dans le registre 24, les valeurs moins élevées réduisent plus ou moins le volume du son.

EXEMPLE D'UN PROGRAMME

Tout d'abord vous devez décider quelles **voix** (ou générateurs de sons) vous voulez employer. Suivant les voix choisies, il faut alors déterminer les quatre réglages mentionnés ci-dessus (volume du son, forme etc.) pour chacune de ces voix. Vous avez la possibilité d'utiliser jusqu'à trois voix simultanément, notre exemple ne porte cependant que sur la **voix no. 1.**

10 SI=54272: FL=SI: FH=SI+1: W=SI+4: A=SI+5: H=SI+6: L=SI+24	1. Définition des registres d'adresses
20 POKE L,15	2. Plein volume du son
30 POKE A,16+9	3. Taux d'appui
40 POKE H,4*16+4	4. Maintien et résonance
50 POKE FH,29:POKE FL,69	5. Haute et basse fréquence, ici pour le «la» du diapason. Les valeurs des autres sons sont indiquées dans la liste de l'annexe P.
60 POKE W,17	6. Forme d'ondes. Doit toujours être réglée en dernier, car le bit le plus bas allume resp. éteind le générateur de sons dans ce registre.
70 FORT=1T0500:NEXT	7. Boucle pour régler la durée du son.
80 POKE W,0:POKE A,0: POKE H,0	8. Eteindre des réglages de formes d'ondes et de courbes enveloppantes.

Après avoir frappé RUN, vous pouvez écouter la note qui est créée au moyen du présent programme.

JOUER DES MÉLODIES AVEC LE COMMODORE 64

Vous n'avez pas besoin d'être un musicien pour créer de la musique sur votre COMMODORE 64. Voilà l'exemple d'un programme qui vous montrera comment faire. Nous employons de nouveau une seule voix des trois voix à disposition.
Effacez le programme antérieur par NEW et frappez ce qui suit:

10 REM ECHELLE	Nom du programme
20 SI=54272:FL=SI:FH=SI+1: E=SI+4:A=SI+5:H=SI+6: L=SI+24	Définition des registres d'adresses
30 POKE L,15	Plein volume du son
40 POKE A,9	Taux d'appui
50 READ X:READ Y	Lire les Hi-Byte et Lo-Byte de la fréquence des lignes DATA 130, 140
60 IFY=−1THEN POKE W,0:END	Lorsque le programme lit –1, il doit s'éteindre
70 POKE FH,X:POKE FL,Y	Mettre (poke) les Hi-Byte et Lo-Byte dans les registres des fréquences (haute/basse fréquence)
80 POKE W,17	Forme d'ondes, allumer le générateur
90 FORT=1TO100:NEXT	Durée du son
100 POKE W,0	Eteindre le générateur
110 FORT=1T050:NEXT	Brève pause pour cesser de résonner
120 GOTO40	Prochain son
130 DATA17,103,19,137,21,237, 23,59,26,20,29,69,32,219,34,207	Ces paires de chiffres représentent les sons de la gamme majeure do (C), toujours une haute et une basse fréquence à tour de rôle
140 DATA−1,−1	Ces données (n'ayant pas de sens comme fréquences) signalent au programme dans la ligne 60 que la gamme musicale est achevée.

Pour produire des sons ressemblant à un clavecin, changer la ligne 80 pour lire:

POKE W,33

Par cette commande POKE, la forme «dent de scie» est sélectionnée qui provoque des sons plus «aigus», caractérisés par davantage de tons élevés que ceux employés jusqu'à présent de forme «triangulaire». Mais le choix de la forme n'est qu'une des possibilités à disposition pour définir le caractère du son. Nous pouvons également modifier le taux d'appui/relâchement pour passer du son d'un clavecin à un son de «banjo». C'est la commande POKE dans la ligne 40 qui en est responsable:

POKE A,3

Comme vous venez de le voir, vous pouvez obtenir des sons identiques à ceux de différents instruments de musique – comme par un véritable synthétiseur. Comment faire, c'est-à-dire comment modifier les registres, vous en trouverez les détails ci-après.

RÉGLAGES IMPORTANTS DE LA TONALITÉ

1. Volume du son – Le choix du volume concerne tous les 3 générateurs de sons du COMMODORE 64. Le registre en question lit l'adresse 54296. Vous obtenez le volume maximal quand vous mettez (poke) dans le registre le chiffre 15:

POKE L, 15 ou POKE 54296,15

Pour éteindre les générateurs de sons, vous écrivez 0 dans le registre:

POKE L,0 ou POKE 54296,0

En principe, vous fixez le volume du son au début d'un programme de musique; mais vous pouvez aussi créer des effets intéressants grâce à des modifications de programmation du volume du son.

2. Forme d'ondes – Comme vous venez de le constater dans notre exemple, les formes d'ondes définissent en large mesure le caractère du son. Pour chaque voix du COMMODORE 64, vous pouvez régler les formes indépendamment.
Pour ce faire, vous avez le choix entre la forme triangulaire, dent de scie, rectangulaire (impulsion) et bruit. Vous trouverez les adresses respectives et leurs contenus qui correspondent aux différentes voix et formes d'ondes dans la liste ci-dessous. Si vous désirez p. e. choisir la forme «triangulaire» pour la première voix, vous devez entrer la commande suivante:

POKE W,17 ou POKE 54276,17

Le premier chiffre (adresse) représente alors le registre et le second (contenu de l'adresse ou du registre) détermine la forme respective.

REGLAGES DES FORMES D'ONDES

	REGISTRE			CONTENU			
VOIX	1	2	3	BRUIT	RECTANGLE	DENT DE SCIE	TRIANGLE
	4	11	18	129	65	33	17

Nous avons utilisé ce tableau dans la ligne 30 de notre programme sur la gamme de sons. Moyennant Poke SI+4,17, nous avons sélectionné la

forme triangulaire que nous avons remplacée en modifiant le caractère du son par une «dent de scie» en changeant le 17 en 33.

Maintenant, voyons comment modifier la courbe enveloppante qui définit le développement du volume dans le cadre du son même. Pensez-y, un son est seulement produit si vous déterminez, comme expliqué ci-dessus, son volume et sa forme.

3. Réglage de la courbe enveloppante – Les valeurs d'appui/relâchement qui peuvent être choisies, comme dans la forme d'ondes, pour chaque voix séparément, sont représentées ensemble par une valeur numérique. Tandis que le paramètre d'appui indique le temps au cours duquel le son monte jusqu'à son volume maximal (réglé d'avance), le paramètre de relâchement mesure la vitesse à laquelle le volume du son descend jusqu'au niveau d'arrêt. Si le niveau d'arrêt 0 a été choisi, le paramètre de relâchement indique la durée de descente jusqu'au volume 0 pour ainsi definir la durée du son. Les adresses correspondant aux différentes voix ainsi que les valeurs sélectionnées des taux d'appui/relâchement sont additionnées et la somme ainsi obtenue est mise (poke) dans le registre concerné.

REGLAGES DES TAUX D'APPUI/RELÂCHEMENT

	REGISTRE			CONTENU	
VOIX	1	2	3	APPUI	RELÂCHEMENT
	5	12	19	15*16 (doux) ... 0*16 (dure)	15 (doux) ... 0 (dure)

Si vous réglez un seul taux d'appui, p. e. en mettant (POKE) 54277,64, le taux de relâchement est automatiquement nul (et vice versa). Par POKE 54277,66, l'appui est réglé sur un taux moyen (64=4*16) et le relâchement sur une basse valeur (2), la valeur 66 représente donc la somme de 64 et 2. Il est plus facile de reconnaître les différents éléments quand l'on se sert de POKE A,4*16+2 (à condition que le registre d'adresse A soit défini auparavant) au lieu de l'expression POKE 54277,66.

Maintenant, un programme-type vous illustrera mieux cet effet. Frappez le mot NEW, appuyez sur la touche RETURN et entrez le programme suivant:

```
10 REM PROGRAMM A EXPERIMENTER
20 SI=54272: FL=SI: FH=SI+1: TL=SI+2:
   TH= SI+3: W=SI+4: A=SI+5: H=SI+6:
   L= SI+24
30 PRINT"PRESSEZ UNE TOUCHE!"        Message du programme
40 GETZ$:IFZ$=""THEN40               Avez-vous appuyé sur la touche?
```

```
50 POKE L,15                    Volume du son
60 POKE A,1*16+5                Taux d'appui/relâchement
70 POKE H,0*16+0                Maintien et résonance
80 POKE TH,8: POKE TL,0         Taux d'impulsions
90 POKE FH,14: POKE FL,162      Fréquence
100 POKE W,17                   Forme d'ondes, allumez le générateur
110 FORT=1TO200:NEXT            Durée du son
120 POKE W,0                    Eteindre le générateur
130 GOTO40                      Recommencez le tout
```

Nous avons utilisé la voix 1 pour créer un son d'une courte montée et d'une courte descente après avoir atteint le volume maximal (ligne 60!). Le résultat ressemble à peu près au rebondissement d'une balle dans un fût d'essence. Pour obtenir un autre son, nous modifions cette ligne.

Nous arrêtons alors le programme par RUN STOP, faisons afficher le programme par LIST (en appuyant ensuite sur la touche RETURN) et changeons la ligne 60 de la manière suivante:

```
60 POKE A,11*16+14
```

L'ordinateur reprendra cette ligne modifiée dans sa mémoire dès que la touche RETURN est frappée.

Le son ainsi obtenu fait penser aux hautbois ou tout autre instrument de type flûte. Faites des essais tout en changeant la forme et la courbe enveloppante pour avoir comment les différentes valeurs de ces paramètres modifient le caractère du son.

Le réglage du maintien détermine le volume du son qui est conservé après l'appui. Comme d'habitude, nous nous servons d'une boucle FOR . . . NEXT pour régler la durée du son. Tout comme avant dans le registre précédent, le maintien et la résonance du son sont définis par un chiffre qui est le résultat de l'addition des valeurs figurant dans la liste ci-dessous:

MAINTIEN/RESONANCE

	REGISTRE			CONTENU	
VOIX	1	2	3	MAINTIEN	RÉSONANCE
	6	13	20	15*16 (fort) ... 0*16 (muet)	15 (lent) ... 0 (rapide)

Remplacez les zéros dans la ligne 70 par n'importe quelle valeur jusqu'à 15 au maximum et écoutez le résultat!

4. Le choix des voix et des notes – Comme déjà mentionné, il faut entrer deux valeurs pour créer un son que nous avons appelées «haute» et «basse» fréquence. L'attribution de ces valeurs notes ressort du tableau de l'annexe P.

Etant donné que les voix sont affectées à différentes adresses (voir tableau ci-après), vous pouvez programmer indépendamment les trois voix de votre COMMODORE 64 et créer ainsi p. e. des pièces de musique à 3 voix.

ADRESSES DES TROIS GENERATEURS DE SONS ET LES VALEURS POKE HAUTE ET BASSE FREQUENCE DES SONS DE L'OCTAVE MOYENNE (5 e)

	REGISTRE			CONTENUS POUR LES NOTES – 5 E OCTAVE												
VOIX	1	2	3	C	C#	D	D#	E	F	F#	G	G#	A#	H#	H	C
HAUTE-FREQ.	1	8	15	35	37	39	41	44	46	49	52	55	58	62	66	70
BASSE FREQ.	0	7	14	3	24	77	163	29	188	132	117	148	226	98	24	6

Pour créer un do (C) avec la voix 1, vous devez employer les commandes POKE suivantes:

POKE 54273,35:POKE 54272,3
ou **POKE SI+1,35:POKE SI,3**

Le même son avec la voix 2 est créé par:

POKE 54280,35:POKE 54279,3
ou **POKE SI+8,35:POKE SI+7,3**

JOUER UNE CHANSON SUR LE COMMODORE 64

Le programme suivant permet de «composer» ou de jouer une chanson. L'ordinateur se sert de la voix 1. Notez que dans la ligne 110 du programme, les adresses des registres souvent utilisés sont attribuées à des variables numériques ce qui facilite leur application. Si p. e. la forme d'ondes est choisie, il suffit de remplacer dans la commande POKE correspondante le chiffre 54276 par la lettre W.

De même, notez la façon dont les lignes DATA sont utilisées. Ceci pour vous faciliter dorénavant leur emploi dans vos propres programmes.

Le présent programme mémorise successivement dans les lignes DATA les trois chiffres nécessaires pour décrire un certain son. Il s'agit de la haute et basse fréquence ainsi que de la **durée du son.**

La durée du son est déterminée par une boucle de 1 jusqu'à la troisième position des lignes DATA. 125 correspond alors à une croche, 250 à une noire, 375 à une noir avec point, 500 à une blanche et 1000 à une ronde. Selon le rythme ou votre goût en matière de musique, ces valeurs peuvent être augmentées ou diminuées.

Prenons le ligne 110, le 17 et le 103 représentent la haute et basse fréquence pour le do (C) et du chiffre 250 à la troisième position résulte une noire. La seconde note est aussi une noire, mais cette fois le mi (E) etc.
Il est également possible d'introduire dans les lignes DATA des valeurs choisies par vous-mêmes dans la liste des notes de l'annexe P. La longueur de votre mélodie n'est limitée que par l'emplacement de mémoire disponible dans votre COMMODORE 64. Il suffit de veiller à ce que l'expression DATA–1,–1,–1 figure dans la dernière ligne du programme. Puis, la ligne 130 achève le programme quand il a atteint cette ligne.

```
10 REM MICHAEL ROW THE BOAT ASHORE
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKEL,15:POKETH,13:POKETL,15:POKEA,3*16+5:POKEH,9
40 READX:READY:READD
50 IFX=-1THENEND
60 POKEFH,X:POKEFL,Y
70 POKEW,65
80 FORT=1TOD:NEXT
90 POKEW,0
100 GOTO40
110 DATA17,103,250,21,237,250,26,20,400,21,237,100,26,20,250,29,69,250
120 DATA26,20,250,0,0,250,21,237,250,26,20,250,29,69,1000,26,20,250,0,0,250
130 DATA-1,-1,0
```

CRÉATION D'EFFETS SONORES

Contrairement aux effets musicaux, les effets purement sonores soulignent généralement les «actions» qui se passent sur l'écran (l'explosion d'un vaisseau spatial etc.) ou ils doivent informer ou prévenir l'utilisateur d'un programme (p. e. qu'il est sur le point d'effacer sa disquette etc.).
Voici quelques exemples pour faire des essais:

1. Modifiez le volume du son pendant qu'une note est jouée, vous pouvez par exemple produire ainsi l'effet d'écho.
2. Faites des «sauts» rapides entre deux hauteurs de sons pour créer un «trémolo».
3. Essayez différentes formes d'ondes.
4. Analysez les détails de la courbe enveloppante.

5. Grâce à la programmation variable des trois voix (p. e. faire durer le son d'une certaine un peu plus longtemps que celui d'une autre voix) des effets sonores étonnants sont possibles.
6. Employez la forme rectangulaire (impulsion) et modifiez la gamme d'impulsions (annexe O).
7. Essayez les générateurs de bruit pour créer des explosions, coups de fusil ou pas etc.
8. Variez les hautes/basses fréquences selon une succession rapide sur plusieurs octaves.

EXEMPLES D'EFFETS SONORES POUR LES DÉMONSTRATIONS

Les programmes suivants peuvent être intégrés, tels quels ou étant modifiés, dans n'importe quel programme BASIC. Ils sont inclus ici pour vous inciter à faire vos propres essais et pour illustrer la gamme d'effets sonores à disposition sur votre COMMODORE 64.

```
10 REM POUPEE
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 POKEL,15:POKETH,15:POKETL,15:POKEA,0*16+0:POKEH,15*16
40 POKEW,65
50 FORX=250TO0STEP-2:POKEFH,40:POKEFL,X:NEXT
60 FORX=150TO0STEP-4:POKEFH,40:POKEFL,X:NEXT
70 POKEW,0

10 REM COUP DE FEU
20 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
30 FORX=15TO0STEP-1
40 POKEL,X:POKEA,15:POKEH,0:POKEFH,40:POKEFL,200:POKEW,129
50 NEXT
60 POKEW,0:POKEA,0

10 REM MOTEURS
20 SI=54272
30 FORK=0TO24:READX:POKESI+K,X:NEXT
40 DATA9,2,0,3,0,0,240
50 DATA12,2,0,4,0,0,192
60 DATA16,2,0,6,0,0,64
70 DATA0,30,243,31:REM FILTRE
80 POKESI+4,65:POKESI+11,65:POKESI+18,65
```

CHAPITRE 8

TRAITEMENT AVANCE DES DONNEES

- READ et DATA
- Moyennes
- Variables à indices
  Tableaux à une entrée
- DIMENSION
- Jets de dés simulés
- Tableaux à deux entrées

READ ET DATA

Vous avez vu comment assigner des valeurs à des variables directement à l'intérieur du programme (A=2) et comment assigner différentes valeurs pendant que le programme est en cours – par l'instruction INPUT. Il se présente de nombreux cas où ni l'une, ni l'autre de ces façons ne s'adapte parfaitement au travail que vous essayez de réaliser, en particulier s'il relie un grand nombre d'informations.

Essayez ce court programme:

```
10 READX
20 PRINT"X EST MAINTENANT :";X
30 GOTO10
40 DATA 1,34,10.5,16,234.56

RUN

X IS NOW : 1
X IS NOW : 34
X IS NOW : 10.5
X IS NOW : 16
X IS NOW : 234.56

?OUT OF DATA ERROR IN 10
READY
```

A la ligne 10, l'ordinateur lit (READ) une valeur de l'instruction DATA et assigne cette valeur à X. Chaque fois, dans la boucle, la valeur suivante de l'instruction DATA est lue et cette valeur est assignée à X et affichée (PRINT). Un pointeur de l'ordinateur conserve automatiquement la trace de la valeur qui est utilisée plus tard.

POINTEUR ↓

40 DATA 1, 34, 10.5, 16, 234.56

Lorsque toutes les valeurs ont été utilisées et que l'ordinateur a de nouveau exécuté la boucle, en cherchant une autre valeur, l'erreur OUT OF DATA (hors des données) a été affichée parce qu'il n'y avait plus de valeurs à lire (READ).

Il est important de suivre précisément le format de l'instruction DATA:

40 DATA 1, 34, 10.5, 16, 234,56

↑ Virgule qui sépare chaque position

↑ Pas de virgule

Les instructions DATA peuvent contenir des nombres entiers, des nombres réels (234,65), ou des nombres exprimés en notation scientifique. Mais vous ne pouvez lire d'autres variables ou prévoir des opérations arithmétiques dans les lignes DATA. Ceci serait incorrect:

40 DATA A, 23/56,2*5

Cependant, vous pouvez utiliser une variable chaîne dans une instruction READ (lecture) et ensuite placer l'information chaîne dans la ligne DATA. Le programme ci-dessous est acceptable:

```
NEW

10 FORX = 1 TO 3
15 READ A$
20 PRINT "A$ EST MAINTENANT : ";A$
30 NEXT
40 DATA CELA,EST,AMUSANT

RUN

A$ IS NOW : THIS
A$ IS NOW : IS
A$ IS NOW : FUN
READY
```

Notez que cette fois l'instruction READ a été placée à l'intérieur d'une boucle FOR . . . NEXT. Cette boucle a ensuite été exécutée pour adapter le nombre des valeurs dans l'instruction DATA.

Dans de nombreux cas, vous devez modifier le nombre des valeurs dans l'instruction DATA chaque fois que le programme est passé. Une façon d'éviter de compter le nombre des valeurs et d'éviter également un message d'erreur OUT OF DATA ERROR consiste à placer un «drapeau» (FLAG) comme dernière valeur dans la ligne DATA. Ceci peut être une valeur que vos données ne prendraient jamais, comme par exemple un nombre négatif ou un nombre très petit ou très grand. Lorsque cette valeur est lue, le programme se branche sur la partie suivante.

Il existe une façon de réutiliser la même donnée plus tard dans le programme en réinitialisant (RESTORE) le pointeur de données au début de la liste des données. Ajoutez la ligne 50 au programme précédent:

50 GOTO 10

Vous continuerez à obtenir le message d'erreur OUT OF DATA parce que lorsque le programme se rebranche sur la ligne 10 pour relire les

données, le pointeur de données indique que toutes les données ont été utilisée. A présent, ajoutez:

45 RESTORE

et faites à nouveau passer le programme. Le pointeur de données a été RESTOREd (réinitialisé) et les données peuvent être lues sans interruption.

MOYENNES

Le programme ci-dessous illustre une utilisation pratique de READ et DATA par la lecture d'un jeu de nombres et le calcul de leur moyenne.

```
NEW

5 T=0:CT=0
10 READ X
20 IF X=-1 THEN 50:REM TEST
25 CT=CT+1
30 T=T+X:REM CALCUL DE LA SOMME
40 GOTO 10
50 PRINT CT ;"VALEURS ONT ETE LUES"
60 PRINT "SOMME =";T
70 PRINT "MOYENNE =";T/CT
80 DATA 75,80,62,91,87,93,78,-1

RUN
THERE WERE 7 VALUES READ
TOTAL = 566
AVERAGE = 80.8571429
```

La ligne 5 met CT, le CompTeur, et T le total égal à zéro. La ligne 10 lit une valeur et assigne la valeur à X. La ligne 20 contrôle pour vérifier si la valeur est notre drapeau (ici A – 1). Si la valeur lue fait partie des données valides, CT est avancé de 1 et X est ajouté au total.

Lorsque le drapeau est lu, le programme se branche sur la ligne 50 qui affiche le nombre des valeurs lues. La ligne 60 affiche le total et la ligne 70 divise le total par le nombre des valeurs pour obtenir la moyenne. En utilisant un drapeau à la fin de DATA, vous pouvez placer n'importe quel nombre de valeurs dans les instructions DATA – qui peuvent s'étendre sur plusieurs lignes – sans vous préoccuper du comptage du nombre des valeurs entrées.

Une autre variation de l'instruction READ fait appel à l'affectation d'in-

formations provenant de la même ligne DATA à des variables différentes. Ces informations peuvent même être un mélange d'une chaîne de données et de valeurs numériques. Vous pouvez réaliser toutes ces opérations dans le programme suivant qui lira un nom, certains scores – disons le bowling – et affichera le nom, les scores et le score moyen:

```
NEW

10 READ N$,A,B,C
20 PRINT "LES POINTS DE "N$;" SONT :";A;" ";B;" ";C
30 PRINT "ET LA MOYENNE EST :";(A+B+C)/3
40 PRINT:GOTO 10
50 DATA MIKE, 190, 185, 165, DICK, 225, 245, 190
60 DATA JOHN, 155, 185, 205, PAUL, 160, 179, 187

RUN

MIKE'S SCORES WERE: 190  185  165
AND THE AVERAGE IS : 180

DICK'S SCORES WERE: 225  245  190
AND THE AVERAGE IS : 220
```

Lors du passage du programme, les instructions DATA ont été établies dans le même ordre que l'instruction READ, mises à part les informations: un nom (une chaîne), ensuite trois valeurs. En d'autres termes, N$ passé pour la première fois obtient les données de MIKE (DATA MIKE), A dans READ correspond à 190 dans l'instruction DATA, B à 185 et C à 165. Le processus est alors répété dans l'ordre pour le reste des informations. (Dick et ses scores, John et ses scores, et Paul et ses scores).

VARIABLES A INDICES

Auparavant, nous avons uniquement utilisé de simples variables BASIC, telles que A, A$, NU pour représenter des valeurs. Il s'agissait d'une seule lettre suivie d'une lettre ou d'un seul chiffre.

Il est douteux que dans tous les programmes que vous pourriez rédiger, nous ayons besoin de plus de noms de variables que possible avec toutes les combinaisons de lettres et de chiffres disponibles. Mais vous êtes limités par le fait que les variables sont utilisées ensemble avec les programmes.

A présent, introduisons le concept de variables à indices.

A(1) ← INDICE
↑
VARIABLE

C'est lu: A indice 1. Une variable à indice est constituée d'une lettre suivie d'un indice entre parenthèses. Veuillez noter la différence entre A, Al et A(1). Chacun est différent. Seul A(1) est une variable à indice.

Les variables à indices, tout comme les variables simples, désignent un emplacement de la mémoire à l'intérieur de l'ordinateur. Imaginez les variables à indices comme des cases pour mémoriser des informations, tout comme les variables simples:

A(0)	
A(1)	
A(2)	
A(3)	
A(4)	

Si vous avez écrit:

A(Ø) = 25:A(3) = 55:A(4) = −45.3

alors le mémoire aura l'aspect suivant:

A(0)	25
A(1)	
A(2)	
A(3)	55
A(4)	−45.3

Ce groupe de variables à indices est également appelé un tableau. Dans ce cas, un tableau à une entrée. PAR la suite, nous introduirons les tableaux à plusieurs entrées.

Les indices peuvent également être plus complexes pour compendre d'autres variables ou calculs. Les notations suivantes sont des variables à indices valides:

A(X) A(X+1) A(2+1) A(5*3)

Les expressions à l'intérieur des parenthèses sont évaluées suivant les mêmes règles que les opérations arithmétiques décrites chapitre 2.

A présent que les règles de base ont été établies, comment les variables à indices peuvent-elles être utilisées? Un moyen consiste à mémoriser une liste des nombres entrés avec les instructions INPUT ou READ. Utilisons les variables à indices pour obtenir les moyennes d'une autre façon.

```
5 PRINTCHR$(147)
10 INPUT "COMBIEN DE CHIFFRES :";X
20 FOR A=1 TO X
30 PRINT "VALEUR#";A;:INPUT B(A)
40 NEXT
50 SU=0
60 FOR A=1 TO X
70 SU=SU+B(A)
80 NEXT
90 PRINT:PRINT "VALEUR MOYENNE = "; SU/X

RUN

HOW MANY NUMBERS :? 5
ENTER VALUE # 1   ? 125
ENTER VALUE # 2   ? 167
ENTER VALUE # 3   ? 189
ENTER VALUE # 4   ? 167
ENTER VALUE # 5   ? 158

AVERAGE =  161.2
```

Il y avait une voie plus simple pour réaliser ce que nous avons fait dans ce programme, mais ainsi la façon dont fonctionnent les variables à indices est mieux démontrée.

La ligne 10 demande combien de nombres seront entrés. Cette variable X agit comme compteur de la boucle à l'intérieur de laquelle des valeurs sont entrées et assignées à la variable à indice B.

A chaque passage de la boucle d'entrée, A est augmenté de 1 et la valeur suivante qui est entrée est assignée au prochain élément, tableau A. Par exemple lors du premier passage la boucle A = 1, si bien que la première valeur entrée est assignée à B(1). Au passage suivant A = 2, la valeur suivante est assignée à B(2), ainsi de suite jusqu'à ce que toutes les valeurs aient été entrées.

Mais à présent, une grande différence entre en jeu. Une fois que toutes les valeurs ont été entrées, elles sont mémorisées dans le tableau, prêtes à être employées dans un grand nombre de variétés. Avant, vous obtenez le total de chaque passage de la boucle INPUT ou READ, mais sans pouvoir revenir aux éléments individuels des données sans avoir relu les informations.

Une autre boucle a été conçue avec les lignes 50 à 80 pour ajouter les différents éléments du tableau et afficher la moyenne. Cette partie séparée du programme indique que toutes les valeurs sont mémorisées et peuvent être obtenues selon les besoins.

Pour démontrer que la totalité des différentes valeurs sont effectivement séparément mémorisées dans un tableau, frappez ce qui suit immédiatement après avoir passé le programme précédent:

FOR A = 1 TO 5 : ? B(A),: NEXT

L'écran indique vos valeurs effectives et les contenus du tableau sont affichés.

DIMENSION

Si vous essayez d'entrer plus de 10 nombres dans l'exemple précédent, vous obtenez une erreur de dimension. Les tableaux jusqu'à 11 éléments (indices 0 à 10 pour un tableau à une entrée) peuvent être utilisés si c'est nécessaire tout comme les variables simples peuvent être utilisées où que ce soit à l'intérieur d'un programme. Les tableaux de plus de 11 éléments ont besoin d'être «déclarés» dans une instruction de dimension.

Ajoutez cette ligne au programme:

5 DIM B(100)

Elle fait savoir à l'ordinateur que vous aurez au maximum 100 éléments dans le tableau.

L'instruction de dimension peut également être utilisée avec une variable si bien que la ligne suivante paut remplacer la ligne 5 (ne pas oublier d'éliminer la ligne 5):

15 DIM B(X)

Ceci dimensionnera le tableau avec le nombre exact de valeurs qui sera entré.

Soyez prudent. Une fois dimensionné, un tableau ne peut être redimensionné dans une autre partie du programme. Cependant, vous pouvez avoir de multiples tableaux à l'intérieur d'un programme et alors les dimensionner tous de la même façon comme ceci:

10 DIM C(29), D(50), E(40)

JETS DE DES SIMULES

Au fur et à mesure que les programmes deviennent plus complexes, l'utilisation de variables à indices réduit le nombre des instructions nécessaires et rend le programme plus simple à rédiger. Une seule

variable à indices peut être utilisée par exemple pour conserver le nombre de fois qu'une face particulière apparaît:

```
1 REM SIMULATION DE
2 PRINT CHR$(147)
10 INPUT "COMBIEN DE JETS ";X
20 FOR L=1 TO X
30 R=INT(6*RND(1))+1
40 F(R)=F(R)+1
50 NEXT L
60 PRINT"JET","NOMBRE DES JETS"
70 FOR C=1 TO 6:PRINTC,F(C):NEXT
```

Le tableau F, pour FACE, sera utilisé pour conserver le nombre de fois qu'une face particulière apparaît. Par exemple, chaque fois que 2 est sorti, F(2) est augmenté de 1. En utilisant le même élément de la ligne pour conserver le nombre effectif sur la face qui apparaît, nous avons évité d'avoir besoin de 5 autres variables (une pour chaque face) et éliminé les instructions numériques pour contrôler et voir quel chiffre est sorti. La ligne 10 demande combien de jets vous souhaitez simuler.

La ligne 20 établit la boucle pour réaliser le jet aléatoire et augmenter l'elément adéquat du tableau position lors de chaque jet.

Après que tous les jets exigés ont été réalisés, la ligne 60 affiche l'entête et la ligne 70 affiche le nombre de fois que chaque face sort.

Un exemple de passage peut avoir cet aspect:

```
HOW MANY ROLLS: ? 1000
FACE                NUMBER OF ROLLS
1                   148
2                   176
3                   178
4                   166
5                   163
6                   169
```

Parfait, au moins il n'était pas chargé!

Juste une comparaison, vous trouverez ci-dessous une façon de rédiger de nouveau le même programme, mais sans utiliser les variables à indices. Ne vous fatiguez pas pour les frapper, mais notez les instructions supplémentaires nécessaires.

```
10 INPUT "COMBIEN DE JETS ";X
20 FOR I=1 TO X
30 R=INT(6*RND(1))+1
40 IFR=1THENF1=F1+1:NEXT
41 IFR=2THENF2=F2+1:NEXT
42 IFR=3THENF3=F3+1:NEXT
43 IFR=4THENF4=F4+1:NEXT
```

```
44 IFR=5THENF5=F5+1:NEXT
45 IFR=6THENF6=F6+1:NEXT
60 PRINT"JETS","NOMBRE DES JETS"
70 PRINT 1,F1
71 PRINT 2,F2
72 PRINT 3,F3
73 PRINT 4,F4
74 PRINT 5,F5
75 PRINT 6,F6
```

Le programme a été doublé en dimension de 8 à 16 lignes. Dans les plus vastes programmes, les économies d'espace résultant de l'utilisation de variables à indices, seront encore plus importantes.

TABLEAU A DEUX ENTREES

Vous avez essayé les tableaux à une entrée au début de ce chapitre. Ce type de tableau a été visualisé sous la forme d'un groupe de cases consécutives dans la mémoire, chacune contenant un élément du tableau. A quoi, pensez-vous, ressemble un tableau à deux entrées?

En premier lieu, un tableau à deux entrées doit être écrit comme suit:

A(4,6)
NOM DU TABLEAU ↑ INDICES

et peut être imaginé sous la forme d'une grille à deux dimensions à l'intérieur de la mémoire:

	Ø	1	2	3	4	5	6
Ø							
1							
2							
3							
4							

Les indices pourraient représenter la ligne et la colonne à l'intérieur du tableau où l'élément particulier de la ligne est mémorisé.

A(3,4) = 255
LIGNE ↑ COLONNE

	Ø	1	2	3	4	5	6
Ø							
1							
2							
3					255		
4							

Si nous assignons la valeur 255 à A(3,4), alors 255 doit être considéré comme étant placé dans la quatrième colonne de la troisième du tableau. Les tableaux à deux entrées suivent les mêmes règles que celles établies pour les tableaux à une entrée:

Ils doivent être dimensionnés:	DIM A(20,20)
Affectation des données:	A(1,1) = 255
Affectation des valeurs à d'autres variables:	AB = A(1,1)
Affichage des valeurs:	PRINT A(1,1)

Si les tableaux à deux entrées fonctionnent comme leur contre-partie plus petite, quelles sont les possibilités supplémentaires qu'ils offrent?

Essayez ceci: Imaginez une manière d'utiliser un tableau à deux entrées pour tabuler les résultats d'un questionnaire pour votre club qui comprend 4 questions, et prévoit 3 réponses pour chaque question. Le problème peut se présenter comme suit:

QUESTIONNAIRE DE CLUB

Q1: FAVORISEZ-VOUS LA RESOLUTION 1?

☐ 1 = OUI ☐ 2 = NON ☐ 3 = ABSTENTION . . . etc.

Le tableau de ce problème peut être représente comme ceci:

	REPONSES OUI	NON	ABSTENTION
QUESTION 1			
QUESTION 2			
QUESTION 3			
QUESTION 4			

Le programme, pour réaliser le calcul effectif du questionnaire, peut ressembler à celui représenté page 103. Ce programme fait appel à beaucoup de techniques de programmation présentées jusqu'à présent. Même si vous n'avez pas besoin du programme à l'instant même, vérifiez pour savoir si vous pouvez suivre le fonctionnement du programme.

Le coeur de ce programme est un tableau à deux entrées de 4 x 3, A(4,3). Les totaux des réponses de chaque réponse possible à chaque question sont conservés dans l'élément approprié du tableau. Pour plus de simplicité, nous n'avons pas utilisé les premières lignes et colonnes (A(0,0) à A(0,4). Rappelez-vous que ces éléments sont toujours présents dans n'importe quel tableau que vous concevez.

Dans la pratique, si l'on répond par oui à la question 1, alors A(1,1) est avancé de 1 – ligne 1 pour la question 1 et colonne 1 pour la réponse oui.

Le reste des questions et des réponses suit le même schéma. Une réponse négative à la question 3 ajouterait 1 à l'élément A (3,2) et ainsi de suite.

```
 20 PRINT"■":REM TOUCHE SHIFT+CLR
 30 FOR R=1 TO 4
 40 PRINT"QUESTION NO :";R
 50 PRINT "1-OUI 2-NON 3-NON DECIDE"
 60 PRINT "QUELLE EST LA REPONSE:";
 61 GET C:IF C<1 OR C>3 THEN 61
 65 PRINT C:PRINT
 70 A(R,C)=A(R,C)+1
 80 NEXT R
 85 PRINT
 90 PRINT "VOULEZ-VOUS ENTRER UNE NOUVELLE REPONSE?
    ":PRINT"REPONSE (O/N)"
100 GET A$:IF A$=""THEN 100
110 IF A$="O"THEN 20
120 IF A$<>"N"THEN 100
130 PRINT"■LES REPONSES EN TOTAL SONT:":PRINT
140 PRINT SPC(18);"REPONSE"
141 PRINT"QUESTION","OUI","NON   NON DECIDE"
142 PRINT"--------  --------------------------"
150 FOR R=1 TO 4
160 PRINT R,A(R,1),A(R,2),A(R,3)
170 NEXT R
RUN

QUESTION # : 1
1-YES  2-NO  3-UNDECIDED
WHAT WAS THE RESPONSE : 1

QUESTION # : 2
1-YES  2-NO  3-UNDECIDED
WHAT WAS THE RESPONSE : 1

And so on...

THE TOTAL RESPONSES WERE:
                          RESPONSE
QUESTION          YES          NO          UNDECIDED

   1              6            1           0
   2              5            2           0
   3              7            0           0
   4              2            4           1
```

ANNEXES

INTRODUCTION

A présent que vous vous êtes familiarisé avec votre COMMODORE 64, nous souhaiterions que vous sachiez que notre service après-vente ne s'arrête pas là. Vous pouvez ne pas le savoir, mais Commodore existe depuis plus de 23 ans. Nous avons introduit, en 1970, le premier ordinateur personnel autonome (le PET). Depuis cette date, nous sommes devenus le principal fabricant d'ordinateurs dans de nombreux pays du monde. Nos possibilités de conception et de fabrication de nos propres puces d'ordinateur nous permettent de vous fournir de nouveaux et de meilleurs ordinateurs personnels à des prix inattendus vu le niveau de perfectionnement technique. Commodore s'éngage à soutenir non seulement vous-même, vous qui êtes l'utilisateur final, mais également le revendeur auprès duquel vous avez acheté votre ordinateur ainsi que les magazines qui publient des articles pratiques vous présentant de nouvelles applications techniques, et aussi les maisons software qui produisent des programmes enfichables dans les disques ou bandes destinés à votre ordinateur. Nous vous encourageons à fonder ou à adhérer à un club d'utilisateurs Commodore où vous pourrez apprendre de nouvelles techniques, échanger des idées et partager vos découvertes. Nous publions deux magazines séparés qui contiennent des conseils de programmation, des informations sur les nouveaux produits et des idées pour les applications informatiques (voir annexe N).

En Amérique du Nord, Commodore met à disposition un réseau d'informations Commodore par le service d'informations CompuServe . . . pour accéder à ce réseau, il vous suffit de disposer de votre ordinateur COMMODORE 64 et du chargeur d'interface téléphonique avantageux VICMODEM (ou d'un autre modem compatible).

Les annexes suivantes contiennent des diagrammes, des tableaux et d'autres informations qui vous aident à programmer votre COMMODORE 64 plus rapidement et plus efficacement. Elles comprennent également des informations importantes sur la grande variété des produits Commodore qui peuvent vous intéresser ainsi qu'une bibliographie, une liste de plus de 20 livres et magazines qui vous aident à développer vos facultés de programmation et à vous tenir au courant des dernières informations relatives à votre ordinateur et ses périphériques.

ANNEXE A

ACCESSOIRES DU COMMODORE 64 ET SOFTWARE

ACCESSOIRES

Le COMMODORE 64 peut recevoir les accessoires et périphériques de stockage du Commodore VIC 20 soit lecteur-enregistreur DATASSETTE, entraîneur de disque, modem, imprimante, si bien que votre système peut être étendu en fonction de vos besoins croissants.

- Lecteur-enregistreur Datassette – Cette unité avantageuse à bande permet de mémoriser les programmes et données sur une bande de cassette et de les restituer à une date ultérieure. Le Datassette peut également être utilisé pour restituer des programmes préenregistrés.
- Disque – La station de disquettes mono VC 1541 utilise des disquettes standards de 5 1/4'' pour mémoriser des programmes et des données. La station permet un accès plus rapide aux données et conserve jusqu'à 170.000 caractères d'informations sur chaque disquette. Les stations de disquettes sont «intelligentes», ce qui signifie qu'elles disposent de leur propre microprocesseur et mémoire. Elles n'occupent pas de place dans la mémoire de l'ordinateur.
- Imprimante – L'imprimante VC 1525 fournit des exemplaires sur papier des programmes, données ou graphiques. Cette imprimante à matrice de 30 caractères par seconde utilise du papier en continu normal et d'autres fournitures peu coûteuses.
- Modules d'interface – Un certain nombre de modules d'interface spécialisés sont disponibles pour le COMMODORE 64 de manière à permettre à différents périphériques standards, tels que modems, imprimantes, contrôleurs et instruments de mesure, d'être branchés au système.

Avec un module spécial IEEE-488, le COMMODORE 64 supporte la gamme complète des périphériques CBM, y compris les stations de disquettes et imprimantes. D'autre part, un carte Z80 vous permet d'utiliser le CP/M* sur le COMMODORE 64, vous procurant un accès à la plus grande base d'applications de micros-ordinateurs disponibles.

SOFTWARE

Plusieurs catégories de software sont offertes pour le COMMODORE 64, vous procurant une grande variété d'applications personnelles, de distraction et d'éducation au choix.

AIDES COMMERCIALES

- Des programmes de feuilles d'enregistrement électronique vous permet de planifier des budgets et de réaliser des analyses «que faire si?». D'autre part, avec le programme graphique en option, vous pouvez créer des graphiques éloquents à partir des données des feuilles.
- La planification financière, telle que l'amortissement de prêts, sera aisément traitée avec les programmes planification financière.
- Un certain nombre de programmes de gestion de temps professionnels facilitent la gestion de rendez-vous et du travail.
- Des programmes de base de données faciles à utiliser vous permettent de retenir des informations . . . d'établir des listes postales . . . des listes de numéros de téléphone . . . des inventaires . . . et d'organiser les informations sous une forme utile.
- Les programmes de traitement de termes techniques font du COMMODORE 64 un processeur de mots performant. La frappe et la révision de mémos, de lettres et d'autres textes deviennent un plaisir.

DISTRACTION

- Des jeux de la plus haute qualité sont disponibles sur les programmes enfichables sur le COMMODORE 64 et procurent des heures de distraction. Ces programmes font appel aux graphiques à haute résolution et à la totalité de la gamme de sons réalisables avec le COMMODORE 64.

* CP/M est une marque déposée de Digital Research Inc.

- Votre COMMODORE 64 vous fournit toute la joie et la distraction possible grâce aux jeux MAX étant donné que ces deux machines possèdent des chargeurs de programmes entièrement compatibles.

EDUCATION

- Le COMMODORE 64 est un précepteur qui ne se fatigue jamais et qui est toujours attentif. A part d'accéder à la plupart des vastes programmes d'éducation PET, le COMMODORE 64 met également à disposition des langages d'éducation supplémentaires, tels que PILOT, LOGO et d'autres programmes avancés importants.

ANNEXE B

FONCTIONNEMENT AVANCE A CASSETTE

En dehors de sauvegarder des copies de vos programmes sur bandes, le COMMODORE 64 peut également mémoriser des valeurs de variables ou d'autres positions de données dans un groupe intitulé un fichier (FILE). Ceci vous permet même de mémoriser plus d'informations que ne peut en contenir en même temps la mémoire centrale de l'ordinateur.

Les instructions utilisées avec les fichiers de données sont OPEN, CLOSE, PRINT#, INPUT# et GET#. Le ST (état) variable du système est utilisé pour contrôler les marqueurs de bande.

On utilise, pour l'enregistrement des données sur bandes, les mêmes concepts que ceux pour l'affichage des informations sur l'écran de l'ordinateur. Mais, à la place d'afficher les informations sur l'écran, les informations sont imprimées sur la bande en utilisant une modification de l'instruction PRINT-PRINT#.

Le programme suivant illustre comment:

```
10 PRINT"ECRIRE SUR BANDE"
20 OPEN1,1,1,"OUVRIR UN FICHIER"
30 PRINT"ENTREZ VOS DONNEES OU FRAPPEZ LE MOT STOP"
50 PRINT
60 INPUT"DATA";A$
70 PRINT#1,A$
80 IFA$<>"STOP"THEN50
90 PRINT
100 PRINT"FERMETURE DU FICHIER"
110 CLOSE1
```

La première chose que vous avez à faire est d'ouvrir un fichier (dans ce cas DATA FILE). La ligne 10 s'en occupe. Le programme suggère les données que vous souhaitez sauvegarder sur la bande ligne 60. La ligne 70 enregistre ce que vous avez frappé – contenu dans A$ – sur la bande. Et le processus continue.

Si vous frappez STOP, la ligne 110 ferme (CLOSE) le fichier. Pour restituer les informations, rebobiner la bande et essayer ceci:

```
10 PRINT "READ-TAPE-PROGRAM"
20 OPEN 1,1,0,"DATA FILE"
30 PRINT "FILE OPEN"
40 PRINT
50 INPUT#1, A$
60 PRINT A$
70 IF A$ = "STOP" THEN END
80 GOTO 40
```

A nouveau, le fichier «DATA FILE» doit d'abord être ouvert (OPEN). A la ligne 50, le programme entre (INPUT) A$ à partir de la bande et affiche (PRINT) également A$ sur l'écran. Ensuite, tout le processus est répété jusqu'à ce que «STOP» soit trouvé, ce qui termine (END) le programme.
Une modification de GET-GET# peut être également utilisée pour relire les données à partir de la bande. Remplacer les lignes 50-80 du programme ci-dessus avec:

```
50 GET#1, A$
60 IF A$ = "" THEN END
70 PRINT A$, ASC(A$)
80 GOTO 50
```

ANNEXE C

BASIC COMMODORE 64

Ce manuel a été pour vous une introduction dans le langage BASIC – suffisante pour avoir une idée de la programmation de l'ordinateur et d'une partie du vocabulaire utilisé. Cette annexe vous fournit une liste complète des règles (SYNTAXE) du BASIC COMMODORE 64 avec des descriptions concises. Veuillez essayer ces instructions. Rappelez-vous que vous ne pouvez endommager l'ordinateur en frappant simplement des programmes et que la meilleure façon d'apprendre à l'utiliser est de l'employer.

Cette annexe est divisée en sections selon les différents types d'opérations en BASIC. Ceux-ci comprennent:

1. **Les variables et opérateurs:** Description des différents types de variables, noms de variables légaux et opérateurs arithmétiques et logiques.
2. **Commandes:** Description des commandes utilisées pour travailler avec les programmes, les éditer, les mémoriser, les effacer.
3. **Instructions:** Description des instructions de programmation BASIC utilisées dans les lignes numérotées des programmes.
4. **Fonctions:** Description des fonctions de chaîne, numériques et d'affichage.

VARIABLES

Le COMMODORE 64 utilise trois types de variables en BASIC. Il s'agit des variables numériques normales, numériques entières et de variables-chaînes (alphanumériques). Les noms des variables peuvent comprendre une seule lettre, une lettre suivie d'un nombre ou deux lettres.

Une variable entière est spécifiée par l'utilisation du signe pourcent (%) après le nom de la variable. Les variables de chaînes possèdent le symbole $ après leur nom.

EXEMPLES

Noms de variables normales: A, A5, BZ
Noms de variables entières: A%, A5%, BZ%
Noms de variables de chaînes: A$, A5$, BZ$

Les tableaux sont des listes de variables portant le même nom, utilisant des nombres supplémentaires pour spécifier l'élément du tableau. Des tableaux sont définis en utilisant l'instruction DIM et peuvent contenir une virgule flottante, un nombre entier ou des variables-chaînes. Le nom de la variable du tableau est suivi par un jeu de parenthèses () entourant le nombre de variables de la liste.

A(7),BZ%(11),A$(50),PT(20,20)

REMARQUE: Il existe trois noms de variables qui sont réservés au Commodore 64 et qui ne peuvent être définis par vous. Ces variables sont: ST, TI et TI$. ST est une variable d'état qui est en rapport avec les opérations d'entrée/sortie. La valeur de ST se modifie s'il existe un problème de chargement d'un programme à partir d'un disque ou d'une bande.

TI et TI$ sont des variables qui sont en rapport avec l'horloge en temps réel incorporée dans le Commodore 64. La variable TI est remise à jour tous les 60èmes de seconde. Elle part à zéro lorsque l'ordinateur est branché et elle n'est réinitialisée que par la modification de la valeur de TI$.

TI$ est une chaîne qui est constamment mise à jour par le système. Les deux premiers caractères contiennent le nombre des heures, les 3ème et 4ème caractères le nombre des minutes et les 5ème et 6ème caractères sont le nombre des secondes. Cette variable peut recevoir n'importe quelle valeur numérique et sera remise à jour à partir de ce point.

TI$ = «101530» règle l'horloge sur 10:15 et 30 secondes le matin.

Cette horloge est effacée lorsque l'ordinateur est arrêté et repart à zéro lorsque le système est remis sous tension.

OPERATEURS

Les opérateurs arithmétiques comprennent les signes suivants:

\+ Addition
– Soustraction
* Multiplication
/ Division
↑ Elévation à une puissance

Sur une ligne contenant plusieurs opérateurs, il existe un ordre déterminé selon lequel les opérations se déroulent toujours. Si plusieurs opérations sont utilisées conjointement sur la même ligne, l'ordinateur

fixe les priorités comme suit: En premier lieu l'élévation à la puissance. Ensuite multiplication et division et en dernier lieu addition et soustraction.

Vous pouvez modifier l'ordre des opérations en plaçant entre parenthèses le calcul à réaliser en premier lieu. Les opérations entre parenthèses ont lieu avant les autres opérations.

Il existe également des opérations d'égalité et d'inégalité:

=	Egal à
<	Inférieur à
>	Supérieur à
< =	Inférieur à ou égal à
> =	Supérieur à ou égal à
<>	Différent de

Finalement, il existe trois opérateurs logiques:

AND	ET (Deux conditions s'appliquent en même temps)
OR	OU (Au moins une des conditions est exact)
NOT	NE . . . PAS (La condition n'est pas exacte)

Ceux-ci sont très souvent utilisés pour associer les formules multiples dans les instructions IF . . . THEN. Par exemple:

IFA = B AND C = D THEN (nécessite deux parties pour être vérifié)
IFA = B OR C = THEN 100 (autorise une partie ou l'autre pour être vérifié)

COMMANDES

CONT (continuer)

Cette commande est utilisée pour redémarrer l'exécution d'un programme qui a été arrêté soit en utilisant la touche STOP, soit une instruction STOP, soit l'instruction SEND à l'intérieur du programme. Le programme redémarre exactement au même endroit où il a été interrompu.

CONT ne fonctionne pas si vous avez modifié ou ajouté des lignes au programme (ou même si vous avez juste déplacé le curseur) ou bien si le programme est arrêté à la suite d'une erreur ou si vous avez provoqué une erreur avant d'essayer de redémarrer le programme. Dans ces cas, vous obtenez un message d'erreur CAN'T CONTINUE ERROR.

LIST

La commande LIST vous permet de jeter un coup d'oeil sur les lignes d'un programme BASIC qui sont dans la mémoire. Vous pouvez demander l'affichage de la totalité du programme ou seulement de certains numéros de lignes:

LIST	Affiche la totalité du programme
LIST 10–	Affiche le programme uniquement de la ligne 10 jusqu'à la fin
LIST 10	N'affiche que la ligne 10
LIST –10	Affiche les lignes du début jusqu'à 10
LIST 10–20	Affiche les lignes 10 à 20 comprises

LOAD

Cette commande est utilisée pour transférer un programme de la bande ou du disque dans la mémoire de manière que le programme puisse être utilisé. Si vous avez seulement frappé LOAD et pressé la touche RETURN, le premier programme trouvé sur la cassette sera mis en mémoire. La commande peut être suivie par un nom de programme entre guillemets. Le nom peut alors être suivi par une virgule et un nombre ou une variable numérique qui agit comme un numéro de périphérique pour indiquer d'où vient le programme.

Si aucun numéro de périphérique n'est donné, le COMMODORE 64 suppose qu'il s'agisse du périphérique numéro 1 qui est l'unité de cassette. L'autre périphérique communément utilisé avec la commande LOAD est l'entraîneur de disque qui est le périphérique no 8.

LOAD	Met en mémoire le programme suivant de la bande
LOAD «HELLO»	Explore la bande pour chercher le programme appelé HELLO et charge le programme s'il est trouvé.
LOAD A$	Cherche le programme dont le nom est dans la variable A$
LOAD «HELLO», 8	Cherche le programme intitulé HELLO sur l'entraîneur de disque
LOAD «*», 8	Cherche le premier programme du disque

NEW

Cette commande efface la totalité du programme dans la mémoire et efface également toutes les variables qui peuvent avoir été utilisées. A moins que le programme n'ait été sauvegardé (SAVE), il est perdu. Soyez très prudent quand vous utilisez cette commande.

La commande NEW peut également être utilisée comme instruction du programme BASIC. Lorsque le programme atteint cette ligne, le programme est effacé. C'est utile quand vous souhaitez tout effacer lorsque le programme est réalisé.

RUN

Cette commande provoque l'exécution d'un programme une fois que le programme est chargé dans la mémoire. S'il n'y a pas de numéro de ligne après RUN, l'ordinateur démarre sur le numéro de ligne le plus bas. Si un numéro de ligne est désigné, le programme commence l'exécution à partir de la ligne spécifiée.

RUN	Démarre le programme sur le numéro de ligne le plus bas
RUN 100	Démarre l'exécution sur la ligne 100
RUN X	UNDEFINED STATEMENT ERROR (message d'erreur instruction indéfinie). Vous devez toujours spécifier un numéro de ligne déterminé, et non une représentation de variable.

SAVE

Cette commande mémorise l'actuel programme sur cassette ou disque. Si vous tapez seulement SAVE et RETURN, le programme est sauvegardé (SAVE) sur la cassette. L'ordinateur n'a aucun moyen de savoir si un programme existe déjà sur cette bande considérée, si bien que vous devez faire très attention à vos bandes sinon vous pouvez effacer un programme valable.

Si vous tapez SAVE suivi d'un nom entre guillements ou d'une variable-chaîne, l'ordinateur fournira le programme de ce nom de manière qu'il puisse être plus aisément localisé et repris à l'avenir. Le nom peut également être suivi d'un numéro de périphérique.

Une virgule et un second nombre, soit 0 soit 1, peuvent être placés après le numéro de périphérique. Si le second nombre est 1, le COMMODORE 64 place un marqueur END-OF-BANDE après votre programme. Ceci signale à l'ordinateur de ne pas chercher plus en avant sur la bande dans le cas où vous donneriez une commande supplémentaire LOAD. Si vous essayez de charger un programme et que l'ordinateur trouve l'un de ces marqueurs, vous obtiendrez le message d'erreur FILE NOT FOUND ERROR (fichier non trouvé).

SAVE	Mémorise le programme sur bande sans nom
SAVE «HELLO»	Mémorise le programme sur bande sous le nom HELLO

SAVE A$	Mémorise le programme sur bande sous le nom A$
SAVE «HELLO», 8	Mémorise le programme sur disque sous le nom HELLO
SAVE «HELLO», 1,1	Mémorise le programme sur bande sous le nom HELLO et fait suivre le programme du marqueur END-OF-TAPE.

VERIFY

Cette commande incite l'ordinateur à contrôler le programme sur le disque ou la bande par rapport à celui dans la mémoire. Ceci est une preuve que le programme est effectivement sauvegardé (SAVE) dans le cas où la bande ou le disque serait détérioré ou quelque chose se serait mal passé pendant l'opération SAVE. VERIFY sans autres indications après la commande incite le COMMODORE 64 à contrôler le programme suivant sur la bande, quel que soit son nom, par rapport au programme présent dans la mémoire.

VERIFY suivit d'un nom de programme ou d'une variable-chaîne recherche le programme concerné et ensuite le contrôle. Les numéros des périphériques peuvent également être incorporés dans la commande de vérification (VERIFY).

VERIFY	Contrôle le programme suivant sur la bande
VERIFY «HELLO»	Cherche le programme HELLO et le contrôle par rapport à la mémoire
VERIFY «HELLO», 8	Cherche le programme HELLO sur le disque et ensuite le contrôle.

INSTRUCTIONS

CLOSE

Cette instruction achève et ferme tous les fichiers utilisés par des instructions OPEN. Le numéro suivant CLOSE est le numéro du fichier à fermer.

CLOSE 2	Seul le fichier no 2 est fermé.

CLR

Cette instruction efface toutes les variables de la mémoire mais laisse le programme proprement dit intact. L'instruction est automatiquement exécutée lorsqu'un ordre RUN est donné.

CMD

CMD envoie ce qui est sorti, ce que normalement devrait aller sur l'écran (par exemple les instructions PRINT, LIST, mais non POKE sur l'écran), au lieu d'aller sur un autre périphérique. Ceci peut être une imprimante ou un fichier de données sur bande ou disque. Ce périphérique ou fichier doit être d'abord ouvert (OPEN). L'instruction CMD doit être suivie d'un numéro ou d'une variable numérique se référant au fichier.

OPEN 1,4	Ouvre le périphérique no 4 qui est l'imprimante
CMD 1	La totalité de la sortie normale passe maintenant à l'imprimante
LIST	La liste du programme passe maintenant à l'imprimante et non sur l'écran.

Pour ramener la sortie sur l'écran, fermer (CLOSE) le fichier avec CLOSE 1.

DATA

Cette instruction est suivie d'une liste d'éléments à utiliser par les instructions READ. Les éléments peuvent être des valeurs numériques ou des chaînes de textes et les éléments sont séparés par des virgules. Les éléments de chaînes n'ont pas besoin d'être à l'intérieur de guillemets à moins qu'ils contiennent des espaces, des doubles points ou des virgules. Si deux virgules n'ont rien entre elles, la valeur sera lue comme étant un zéro pour un nombre, ou une chaîne vide.

10 DATA 12, 14.5, «HALLO, PARTNER», 3.14 TEIL 1

DEF FN

Cette instruction vous permet de définir un calcul complexe comme étant une fonction d'un nom court. Dans le cas d'une longue formule qui est utilisée de nombreuses fois à l'intérieur du programme, ceci peut permettre d'économiser du temps et de la place.

Le nom de la fonction sera FN et n'importe quel nom de variable légal (1 ou 2 caractères de long). En premier lieu, vous devez définir la fonction en utilisant l'instruction DEF suivie du nom de la fonction. Un jeu de parenthèses renfermant une valeur numérique suit le nom. Ensuite, suit la formule effective que vous souhaitez définir avec la variable au bon endroit. Vous pouvez alors «appeler» la formule en substituant un nombre quelconque à la variable.

```
1Ø DEF FNA(X) = 12*(34.75 – X/.3)
2Ø PRINT FNA(7)
```

7 est inséré là où X se trouve dans la formule

Pour cet exemple, le résultat serait 137.

DIM (dimension d'un tableau)

Lorsque vous utilisez plus de 11 éléments d'un tableau, vous devez exécuter l'instruction DIM pour le tableau. Rappelez-vous que la totalité du tableau occupe de la place dans la mémoire et, en conséquence, ne créez pas de tableaux plus grands que necessaire.

Multipliez le nombre total d'éléments dans chaque dimension du tableau pour représenter le nombre de variables crées avec DIM.

```
1Ø DIM A$(4Ø), B7(15), CC%(4,4,4)
```

41 éléments 16 éléments 125 éléments

Vous pouvez dimensionner plus d'un tableau dans une instruction DIM. Cependant, veillez à ne pas dimensionner plusieurs fois le même tableau.

END

Lorsqu'un programme rencontre une instruction END, le programme s'arrête, comme s'il manquait de lignes. Vous pouvez utiliser CONT pour redémarrer le programme.

FOR . . . TO . . . STEP

Cette instruction agit avec l'instruction NEXT pour répéter une section du programme un nombre déterminé de fois. Le format est le suivant:

FOR (nom de variable) = (démarrage du comptage) TO (fin du comptage) STEP (variable de boucle) = (comptage terminé).

La variable de boucle est incrémentée ou décrémentée pendant le programme. En l'absence de STEP (PAS) spécifié, l'on suppose que STEP soit égal à 1. Le démarrage du comptage et la fin du comptage sont les limites de la valeur de la variable de boucle.

```
1Ø FOR L = 1 TO 1Ø STEP .1
2Ø PRINT L
3Ø NEXT L
```

La fin de la valeur de la boucle peut être suivie du mot STEP et d'un autre nombre ou variable. Dans ce cas, la valeur suivant STEP est ajoutée chaque fois à la place de 1. Ceci vous permet de compter en arrière ou de compter par fractions.

GET

L'instruction GET vous permet d'obtenir des données à partir du clavier, un caractère à la fois. Lorsque GET est exécuté, le caractère qui est frappé est assigné à la variable. Si aucun caractère n'est frappé, un caractère zéro (blanc) est assigné.

GET est suivi d'un nom de variable, généralement une variable-chaîne. Si une variable numérique a été utilisée et si une touche non numérique est pressée, le programme s'arrêtera par un message d'erreur. L'instruction GET peut être placée dans une boucle en contrôlant tout résultat nul (blanc). Cette boucle se poursuit jusqu'à ce qu'une touche soit pressée.

```
1Ø GET A$: IF A$ = «» THEN 10
```

GET#

L'instruction GET# est utilisée avec un périphérique ou un fichier auparavant OPENed (ouvert) pour entrer un caractère à la fois à partir de ce periphérique ou de ce fichier.

```
GET#1,A$
```

Ceci entrerait un caractère d'un fichier de données.

GOSUB

Cette instruction est similaire à GOTO, mis à part que l'ordinateur se rappelle de la ligne de programme qu'il a exécutée en dernier lieu avant GOSUB. Lorsqu'une ligne comportant une instruction RETURN est rencontrée, le programme revient à l'instruction suivant immédiatement GOSUB. Ceci est utile lorsqu'il existe dans votre programme une routine qui se présente dans plusieurs parties du programme. Au lieu de retaper constamment la routine, vous pouvez vous servir de GOSUB chaque fois que la routine est nécessaire.

```
2Ø GOSUB 8ØØ
```

GOTO OU GO TO

Lorsqu'une instruction comportant la commande GOTO est atteinte, la ligne suivante à exécuter sera celle comportant le numéro de ligne suivant le mot GOTO.

IF . . . THEN

IF . . . THEN permet à l'ordinateur d'analyser une situation et d'entamer deux voies d'action possibles suivant le résultat. Si l'expression est confirmée, l'instruction suivant THEN est alors exécutée. Ceci peut être une instruction BASIC.

Si l'expression n'est pas confirmée, le programme passe directement à la ligne suivante.

L'expression évaluée peut être une variable ou une formule, dans ce cas elle est considérée comme étant confirmée si elle n'est pas nulle et non confirmée (fausse) si elle est nulle. Dans la plupart des cas, il s'agit d'une expression impliquant les opérateurs relationnels (=, <, >, <=, =>, <>, AND, OR, NOT).

```
1Ø IF X>Ø THEN END
```

INPUT

L'instruction INPUT permet au programme d'obtenir des données de l'utilisateur en assignant ces données à une variable. Le programme s'arrête, affiche un point d'interrogation (?) sur l'écran et attend que l'utilisateur frappe la réponse et appuie sur la touche RETURN. INPUT est suivi d'un nom de variable, ou d'une liste de noms de variables, séparés par des virgules. Un message peut être placé à l'intérieur de guillemets avant que la liste des noms de variables soit entrée (INPUT). Si plus d'une variable doit être entrée (INPUT), elles doivent être séparées par des virgules lorsqu'elles sont frappées.

INPUT#

INPUT# est similaire à INPUT mais tire les données d'un fichier ou d'un périphérique précédemment OPENed (ouvert).

LET

LET n'est presque jamais utilisé dans des programmes vu qu'il est optionnel, mais cette instruction est le coeur de tous les programmes BASIC. Le nom de la variable qui doit être assigné, le résultat d'un calcul, est sur le côté gauche du signe égal et la formule à droite.

NEXT

NEXT est toujours utilisé en liaison avec l'instruction FOR. Lorsque le programme atteint une instruction NEXT, il contrôle l'instruction FOR pour déterminer si la limite de la boucle a été atteinte. Si la boucle n'est pas terminée, la variable de la boucle est augmentée de la valeur STEP spécifiée. Si la boucle est terminée, l'exécution progresse avec l'instruction suivant NEXT.

NEXT peut être suivi d'un nom de variable, ou de la liste de noms de variables, séparés par des virgules. S'il n'y a pas de noms mentionnés, la dernière boucle démarrée est celle achevée. Si des variables sont données, elles sont achevées dans l'ordre de gauche à droite.

```
1Ø FOR X=1 TO 1ØØ : NEXT
```

ON

Cette commande transforme les commandes GOTO et GOSUB en versions spéciales de l'instruction IF. ON est suivi d'une formule qui est évaluée. Si le résultat du calcul est 1, la première ligne de la liste est exécutée; si le résultat est 2, la seconde ligne est exécutée, et ainsi de suite. Si le résultat est 0, négatif ou supérieur à la liste des nombres, la dernière ligne exécutée sera l'instruction suivant l'instruction ON.

```
1Ø INPUT X
2Ø ON X GOTO 1Ø,2Ø,3Ø,4Ø,5Ø
```

OPEN

L'instruction OPEN permet au COMMODORE 64 d'accéder à des périphériques, tels que le lecteur enregistreur de cassette et le disque de données, une imprimante ou même l'écran. OPEN est suivi d'un nombre (0-255) qui est le nombre auquel toutes les instructions qui suivent se réfèrent. Il s'agit généralement d'un second nombre après le premier qui est le numéro du périphérique.
Les numéros de périphériques sont les suivants:

Ø	Ecran
1	Cassette
4	Imprimante
8	Disque

Après le numéro du périphérique peut se trouver un troisième nombre à nouveau séparé par une virgule qui est l'adresse secondaire. Dans le cas de la cassette, il s'agit de 0 pour lecture, 1 pour enregistrement et 2 pour enregistrement avec marqueur de fin de bande.

Dans le cas du disque, le nombre se réfère à la mémoire tampon ou au numéro de canal. Dans l'imprimante, l'adresse secondaire commande les caractéristiques, telles que l'impression étendue. Pour plus de détails, voir le manuel de référence du programmeur du COMMODORE 64.

1Ø OPEN 1,0	OPENs (ouvre) l'écran (SCREEN) comme un périphérique
2Ø OPEN 2,1,Ø,«D»	OPENs (ouvre) la cassette pour la lecture, le fichier à rechercher est D
3Ø OPEN 3,4	OPENs (ouvre) l'imprimante
4Ø OPEN 4,8,15	OPENs (ouvre) le canal des données sur le disque

Voir également: CLOSE, CMD, GET#, INPUT# et PRINT#, la variable de système ST et l'annexe B.

POKE

POKE est toujours suivi de deux nombres ou formules. La première position est un emplacement de mémoire; la seconde position est une valeur décimale de 0 à 255 qui doit être placée dans l'emplacement de la mémoire pour remplacer toute valeur auparavant mémorisée.

```
1Ø POKE 53281,Ø
2Ø S = 4Ø96*13
3Ø POKE S+29,8
```

PRINT

L'instruction PRINT est la première que la plupart des gens apprennent à utiliser, mais il existe un certain nombre de variations auquelles il faut faire attention. PRINT peut être suivi de:

Chaîne-texte avec des guillemets
Noms de variables
Fonctions
Signes de ponctuation.

Les signes de ponctuation sont utilisés pour faciliter le formatage des données sur l'écran. La virgule divise l'écran en 4 colonnes, alors que le point-virgule supprime tous les espacements. Aucun signe ne peut être le dernier symbole d'une ligne. Ceci fait que l'affichage de l'exemple suivant est réalisée comme s'il y avait une continuation de la même instruction PRINT.

```
10 PRINT «HALLO»
2Ø PRINT «HALLO»,A$
```

3Ø PRINT A+B
4Ø PRINT J;
6Ø PRINT A,B,C,D
Voir également: les fonctions POS, SPC et TAB

PRINT#

Il n'y a pas une grande différence entre cette instruction et PRINT. PRINT# est suivi d'un nombre qui se réfère au périphérique ou au fichier de données précédemment OPENed (ouvert). Ce nombre est suivi d'une virgule et une liste doit être affichée. La virgule et le point-virgule ont le même effet que s'ils se trouvaient dans PRINT. Veuillez remarquer que certains périphériques ne fonctionnent pas avec TAB et SPC.

1ØØ PRINT#1, «DATEN INHALTE»;A%,B1,C$

READ

READ est utilisé pour assigner les informations provenant des instructions DATA à des variables, de telle manière que les informations soient utilisées. L'on doit veiller à éviter les chaînes de lecture (READing) où READ est supposé d'être un nombre, ce qui entraînerait un message d'erreur TYPE MISMATCH ERROR (désaccord de frappe).

REM (remarque)

REMark est un rappel à celui qui lit une LISTe du programme. Cette instruction peut expliquer une partie du programme ou fournir des instructions supplémentaires. Les instructions REM n'affectent en aucune manière le fonctionnement du programme sauf qu'elle augmente sa longueur. REM peut être suivi de n'importe quel texte.

RESTORE

Lorsque cette instruction est exécutée dans un programme, le pointeur auquel un élément d'une instruction DATA sera ensuite lu (READ) est réinitialisé sur le premier élément de la liste. Ceci vous confère la possibilité de re-READ (relire) les informations. RESTORE est placé indépendamment sur une ligne.

RETURN

Cette instruction est toujours utilisée en liaison avec GOSUB. Lorsque le programme rencontre une instruction RETURN, il passe immédiatement à l'instruction suivant la commande GOSUB. Si aucune commande GOSUB n'a été auparavant générée, un message d'erreur RETURN WITHOUT GOSUB ERROR (RETURN SANS GOSUB) apparaît.

STOP

Cette instruction arrête l'exécution du programme. Le message BREAK IN xxx sera affiché où xxx est le numéro de ligne contenant STOP. Le programme peut être redémarré en utilisant la commande CONT. STOP est normalement utilisé dans la mise au point d'un programme.

SYS

SYS est suivi d'un nombre décimal ou d'une valeur numérique comprise entre 0 et 65535. Le programme commencera alors l'exécution du programme en langage machine en commencant à partir de cet emplacement de la mémoire. C'est similaire à la fonction USR mais ne permet pas le passage de paramètres.

WAIT

WAIT est utilisé pour arrêter le programme jusqu'à ce que les contenus d'un emplacement de la mémoire se modifient d'une façon spécifique. WAIT est suivi d'un emplacement de mémoire (X) comportant jusqu'à deux variables. Le format est le suivant:

WAIT X,Y,Z

Les contenus de l'emplacement de la mémoire sont en premier lieu traités avec l'instruction logique exclusive ou avec le troisième nombre, s'il est présent, et ensuite avec AND avec le second nombre. Si le résultat est nul, le programme revient à cet emplacement de la mémoire et effectue à nouveau le contrôle. Lorsque le résultat est différent de zéro, le programme est poursuivi avec l'instruction suivante.

FONCTIONS NUMERIQUES

ABS(X) (valeur absolue)

ABS fournit la valeur absolue du nombre, sans son signe (+ ou –). La réponse est toujours positive.

ATN(X) (arctangente)

Elle fournit l'angle mesuré en radians dans la tangente X.

COS(X) (cosinus)

Elle fournit la valeur du cosinus de X, où X est un nombre mesuré en radians.

EXP(X)

Elle fournit la valeur de la constante mathématique e (. . .) élevée à la puissance de X.

FNxx(X)

Elle fournit la valeur de la fonction xx définie par l'utilisateur créée dans une instruction DEF FNxx(X).

INT(X)

Elle fournit la valeur tronquée de X, c'est-à-dire avec toutes les positions décimales à droite du point décimal éliminées. Le résultat est toujours inférieur à ou égal à X. En conséquence, tous les nombres négatifs, avec des positions décimales, deviennent des entiers inférieurs à la valeur courante.

LOG(X) (logarithme)

Elle fournit le log naturel de X. Le log naturel de la base e (voir EXP(X)). Pour convertir en log base 10, diviser simplement par LOG(10).

PEEK(X)

Elle est utilisée pour trouver les contenus de l'emplacement X de la mémoire compris entre 0 et 65535 en fournissant un résultat compris entre 0 et 255. PEEK est fréquemment utilisé en liaison avec l'instruction POKE.

RND(X) (nombre aléatoire)

RND(X) fournit un nombre aléatoire compris entre 0 et 1. Le premier nombre aléatoire doit être généré par la formule RND(-TI) pour que les choses commencent différemment chaque fois. Après quoi, X doit être I ou n'importe quel nombre positif. Si X est nul, le résultat sera le même nombre aléatoire que le dernier. Une valeur négative de X réinitialise le générateur. L'utilisation du même nombre négatif pour X entraîne la même séquence de nombres «aléatoires». La formule de génération d'un nombre compris entre X et Y est la suivante:

N = INT (RND(1)*Y)+X

où Y est la limite supérieure
X est la limite inférieure des nombres souhaités.

SGN(X) (signe)

Cette fonction fournit le signe (positif, négatif ou nul) de X. Le résultat sera +1 s'il est positif, 0 s'il est nul et –1 s'il est négatif.

SIN(X) (sinus)

SIN(X) est la fonction sinusoîdale trigonométrique. Le résultat sera le sinus X où X est un angle exprimé en radians.

SQR(X) (racine carrée)

Cette fonction fournit la racine carrée de X où X est un nombre positif ou 0. Si X est négatif, il en résulte le message d'erreur ILLEGAL QUANTITY ERROR (erreur quantité illégale).

TAN(X) (tangente)

Le résultat sera la tangente de X où X est un angle en radians.

USR(X)

Lorsque cette fonction est utilisée, le programme saute un programme en langage machine dont le point de départ est contenu dans des emplacements de la mémoire. Le paramètre X est amené au programme en langage machine qui fournit de nouveau une autre valeur au programme BASIC. Se reporter au manuel de référence du programmeur COMMODORE 64 pour plus de détails sur cette fonction et la programmation en langage machine.

FONCTIONS – CHAINES

ASC(X$)

Cette fonction fournit le code ASCII du premier caractère de X$.

CHR$(X)

Il s'agit de l'opposé de ASC et fournit un caractère de chaîne dont le code ASCII est X.

LEFT$(X$,X)

Elle fournit une chaîne contenant les caractères X les plus à gauche de X$.

LEN(X$)

Le nombre de caractères (y compris les espaces et les autres symboles) de la chaîne X$ sera fourni.

MID$(X$,S,X)

Elle fournit une chaîne contenant X caractères en partant du Sème caractère de X$.

RIGHT$(X$,X)

Fournit le caractère X le plus à droite de X$.

STR$(X)

Elle fournit une chaîne qui est identique à la version affichée de X.

VAL(X$)

Cette fonction convertit X$ en un nombre et essentiellement l'opération inverse de STR$. La chaîne est examinée à partir du caractère le plus à gauche en direction de la droite, tant que les caractères sont dans un format de nombre

1Ø X = VAL (,,123.456")	X = 123.456
1Ø X = VAL (,,12A13B")	X = 12
1Ø X = VAL (,,RIUØ17")	X = Ø
1Ø X = VAL (,,–1.23.15.67")	X = –1.23

FRE(X)

Cette fonction fournit le nombre d'octets inutilisés disponibles dans la mémoire, quelle que soit la valeur de X.

POS(X)

Cette fonction fournit le numéro de la colonne (0-39) sur laquelle la prochaine instruction PRINT commencera sur l'écran. X peut avoir une valeur quelconque et n'est pas utilisé.

SPC(X)

Cette fonction est utilisée dans une instruction PRINT pour faire sauter les espaces X en avant.

TAB(X)

TAB est également utilisé dans une instruction PRINT. La position suivante à afficher se trouvera dans la colonne X.

ANNEXE D

ABREVIATIONS DES MOTS CLES BASIC

Pour gagner du temps lors de la frappe des programmes et commandes, le BASIC COMMODORE 64 permet à l'utilisateur d'abréger la plupart des mots clés. L'abréviation de PRINT est un point d'interrogation. Les abréviations des autres mots sont obtenues en frappant la première ou la seconde lettre du mot suivie de la lettre suivante SHIFTed (décalée) du mot. Si les abréviations sont utilisées dans une ligne de programme, le mot clé sera sorti sur la liste sous forme complète. Remarquez que certains des mots clés, lorsqu'ils sont abrégés, comprennent une parenthèse gauche.

Commande	Abréviation	Représentation SUR L'ÉCRAN
ABS	A SHIFT B	A
AND	A SHIFT	A
ASC	A SHIFT S	A
ATN	A SHIFT T	A
CHR$	C SHIFT H	C
CLOSE	CL SHIFT O	CL
CLR	C SHIFT L	C
CMD	C SHIFT M	C
CONT	C SHIFT O	C
DATA	D SHIFT A	D
DEF	D SHIFT E	D
DIM	D SHIFT I	D
END	E SHIFT N	E
EXP	E SHIFT X	E

Commande	Abréviation	Représentation SUR L'ÉCRAN
FOR	F SHIFT O	F
FRE	F SHIFT R	F
GET	G SHIFT E	G
GOSUB	GO SHIFT S	GO
GOTO	SHIFT O	G
INPUT#	I SHIFT N	I
LET	L SHIFT E	L
LEFT$	LE SHIFT F	LE
LIST	L SHIFT I	L
LOAD	L SHIFT O	L
MID$	M SHIFT I	M
NEXT	N SHIFT E	N
NOT	N SHIFT O	N
OPEN	O SHIFT P	O

Commande	Abréviation	Représentation SUR L'ÉCRAN
PEEK	P SHIFT E	P
POKE	P SHIFT O	P
PRINT	?	?
PRINT#	P SHIFT R	P
READ	R SHIFT E	R
RESTORE	RE SHIFT S	RE
RETURN	RE SHIFT T	RE
RIGHT$	R SHIFT I	R
RND	R SHIFT N	R
RUN	R SHIFT	R
SAVE	S SHIFT A	S
SGN	S SHIFT G	S
SIN	S SHIFT I	S

Commande	Abréviation	Représentation SUR L'ÉCRAN
SPC(	S SHIFT P	S
SQR	S SHIFT Q	S
STEP	ST SHIFT E	ST
STOP	S SHIFT T	S
STR$	ST SHIFT R	ST
SYS	S SHIFT Y	S
TAB	T SHIFT A	T
THEN	T SHIFT H	T
USR	U SHIFT S	U
VAL	V SHIFT A	V
VERIFY	V SHIFT E	V
WAIT	W SHIFT A	W

ANNEXE E

CODES D'AFFICHAGE SUR L'ECRAN

Le tableau suivant fournit une liste de tous les caractères incorporés dans le jeu de caractères du COMMODORE 64. Il indique les nombres qui doivent être POKEd (mis) dans la mémoire de l'écran (emplacements 1024-2023) pour obtenir un caractère voulu. Le caractère auquel correspond un nombre PEEKed (supprimé) de l'écran est également indiqué.

Deux jeux de caractères sont disponibles, mais seul un jeu à la fois. Ceci signifie que vous ne pouvez avoir de caractères d'un jeu sur l'écran et simultanément avoir des caractères de l'autre jeu affichés. Les jeux sont commutés en appuyant simultanément sur les touches SHIFT et COMMODORE.

A partir du BASIC, POKE 53272,29 passe au mode case supérieure et POKE 53272,31 passe au mode case inférieure. Tout nombre sur le tableau peut également être représenté de façon inverse (REVERSE). Le code caractère inversé peut être obtenu en ajoutant 128 aux valeurs indiquées.

Si vous souhaitez afficher un cercle plein sur l'emplacement 1504, POKE (mettre) le code du cercle (81) sur l'emplacement 1504: POKE 1504,81.

Ceci est un emplacement de mémoire correspondant pour contrôler la couleur de chaque caractère affiché sur l'écran (emplacements . . .). Pour changer la couleur du cercle au jaune (code de couleur 7), vous devriez POKE (mettre) le caractère de couleur dans l'emplacement correspondant de la mémoire (57776): POKE 55576,7.

Se reporter à l'annexe G en ce qui concerne les cartes complètes des mémoires de l'écran et des couleurs avec les codes de couleurs.

CODES DE L'ECRAN

Jeu 1	Jeu 2	Poke	Jeu 1	Jeu 2	Poke	Jeu 1	Jeu 2	Poke
@		0	C	c	3	F	f	6
A	a	1	D	d	4	G	g	7
B	b	2	E	e	5	H	h	8
I	i	9	%		37		A	65
J	j	10	&		38		B	66
K	k	11	'		39		C	67
L	l	12	(		40		D	68
M	m	13	)		41		E	69
N	n	14	*		42		F	70
O	o	15	+		43		G	71
P	p	16	,		44		H	72
Q	q	17	–		45		I	73
R	r	18	.		46		J	74
S	s	19	/		47		K	75
T	t	20	0		48		L	76
U	u	21	1		49		M	77
V	v	22	2		50		N	78
W	w	23	3		51		O	79
X	x	24	4		52		P	80
Y	y	25	5		53		Q	81
Z	z	26	6		54		R	82
[		27	7		55		S	83
£		28	8		56		T	84
]		29	9		57		U	85
↑		30			58		V	86
←		31	;		59		W	87
SPACE		32	<		60		X	88
!		33	=		61		Y	89

Jeu 1	Jeu 2	Poke	Jeu 1	Jeu 2	Poke	Jeu 1	Jeu 2	Poke
"		34	>		62		Z	90
#		35	?		63			91
$		36			64			92
		93			105			117
		94			106			118
		95			107			119
SPACE		96			108			120
		97			109			121
		98			110			122
		99			111			123
		100			112			124
		101			113			125
		102			114			126
		103			115			127
		104			116			

Les codes de 128 à 255 sont les images inversées des codes de 0 à 127.

ANNEXE F

CODES ASCII ET CHR$

Cette annexe vous indique quels sont les caractères qui apparaissent si vous affichez (PRINT) CHR$(X) pour toutes les valeurs possibles de X. Elle montre également les valeurs obtenues en frappant PRINT ASC(«X») où X est un caractère quelconque que vous pouvez frapper. C'est utile dans l'évaluation du caractère reçu dans une instruction GET en convertissant la case supérieure/inférieure et en imprimant les commandes basées sur des caractères (tels que passage à la case supérieure/inférieure) qui pourraient ne pas être placées entre guillemets.

Signe	CHR$	Signe	CHR$	Signe	CHR$	Signe	CHR$
	0		17	"	34	3	51
	1		18	#	35	4	52
	2		19	$	36	5	53
	3		20	%	37	6	54
	4		21	&	38	7	55
	5		22	·	39	8	56
	6		23	(	40	9	57
	7		24	)	41		58
disabilita 	8		25	*	42	;	59
abilita 	9		26	+	43	<	60
	10		27	,	44	=	61
	11		28	–	45	>	62
	12		29	.	46	?	63
	13		30	/	47	@	64
	14		31	0	48	A	65
	15		32	1	49	B	66
	16	!	33	2	50	C	67

Signe	CHR$	Signe	CHR$	Signe	CHR$	Signe	CHR$
D	68		97		126		155
E	69		98		127		156
F	70		99		128		157
G	71		100		129		158
H	72		101		130		159
I	73		102		131		160
J	74		103		132		161
K	75		104	f1	133		162
L	76		105	f3	134		163
M	77		106	f5	135		164
N	78		107	f7	136		165
O	79		108	f2	137		166
P	80		109	f4	138		167
Q	81		110	f6	139		168
R	82		111	f8	140		169
S	83		112		141		170
T	84		113		142		171
U	85		114		143		172
V	86		115		144		173
W	87		116		145		174
X	88		117		146		175
Y	89		118		147		176
Z	90		119		148		177
[	91		120		149		178
£	92		121		150		179
]	93		122		151		180
↑	94		123		152		181
←	95		124		153		182
	96		125		154		183

Signe	CHR$	Signe	CHR$	Signe	CHR$	Signe	CHR$
	184		186		188		190
	185		187		189		191

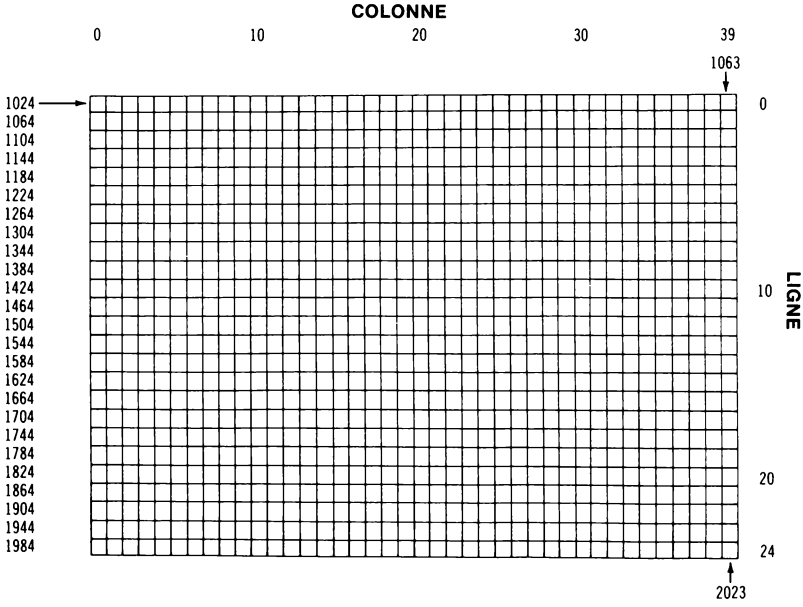
Codes	**192-233**	**comme Codes**	**96-127**
Codes	**224-254**	**comme Codes**	**160-190**
Code	**255**	**comme Code**	**126**

ANNEXE G

MEMOIRES DE L'ECRAN ET DES COULEURS

Les tableaux suivants donnent une liste des emplacements de la mémoire qui commandent la mise en place des caractères sur l'écran et des emplacements utilisés pour modifier les couleurs des différents caractères, de même que l'indication des codes de couleur des caractères.

MEMOIRE DE L'ECRAN

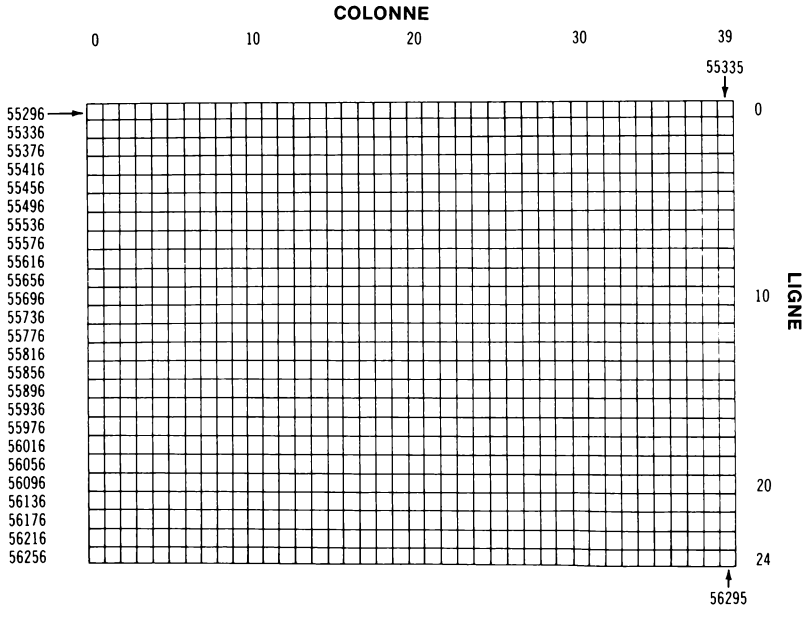


Les valeurs effectives à mettre (POKE) dans un emplacement de mémoire de couleur pour modifier la couleur d'un caractère sont les suivantes:

0	NOIR	8	ORANGE
1	BLANC	9	BRUN
2	ROUGE	10	ROUGE clair
3	TURQUOISE	11	GRIS 1
4	POURPRE	12	GRIS 2
5	VERT	13	VERT clair
6	BLEU	14	BLEU clair
7	JAUNE	15	GRIS 3

Par exemple, pour modifier la couleur d'un caractère situé dans le coin supérieur gauche de l'écran en rouge, frapper: POKE 55296,2.

MEMOIRE DES COULEURS



ANNEXE H

DERIVATION DE FONCTIONS MATHEMATIQUES

Les fonctions qui ne sont pas intrinsèques au BASIC COMMODORE 64 peuvent être calculées comme suit:

FONCTION	EQUIVALENT BASIC
SECANTE	SEC(X) = 1/COS(X)
COSECANTE	CSC(X) = 1/SIN(X)
COTANGENTE	COT(X) = 1/TAN(X)
SINUS INVERSE	ARSIN(X) = ATN(X/SQR(1−X↑2))
COSINUS INVERSE	ARCOS(X) = −ATN(X/SQR(1−X↑2))+π/2
COTANGENTE INVERSE	ARCOT(X) = ATN(X)+π/2
SECANTE INVERSE	ARSEC(X) = ATN(X/SQR(X↑2−1))
COSECANTE INVERSE	ARCSC(X) = ATN(X/SQR(X↑2−1)) +(SGN(X)−1)•π/2
SINUS HYPERBOLIQUE	SINH(X) = (EXP(X)−EXP(−X))/2
COSINUS HYPERBOLIQUE	COSH(X) = (EXP(X)+EXP(−X))/2
TANGENTE HYPERBOLIQUE	TANH(X) = EXP(−X)/(EXP(X)+EXP (−X))•2+1
COTANGENTE HYPERBOLIQUE	COTH(X) = EXP(−X)/(EXP(X)−EXP (−X))•2+1
SECANTE HYPERBOLIQUE	SECH(X) = 2/(EXP(X)+EXP(−X))
COSECANTE HYPERBOLIQUE	CSCH(X) = 2/(EXP(X)−EXP(−X))
SINUS HYPERBOLIQUE INVERSE	ARSINH(X) = LOG(X+SQR(X↑2+1))
COSINUS HYPERBOLIQUE INVERSE	ARCOSH(X) = LOG(X+SQR(X↑2−1))
TANGENTE HYPERBOLIQUE INVERSE	ARTANH(X) = LOG((1+X)/(1−X))/2
COTANGENTE HYPERBOLIQUE INVERSE	ARCOTH(X) = LOG((X+1)/(X−1))/2
SECANTE HYPERBOLIQUE INVERSE	ARSECH(X) = LOG((SQR(1−X↑2)+1)/X)
COSECANTE HYPERBOLIQUE INVERSE	ARCSCH(X) = LOG((SQR(1+X↑2)+1)/X) •SGN(X)

ANNEXE I

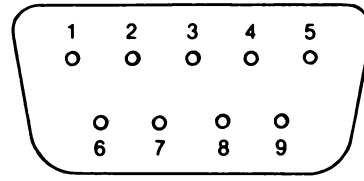
SORTIES DE BROCHES DES PERIPHERIQUES D'ENTREE/SORTIE

Cette annexe est conçue pour vous présenter les connexions qui peuvent être établies sur votre COMMODORE 64.

1) Entrée/sortie jeu
2) Fente chargeur de programmes
3) Audio/vidéo
4) Entrée/sortie sérielles (disque/imprimante)
5) Sortie modulateur
6) Cassette
7) Port utilisateur

Port commande 1

Broche	Type	Remarque
1	JOYA0	
2	JOYA1	
3	JOYA2	
4	JOYA3	
5	POT AY	
6	BUTTON A/LP	
7	+5V	MAX. 100mA
8	GND	
9	POT AX	



Port commande 2

Broche	Type	Remarque
1	JOYB0	
2	JOYB1	
3	JOYB2	
4	JOYB3	
5	POT BY	
6	BUTTON B	
7	+5V	MAX. 100mA
8	GND	
9	POT BX	

* Bouton = Bouton fen du levier de commande
** POT = Potentiomètre du Paddle

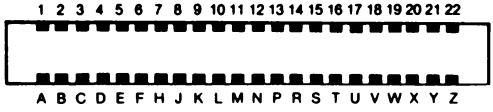
Fente d'extension du chargeur de programmes

Broche	Type
22	GND
21	CD0
20	CD1
19	CD2
18	CD3
17	CD4
16	CD5
15	CD6
14	CD7
13	DMA
12	BA

Broche	Type
11	ROML
10	I/O2
9	EXROM
8	GAME
7	I/O1
6	Dot Clock
5	CR/W
4	IRQ
3	+5V
2	+5V
1	GND

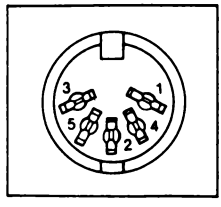
Broche	Type
Z	GND
Y	CA0
X	CA1
W	CA2
V	CA3
U	CA4
T	CA5
S	CA6
R	CA7
P	CA8
N	CA9

Broche	Type
M	CA10
L	CA11
K	CA12
J	CA13
H	CA14
F	CA15
E	S02
D	NMI
C	RESET
B	ROMH
A	GND



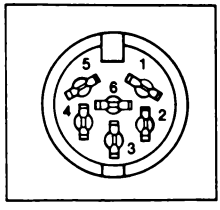
Audio/vidéo

Broche	Type
1	LUMINOSITE
2	MASSE
3	SORTIE AUDIO
4	SORTIE VIDEO
5	ENTREE AUDIO



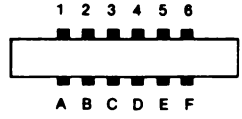
Entrée/sortie sérielles

Broche	Type
1	SRQIN SERIALE
2	MASSA
3	IN/OUT ATN SERIALE
4	IN/OUT CLK SERIALE
5	IN/OUT DATI SERIALI
6	RESET



Cassette

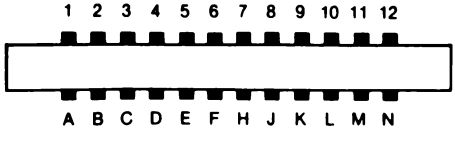
Broche	Type
A-1	MASSE
B-2	+5V
C-3	MOTEUR CASSETTE
D-4	LECTURE CASSETTE
E-5	ENREGISTREMENT CASSETTE
F-6	DETECTION CASSETTE



Entrée/sortie utilisateur

Broche	Type	Remarque
1	GND	
2	+5V	MAX. 100 mA
3	RESET	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	PC2	
9	SER. ATN IN	
10	9 VAC	MAX. 100 mA
11	9 VAC	MAX. 100 mA
12	GND	

Broche	Type	Remarque
A	GND	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	



ANNEXE J

Nous avons inclu un certain nombre de programmes utiles pour vous permettre d'essayer votre COMMODORE 64. Ces programmes s'avèront aussi bien divertissant qu'utiles.

PROGRAMMES A ESSAYER

```
100 PRINT"■■■■■■■■■    TROUVER LE MOT CLE       "
110 INPUT"■REGLES DU JEU (O/N)";Z$:IFASC(Z$)=78THEN170
120 PRINT"■TU DOIS TROUVER LE MOT CLE SECRET!"
130 PRINT"■IL CONTIENT 5 LETTRES"
140 PRINT"■APRES CHAQUE ESSAIS TU SAURAS,"
150 PRINT"■COMBIEN DE LETTRES SONT JUSTES."
160 PRINT"■UN CONSEIL: PAS DEVINER SANS REFLECHIR"
170 DATA  BCPSE,BMMFF,BSSFU,CFUPO,CJKPV
180 DATA  CPVMF,DBOBM,DIBNQ,DPFVS,EFCJU
190 DATA  EJWJO,FMFWF,FYDFT,GFUFS,GJYFS
200 DATA  HBSEF,HSFWF,IPNNF,JTBSE,KBNCF
210 DATA  KBEJT,MBNQF,MPZFS,NBMJO,NPVMF
220 DATA  OBQQF,OPFVE,PODMF,PVUJM,QBSJT
230 DATA  QFUJU,RVBOE,SBGMF,SPEFS,TDPSF
240 DATA  TPDMF,UFJOU,USBJO,VMUSB,VOJPO
250 DATA  WBTUF,WJSJM,XBHPO,YFSFT,ZBDIU
260 DATA  [FCSF,TPFVS,SFWFS,QFQJO,EBOTF
270 N=50
280 DIMN$(N),Z(5),Y(5)
290 FORJ=1TON:READN$(J):NEXTJ
300 T=TI
310 T=T/1000:IFT>=1THEN310
320 Z=RND(-T)
330 G=0:N$=N$(RND(1)*N+1)
340 PRINT"■■DEVINE LE MOT CLE!":IFR>0THEN380
350 PRINT"■C'EST UN MOT SIGNIFICATIF DE 5 LETTRES"
360 PRINT"TU DEVINES ET MOI JE TE DIS,"
370 PRINT"COMBIEN DE LETTRES SONT JUSTES"
380 G=G+1:INPUT"TON ESSAI";Z$
390 IFLEN(Z$)<>5THENPRINT"■5 LETTRES!":GOTO380
```

```
400 V=0:M=0:H=0
410 FORJ=1TO5
420 Z=ASC(MID$(Z$,J,1)):Y=ASC(MID$(N$,J,1))-1:IFY=64THENY=90
430 IFZ<65ORZ>90THENPRINT"■CODE INVALABLE!":GOTO380
440 IFZ=65ORZ=69ORZ=73ORZ=79ORZ=85ORZ=89THENV=V+1
450 IFZ=YTHENM=M+1
460 Z(J)=Z:Y(J)=Y:NEXTJ
470 IFM=5THEN580
480 IFV=0ORV=5THENPRINT"■QUELLE BETISE!":GOTO380
490 FORJ=1TO5:Y=Y(J)
500 FORK=1TO5:IFY=Z(K)THENH=H+1:Z(K)=0:GOTO520
510 NEXTK
520 NEXTJ
530 PRINT"●I■■■■■■■■■■■■■■■■■■■■■■■■■■I";H;"LETTRES JUSTES"
540 IFG<30THEN380
550 PRINT"■RENONCE! LE MOT CLE S'APPELLE '";
560 FORJ=1TO5:PRINTCHR$(Y(J));:NEXTJ
570 PRINT"=':GOTO590
580 PRINT"TU AS REUSSI EN"G"ESSAIS!"
590 INPUT"■UN NOUVEAU MOT";Z$
600 R=1:IFASC(Z$)<>78THEN330
```

```
100 REM MICHAEL ROW THE BOAT ASHORE
110 SI=54272:W1=SI+4:W2=SI+11:W3=SI+18
120 F0(0)=3000:F0(1)=1508:F0(2)=6030
130 D0=180
140 HT=2↑(1/12):REM DEMI PAS
150 DEFFNH(F)=INT(FF/256):REM HAUTE FREQUENCE
160 DEFFNL(F)=F-256*FNH(F):REM BASSE FREQUENCE
170 FORK=0TO20:READX:POKESI+K,X:NEXT:REM TIMBRE
180 READWA,WB,WC:REM FORMES D'ONDES
190 DIMF(100,6)
200 PRINT"J VOIX1   VOIX2   VOIX3   DUREE"
210 N=N+1:FORK=0TO2:READF :REM NOTE DEPUIS DATA
220 F=INT(F0(K)*HT↑F+.5):HB=FNH(F):LB=FNL(F):PRINTHB;LB,:REM FREQUENCE H/B
230 F(N,2*K)=LB:F(N,2*K+1)=HB
240 NEXTK
250 READD:PRINT"   "D:F(N,6)=D:REM DUREE
260 IFD>0THEN210
270 POKESI+24,15
280 FORI=1TON-1
290 FORK=0TO2:POKESI+7*K,F(I,2*K):POKESI+7*K+1,F(I,2*K+1)NEXTK:REM 3 FREQU.
300 POKEW1,WA:POKEW2,WB:POKEW3,WC:REM ALLUMER LES GENERATEURS
310 FORK=1TOF(I,6)*D0-150:NEXT
320 POKEW1,0:POKEW2,0:POKEW3,0:REM ETEINDRE LES GENERATEURS
330 FORK=1TO50:NEXT
340 NEXTI
350 GOTO280
360 DATA0,0,0,1,0,0,255:REM REGISTRE SID
370 DATA0,0,0,1,0,0,255
380 DATA0,0,0,1,0,0,255
390 DATA37,33,33:REM FORME D'ONDES
400 DATA0,0,0,2:REM MELODIE
410 DATA4,0,0,2
420 DATA7,4,0,3
430 DATA4,0,0,1
440 DATA7,4,0,2
450 DATA9,5,0,2
460 DATA7,4,-5,3
470 DATA-99,-99,-99,1
480 DATA4,0,0,2
490 DATA7,0,0,2
500 DATA9,5,5,8
510 DATA7,4,0,3
520 DATA-99,-99,-99,1
530 DATA4,0,0,2
540 DATA7,4,0,2
550 DATA7,4,0,3
560 DATA4,0,0,1
570 DATA5,2,0,2
580 DATA4,0,0,2
590 DATA2,-1,-5,3
600 DATA-99,-99,-99,1
610 DATA0,-5,0,2
620 DATA2,-1,0,2
630 DATA4,0,0,4
640 DATA2,-5,-5,4
650 DATA0,0,0,3
660 DATA-99,-99,-99,1
670 DATA-99,-99,-99,1
```

```
100 REM ALPHABETHE
110 DIMA$(26)
120 Z$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
130 Z1$="12345678901234567890123456"
140 PRINT"LE BUT DU JEU CONSISTE A RANGER UNE"
150 PRINT"SERIE DE LETTRES EN ORDRE ALPHABETIQUE."
160 PRINT"POUR REALISER CECI IL VOUS EST PERMIS"
170 PRINT"DE TOURNER A CHAQUE FOIS LE DEBUT.      BEAUCOUP DE CHANCE!"
180 PRINT"QUELLE LONGUEUR DOIT AVOIR LA SERIE "
190 INPUT"AU MAXIMUM ";S
200 IFS<1ORS>26THEN180
210 FORI=1TOS:A$(I)=MID$(Z$,I,1):NEXTI
220 REM ECHANGE PAR HASARD
230 FORI=1TOS:K=INT(RND(1)*S+1)
240 T$=A$(I):A$(I)=A$(K):A$(K)=T$
250 NEXTI
260 GOSUB470:T=0
270 REM TOURNER UNE PART DU STRING
280 T=T+1
290 INPUT"COMBIEN DOIS-JE TOURNER";R%
300 IFR%=0THEN440
310 IFR%>0ANDR%<=STHEN330
320 PRINT"DOIT SE TROUVER ENTRE 1 ET S!":GOTO290
330 R=INT(R%/2):FORI=1TOR
340 T$=A$(I)
350 T$=A$(I):A$(I)=A$(R%-I+1):A$(R%-I+1)=T$
360 NEXTI:GOSUB470
370 REM EXAMINER LA SERIE
380 C=1:FORI=2TOS
390 IFA$(I)>A$(I-1)THEN410
400 C=0
410 NEXTI
420 IFC=0THEN270
430 PRINT"REUSSI EN"T" ESSAIS!"
440 INPUT"ENCORE UNE FOIS";Y$
450 IFLEFT$(Y$,1)="O"ORY$="OK"ORY$="1"THEN180
460 END
470 PRINT:PRINTLEFT$(Z1$,S)
480 FORI=1TOS:PRINTA$(I);:NEXTI
490 PRINT"":RETURN
```

```
100 REM PIANO ELECTRONIQUE
110 PRINT"□ ▌▌▌▌ | ▌▌▌▌▌▌ | ▌▌▌▌ | ▌▌▌▌▌▌ | "
120 PRINT" ▌▌▌▌ | ▌▌▌▌▌▌ | ▌▌▌▌ | ▌▌▌▌▌▌ | "
130 PRINT" ▌▌▌▌ | ▌▌▌▌▌▌ | ▌▌▌▌ | ▌▌▌▌▌▌ | "
140 PRINT" ▌! ! ! ! ! ! ! ! ! ! ! ! ! ! !"
150 PRINT" ▌Q!W!E!R!T!Y!U!I!O!P!@!*!↑!;!=!"
160 PRINT"▌    'TOUCHE ESPACE':      PEDALE"
170 PRINT"▌'F1','F3','F5','F7':OCTAVE"
180 PRINT"▌'F2','F4','F6','F8':FORME D'ONDES
190 PRINT"▌PATIENCE, JE CALCULE"
200 SI=13*4096+1024:DIMF(26):DIMK(255)
210 FORI=0TO28:POKESI+I,0:NEXT
220 F1=7339:FORI=1TO26:F(27-I)=F1*5.8+30:F1=F1/2↑(1/12):NEXT
230 K$="1Q2W3ER5T6Y7UI9O0P@-*/↑▌;=":REM TOUCHES UTILISES
240 FORI=1TOLEN(K$):K(ASC(MID$(K$,I)))=I:NEXT
250 PRINT"□
260 AN=0:AB=0:HA=15:AU=9:HH=HA*16+AU:AS=AN*16+AB:WF=16:M=1:OK=4:HB=256:Z=0
270 FORI=0TO2:POKESI+5+I*7,AS:POKESI+6+I*7,HT
280 POKESI+2+I*7,4000AND255:POKESI+3+I*7,4000/256:NEXT
290 POKESI+24,15      +16+64:POKES+23,7:REM ENLEVER FILTRE
300 GETA$:IFA$=""THEN300
310 FR=F(K(ASC(A$)))/M:FL=SI+V*7:W=FL+4:IFFR=ZTHEN420
320 POKEFL+6,Z
330 POKEFL+5,Z
340 POKEW,8:POKEW,0:REM RESET
350 POKEFL,FR-HB*INT(FR/HB):REM BASSE FREQUENCE
360 POKEFL+1,FR/HB:REM HAUTE FREQUENCE
370 POKEFL+6,HH:REM MAINTENIR
380 POKEW,WF+1:FORI=1TO50*AN:NEXT
390 POKEW,WF
400 IFP=1THENV=V+1:IFV=3THENV=0
410 GOTO300
415 REM CONTROLE DESTOUCHES DE FONCTION
420 IFA$="▔"THENM=1:OK=4:GOTO300
430 IFA$="_"THENM=2:OK=3:GOTO300
440 IFA$="|"THENM=4:OK=2:GOTO300
450 IFA$=" '"THENM=8:OK=1:GOTO300
460 IFA$="\"THENWF=16:GOTO300
470 IFA$=" \"THENWF=32:GOTO300
480 IFA$="/"THENWF=64:GOTO300
490 IFA$=" /"THENWF=128:GOTO300
500 IFA$=" "THENP=1-P:HH=(HHANDNOT2)OR(NOTHHAND2):GOTO300
510 IFA$="□"THEN210
520 GOTO300

READY.
```

ANNEXE K

CONVERSION DE PROGRAMMES BASIC STANDARDS EN BASIC COMMODORE 64

Si vous avez des programmes rédigés dans un langage BASIC autre que le BASIC Commodore, certains ajustements mineurs peuvent s'avérer nécessaires avant de les passer sur le COMMODORE 64. Nous avons inclu certaines suggestions pour rendre la conversion plus facile.

Dimensions des chaînes

Supprimer toutes les instructions qui sont utilisées pour déclarer les longueurs des chaînes. Une instruction telle que DIM A$ (I,J) qui dimensionne un tableau de chaîne de J éléments de la longueur I, doit être convertie dans l'instruction BASIC Commodore DIM A$ (J).

Certains langages BASIC utilisent une virgule ou une abrévation pour la concaténation des chaînes. Certains d'eux doivent être modifiés en un signe plus, qui est l'opérateur BASIC Commodore pour la concaténation des chaînes.

Dans le BASIC COMMODORE 64, les fonctions MID$, RIGHT$ et LEFT$ sont utilisées pour prendre des sous-chaînes de chaînes. Des formules telles que A$ (I) pour accéder au I-ième caractère de A$ ou A$ (I,J) pour prendre une sous-chaîne de A$ de la position I à J, doivent être modifiées comme suit:

Autres langages BASIC	**BASIC COMMODORE 64**
A$(I) = X$	A$ = LEFT$(A$,I–1)+X$+MID$(A$,I+1)
A$(I,J) = X$	A$ = LEFT$(A$,I–1)+X$+MID$(A$,J+1)

Assignations multiples

Certains langages BASIC permettent des instructions de la forme suivante pour établir B et C égal à zéro:

```
1Ø LET B=C=Ø
```

Le BASIC COMMODORE 64 interpréterait le second signe égal comme un opérateur logique et établirait B = –1 si C = 0. Par contre, convertirait cette instruction en:

```
1Ø B=Ø : C=Ø
```

Attributions multiples

Certains langages BASIC utilisent une barre de fraction (/) pour séparer plusieurs instructions sur une ligne. Avec le BASIC COMMODORE 64, séparer toutes les instructions par (:).

Fonctions MAT

Les programmes utilisant les fonctions MAT disponibles dans certains langages BASIC doivent être rédigées à nouveau en utilisant des boucles FOR . . . NEXT pour être exécutés correctement.

ANNEXE L

MESSAGES D'ERREURS

Cette annexe comprend une liste des messages d'erreur générés par le COMMODORE 64, avec une description des causes.

BAD DATA (données erronées). Des données d'une chaîne ont été reçues d'un fichier ouvert, mais le programme attendait des données numériques.
BAD SUBSCRIPT (mauvais indice). Le programme essayait de se référer à un élément d'un tableau dont le numéro est à l'extérieur des limites spécifiées dans l'instruction DIM.
CAN'T CONTINUE (impossible de continuer). La commande CONT ne fonctionne pas, soit parce que le programme n'a jamais été passé (RUN), soit parce qu'il y a eu une erreur ou une ligne a été éditée.
DEVICE NOT PRESENT (périphérique non présent). Le périphérique d'entrée/sortie nécessaire n'était pas disponible pour une instruction OPEN, CLOSE, CMD, PRINT#, INPUT# ou GET#.
DIVISION BY ZERO (division par zéro). La division par zéro est une excentricité mathématique et n'est pas autorisée.
EXTRA IGNORED (supplément ignoré). Des éléments de données trop nombreux ont été frappés en réponse à une instruction ENTREE. Seuls les premiers éléments ont été acceptés.
FILE NOT FOUND (fichier non trouvé). Si vous cherchez un fichier sur une bande et trouvez une marque de fin de bande (END OF TAPE). Si vous cherchez sur un disque et qu'aucun fichier portant ce nom n'existe.
FILE NOT OPEN (fichier non ouvert). Le fichier spécifié dans une instruction CLOSE, CMD, PRINT#, INPUT# ou GET# doit d'abord être ouvert (OPEN).
FILE OPEN (fichier ouvert). Il s'agissait d'une tentative d'ouvrir un fichier en utilisant le numéro d'un fichier déjà ouvert.
FORMULA TOO COMPLEX (formule trop complexe). L'expression de la chaîne évaluée devrait être partagée au moins en deux parties pour que le système la traite.
ILLEGAL DIRECT (illégal en mode direct). L'instruction INPUT ne peut être utilisée qu'avec un programme et non dans le mode direct.
ILLEGAL QUANTITY (quantité illégale). Un numéro utilisé comme argument d'une fonction d'une instruction est en dehors des limites autorisées.

LOAD (chargement). Problème avec le programme sur bande.
NEXT WITHOUT FOR (NEXT sans FOR). C'est provoqué soit par des boucles incorrectement imbriquées, soit par un nom de variable dans une instruction NEXT qui ne correspond pas à celui d'une instruction FOR.
NOT INPUT FILE (fichier pas pour entrée). Il s'agit d'une tentative d'entrer (INPUT) ou obtenir (GET) des données d'un fichier qui a été spécifié pour être réservé à la sortie.
NOT OUTPUT FILE (fichier pas pour sortie). Il s'agit d'une tentative d'imprimer (PRINT) des données dans un fichier qui a été spécifié comme étant uniquement prévu pour l'entrée.
OUT OF DATA (données épuisées). Une instruction READ a été exécutée, mais il ne subsiste plus de données non lues dans une instruction DATA.
OUT OF MEMORY (mémoire pleine). Il n'y a plus de RAM disponible pour le programme ou les variables. Ceci peut également se présenter lorsque de trop nombreuses FOR ont été imbriqués ou lorsqu'il y a un trop grand nombre de GOSUB en présence.
OVERFLOW (débordement). Le résultat d'une comparaison est supérieur au nombre maximal autorisé qui est l. . .
REDIM'D ARRAY (tableau redimensionné). Un tableau ne peut être dimensionné qu'une fois. Si une variable de tableau est utilisée avant que le tableau soit dimensionné, une opération DIM (dimensionnement) automatique est réalisée sur ce tableau, fixant le nombre des éléments à dix, et tout dimensionnement (DIM) ultérieur provoque cette erreur.
REDO FROM START (recommencer depuis le début). Des données alphabétiques ont été frappées pendant une instruction INPUT, alors que des données numériques étaient attendus. Il suffit de frapper encore une fois l'entrée de telle manière qu'elle soit correcte et le programme continue de lui-même.
RETURN WITHOUT GOSUB (RETURN sans GOSUB). Une instruction RETURN a été rencontrée et aucune commande GOSUB n'a été sortie.
STRING TOO LONG (chaîne trop longue). Une chaîne peut contenir jusqu'à 255 caractères.
?SYNTAX ERROR (erreur de syntaxe). Une instruction n'est pas reconnaissable par le COMMODORE 64. Les parenthèses font défaut ou hors de parenthèses, mauvaise épellation d'un mot-clé.
TYPE MISMATCH (désaccord de frappe). Cette erreur se présente lorsqu'un nombre est utilisé à la place d'une chaîne de caractères ou vice versa.

UNDEF'D FUNCTION (fonction non définie). L'on s'est référé à une fonction définie par l'utilisateur, mais elle n'a jamais été définie en utilisant l'instruction DEF FN.

UNDEF'D STATEMENT (instruction non définie). Il s'agit d'une tentative d'utiliser GOT ou GOSUB ou RUN sur un numéro de ligne qui n'existe pas.

VERIFY (Vérification). Le programme sur la bande ou sur le disque ne correspond pas au programme actuellement en mémoire.

ANNEXE M

CARTE DES REGISTRES DES SYLPHES

Adresse de base VIC = 53248_{Dez} = $D000_{Hex}$

Registre # Dez	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
0	0	S0X7	S0X6	S0X5	S0X4	S0X3	S0X2	S0X1	S0X0	SYLPHE 0 X
1	1	S0Y7							S0Y0	SYLPHE 0 Y
2	2	S1X7							S1X0	SYLPHE 1 X
3	3	S1Y7							S1Y0	SYLPHE 1 Y
4	4	S2X7							S2X0	SYLPHE 2 X
5	5	S2Y7							S2Y0	SYLPHE 2 Y
6	6	S3X7							S3X0	SYLPHE 3 X
7	7	S3Y7							S3Y0	SYLPHE 3 Y
8	8	S4X7							S4X0	SYLPHE 4 X
9	9	S4Y7							S4Y0	SYLPHE 4 Y
10	A	S5X7							S5X0	SYLPHE 5 X
11	B	S5Y7							S5Y0	SYLPHE 5 Y
12	C	S6X7							S6X0	SYLPHE 6 X
13	D	S6Y7							S6Y0	SYLPHE 6 Y
14	E	S7X7							S7X0	SYLPHE 7 X
15	F	S7Y7							S7Y0	SYLPHE 7 Y
16	10	S7X8	S6X8	S5X8	S4X8	S3X8	S2X8	S1X8	S0X8	Octets maximales des valeurs X
17	11	RC8	EC5	BSM	BLNK	RSEL	YSCL2	YSCL1	YSCL0	
18	12	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	GRILLE
19	13	LPX7							LPX0	Stick lumineux X
20	14	LPY7							LPY0	Stick lumineux Y
21	15	SE7							SE0	Apparition Sylphe (On/Off)
22	16	N.C.	N.C.	RST	MCM	CSEL	XSCL2	XSCL1	XSCL0	
23	17	SEXY7							SEXY0	Dilatation Sylphe Y

Registre # Dez	Hex	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
24	18	VS13	VS12	VS11	CB13	CB12	CB11	CB10	N.C.	Mémoire des caractères de l'ECRAN
25	19	IRQ	N.C.	N.C.	N.C.	LPIRQ	ISSC	ISBC	RIRQ	Demande d' interruption
26	1A	N.C.	N.C.	N.C.	N.C.	MLPI	MISSC	MISBC	MRIRQ	MASQUES des demandes d'interruption
27	1B	BSP7							BSP0	PRIORITE fond/ sylphe
28	1C	SCM7							SCM0	SELECTION DE SYLPHE MULTICOLORE
29	1D	SEXX7							SEXX0	DILATATION SYLPHE X
30	1E	SSC7							SSC0	COLLISION sylphe-sylphe
31	1F	SBC7							SBC0	COLLISION sylphe-fond
		INFORMATION COULEUR								
32	20									Cadre
33	21									Fond 0
34	22									Fond 1
35	23									Fond 2
36	24									Fond 3
37	25						Sylphe Multicolore			SMC0
38	26									SMC1
39	27									Couleur Sylphe 0
40	28									Couleur Sylphe 1
41	29									Couleur Sylphe 2
42	2A									Couleur Sylphe 3
43	2B									Couleur Sylphe 4
44	26									Couleur Sylphe 5
45	2D									Couleur Sylphe 6
46	2E									Couleur Sylphe 7

L'information couleur se trouve dans le tableau à la page 139. En mode multicolore seulement les codes couleur 0 . . . 7 doivent être utilisés.

ANNEXE N

REGLAGES DE COMMANDE DE SON DU COMMODORE 64

Ce tableau pratique vous fournit les nombres-clé dont vous avez besoin dans vos programmes sonores, en fonction des troix voix du COMMODORE 64 que vous voulez utiliser. Pour introduire ou régler une commande sonore dans votre programme BASIC, il vous faut des commandes POKE (registre), (contenu) (voir exemple suivant). Chez les registres divisés vous devez additionner toutes les valeurs, p. ex. montée moyenne, descente moyenne de la voix 2.

POKE 54272+12,5*16+7 OU POKE 54284,87

Adresse de base — Registre — Frappe — Descente

Se rappeler que vous devez régler le volume avant de pouvoir générer le son. POKE 54296 suivi d'un nombre de 0 à 15 règle le volume des trois voix.

REGISTRES DE CONTRÔLE DE SONS

Adresse de base du SID: $54272_{Dez} = D400_{Hex}$

VOIX	REGISTRE 1	2	3	CONTENU				
0	7	14	FREQUENCE BASSE (0 ... 255)				
1	8	15	FRÉQUENCE HAUTE (0 ... 255)				
2	9	16	TAUX D'IMPULSIONS, BASSE FRÉQ. (0 ... 255)				
3	10	17	TAUX D'IMPULSIONS, HAUTE FRÉQ. (0 ... 15)				
4	11	18	Forme d'ondes	BRUIT 129	RECTANGLE 65	DENT DE SCIE 33	TRIANGLE 17
5	12	19	TAUX D'APPUI/RELÂCHEMENT 0*16 (dure) ... 15*16 (doux)		DESCENTE 0 (dure) ... 15*16 (doux)		
6	13	20	0*16 (muet) ... 15*16 (fort)		0 (rapide) ... 15 (lent)		
24	24	24	HAUTEUR: 0 (muet) ... 15 (hauteur maximale)				

EXEMPLE: Son de longue durée C-5 en voix 2, forme d'ondes: Triangle

```
SI=54272
POKE SI+24,15:POKE SI+7,207:POKE SI+8,34:POKE SI+13,240
```

(Hauteur) (Fréq. basse) (Fréq. haute) (maintenir fort 15*16)

Allumer le son: **POKE SI+11,17**
Eteindre le son: **POKE SI+11,0**

D'AUTRES REGISTRES DU SID

REGISTRE	CONTENU							
21	LIMITER FRÉQUENCE FILTRE, BASSE (0 ... 7)							
22	LIMITER FRÉQUENCE FILTRE, HAUTE (0 ... 255)							
23	RÉSONANCES				ALLUMER LE FILTRE			
	0 (aucune) ... 15*16 (forte)				externe	voix 3	voix 2	voix 1
					8	4	2	1
24	MODE FILTRE				HAUTEUR			
	voix 3	haut	Bande	bas				
	off	Passe	Passe	Passe				
	128	64	32	16	0 (muet) ... 15 (fort)			

En plus le SID possède 4 registres qui n'ont rien à voir avec les sons, mais à partir desquels le microprocesseur du COMMODORE 64 peut lire certaines informations.

REGISTRE	CONTENU
25	PADDLE X
26	PADDLE Y
27	OSCILLATEUR 3
28	COURBE ENVELOPPANTE 3

Pour lire p. ex. la position des paddles qui sont branchés aux ports de jeux, vous prenez les commandes:

```
SI=54272:X=PEEK(SI+25):Y=PEEK(SI+26)
```

Les variables X et Y contiennent des valeurs entre 0 et 255, dépendant de la position des paddles.

Dans les registres 27 et 28 la valeur actuelle de l'oscillateur et du générateur des courbes enveloppantes de la voix 3 peut être lue, pour créer p. ex. des générateurs par hasard ou afin d'influencer d'autres voix à l'aide de ces valeurs.

Essayez ces réglages pour simuler differents instruments

Instrument	Forme d'ondes		Montée/descente	Appui/relâchement	Taux d'impulsions
Piano	Impulsion	65	9	0	Haute-0, Basse-255
Flûte	Triangle	17	96	0	
Clavecin	Dent de scie	33	9	0	
Xylophone	Triangle	17	9	0	
Orgue	Triangle	17	0	240	
Accordéon	Triangle	17	102	0	
Trompette	Dent de scie	33	96	0	

REMARQUE: Les réglages de montée/descente et d'appui/relâchement doivent toujours être mis (POKE) dans votre programme **avant** que la forme d'ondes soit mise (POKE).

ANNEXE O

VALEUR DES NOTES DE MUSIQUE

Cette annexe contient une liste complète des numéros de notes, de la note effective et des valeurs à mettre (POKE) dans les registres HI FREQ et LOW FREQ de la puce sonore pour produire la note indiquée.

No.	Note octave	Fréquence (Hz)	Paramètre	octet-Hi	octet-Lo
0	C-0	16.4	278	1	22
1	C#-0	17.3	295	1	39
2	D-0	18.4	313	1	57
3	D#-0	19.4	331	1	75
4	E-0	20.6	351	1	95
5	F-0	21.8	372	1	116
6	F#-0	23.1	394	1	138
7	G-0	24.5	417	1	161
8	G#-0	26.0	442	1	186
9	A-0	27.5	468	1	212
10	A#-0	29.1	496	1	240
11	H-0	30.9	526	2	14
12	C-1	32.7	557	2	45
13	C#-1	34.6	590	2	78
14	D-1	36.7	625	2	113
15	D#-1	38.9	662	2	150
16	E-1	41.2	702	2	190
17	F-1	43.7	743	2	231
18	F#-1	46.2	788	3	20
19	G-1	49.0	834	3	66
20	G#-1	51.9	884	3	116
21	A-1	55.0	937	3	169
22	A#-1	58.3	992	3	224
23	H-1	61.7	1051	4	27
24	C-2	65.4	1114	4	90
25	C#-2	69.3	1180	4	156
26	D-2	73.4	1250	4	226
27	D#-2	77.8	1325	5	45
28	E-2	82.4	1403	5	123
29	F-2	87.3	1487	5	207
30	F#-2	92.5	1575	6	39
31	G-2	98.0	1669	6	133
32	G#-2	103.8	1768	6	232
33	A-2	110.0	1873	7	81
34	A#-2	116.5	1985	7	193
35	H-2	123.5	2103	8	55
36	C-3	130.8	2228	8	180
37	C#-3	138.6	2360	9	56
38	D-3	146.8	2500	9	196
39	D#-3	155.6	2649	10	89
40	E-3	164.8	2807	10	247
41	F-3	174.6	2974	11	158
42	F#-3	185.0	3150	12	78
43	G-3	196.0	3338	13	10
44	G#-3	207.7	3536	13	208
45	A-3	220.0	3746	14	162
46	A#-3	233.1	3969	15	129
47	H-3	246.9	4205	16	109
48	C-4	261.6	4455	17	103
49	C#-4	277.2	4720	18	112

No.	Note octave	Fréquence (Hz)	Paramètre	octet-Hi	octet-Lo
50	D-4	293.7	5001	19	137
51	D#-4	311.1	5298	20	178
52	E-4	329.6	5613	21	237
53	F-4	349.2	5947	23	59
54	F#-4	370.0	6301	24	157
55	G-4	392.0	6676	26	20
56	G#-4	415.3	7072	27	160
57	A-4	440.0	7493	29	69
58	A#-4	466.2	7939	31	3
59	H-4	493.9	8411	32	219
60	C-5	523.3	8911	34	207
61	C#-5	554.4	9441	36	225
62	D-5	587.3	10002	39	18
63	D#-5	622.3	10597	41	101
64	E-5	659.3	11227	43	219
65	F-5	698.5	11894	46	118
66	F#-5	740.0	12602	49	58
67	G-5	784.0	13351	52	39
68	G#-5	830.6	14145	55	65
69	A-5	880.0	14986	58	138
70	A#-5	932.3	15877	62	5
71	H-5	987.8	16821	65	181
72	C-6	1046.5	17821	69	157
73	C#-6	1108.7	18881	73	193
74	D-6	1174.7	20004	78	36
75	D#-6	1244.5	21193	82	201
76	E-6	1318.5	22454	87	182
77	F-6	1396.9	23789	92	237
78	F#-6	1480.0	25203	98	115
79	G-6	1568.0	26702	104	78
80	G#-6	1661.2	28290	110	130
81	A-6	1760.0	29972	117	20
82	A#-6	1864.7	31754	124	10
83	H-6	1975.5	33642	131	106
84	C-7	2093.0	35643	139	59
85	C#-7	2217.5	37762	147	130
86	D-7	2349.3	40008	156	72
87	D#-7	2489.0	42387	165	147
88	E-7	2637.0	44907	175	107
89	F-7	2793.8	47578	185	218
90	F#-7	2960.0	50407	196	231
91	G-7	3136.0	53404	208	156
92	G#-7	3322.4	56580	221	4
93	A-7	3520.0	59944	234	40
94	A#-7	3729.3	63508	248	20

Les valeurs de ce tableau ne sont pas impératives! Si vous utilisez plusieures voix, vous ferez bien de „désaccorder“ un peu la deuxième et troisième voix, c'est-à-dire de changer l'octet-Lo du tableau légèrement (!). Vous obtiendrez une meilleure sonorité.

ANNEXE P

Topogramme du Commodore 64

(★ adresses utiles)

	Hex	Décimal	Description
	0000	0	6510 Registre d'acheminement des données
	0001	1	6510 Registre de sortie
	0002	2	non utilisé
	0003 – 0004	3 – 4	Vecteur de conversion flottant-fixe
	0005 – 0006	5 – 6	Vecteur de conversion fixe – flottant
	0007	7	Signe de recherche
	0008	8	Indicateur du mode „guillemets"
	0009	9	Compteur de colonne tabulaire
	000A	10	0 = LOAD, 1 = VERIFY
	000B	11	Repère du tampon d'entrée de données/nombre d'éléments
	000C	12	Indicateur du format standard (DIM)
	000D	13	Typ: FF = chaîne, 00 = numérique
	000E	14	Type: 80 = nombre entier, 00 = virgule flottante
	000F	15	Indicateur DATA/LIST
	0010	16	Elément/Indicateur FNx
	0011	17	00 = INPUT, 40 = GET, 98 = READ
	0012	18	Préfixe ATN
	0013	19	Indicateur d'égalité pour la comparaison avec l'actuel appareil d'entrée/sortie
★	0014 – 0015	20 – 21	Valeur entière
	0016	22	Repère sur piles enchaînées temporaires
	0017 – 0018	23 – 24	Dernier vecteur de chaîne temporaire
	0019 – 0021	25 – 33	Pile de chaînes temporaires
	0022 – 0025	34 – 37	Secteur pour repère auxiliaire
	0026 – 002A	38 – 42	Secteur pour le résultat d'une multiplication
★	002B – 002C	43 – 44	Repère sur le lancement du BASIC
★	002D – 002E	45 – 46	Repère sur le début d'une variable
★	002F – 0030	47 – 48	Repère sur le début des zones
★	0031 – 0032	49 – 50	Repère sur la fin des zones

★	0033 – 0034	51 – 52	Repère sur la mémoire de chaînes (mouvement vers le bas)
	0035 – 0036	53 – 54	Repère auxiliaire pour chaînes
★	0037 – 0038	55 – 56	Repère sur la limite de la mémoire
	0039 – 003A	57 – 58	Numéro de l'actuelle ligne BASIC
	003B – 003C	59 – 60	Numéro de la ligne BASIC précédente
	003D – 003E	61 – 62	Repère sur l'instruction BASIC pour CONT
	003F – 0040	63 – 64	Numéro de l'actuelle ligne DATA
	0041 – 0042	65 – 66	Adresse de l'actuel élément DATA
★	0043 – 0044	67 – 68	Vecteur saut pour entrée
	0045 – 0046	69 – 70	Actuel nom de variable
	0047 – 0048	71 – 72	Adresse de l'actuelle variable
	0049 – 004A	73 – 74	Repère variable pour FOR . . . NEXT
	004B – 004C	75 – 76	Mémoire intermédiaire pour repère BASIC
	004D	77	Accumulateur pour symboles comparatifs
	004E – 0053	78 – 83	Secteur de travail ayant diverses utilisations (repère etc.)
	0054 – 0056	84 – 86	Vecteur saut pour fonctions
	0057 – 0060	87 – 96	Secteur ayant diverses utilisations pour opérations numériques
★	0061	97	Accumulateur virgule flottante #1 (FAC): exponent
★	0062 – 0065	98 – 101	Accumulateur virgule flottante #1 (FAC): mantisse
★	0066	102	Accumulateur virgule flottante #1 (FAC): préfixe
	0067	103	Repère pour interprétation polynôme
	0069 – 006E	105 – 110	Accumulateur virgule flottante #2 exposant etc.
	006F	111	Comparaison des préfixes accu #1 / accu #2
	0070	112	Position d'une basse valeur accu #1 (d'arrondi)
	0071 – 0072	113 – 114	Longueur du tampon-cassette
★	0073 – 008A	115 – 138	CHRGET sous-programme (va chercher un caractère)
	007A – 007B	122 – 123	Repère BASIC à l'intérieur du sous-programme

	008B – 008F	139 – 143	Valeur de lancement du générateur aléatoire
★	0090	144	Octet d'état ST
	0091	145	Indicateur pour touches STOP et RVS
	0092	146	Constante de temps pour cassette
	0093	147	0 = LOAD, 1 = VERIFY
	0094	148	Sortie sérielle: indicateur pour caractère repositionné
	0095	149	Caractère repositionné
	0096	150	Recevoir EOT de la cassette (cassette Sync#)
	0097	151	Mémoire pour registres
★	0098	152	Nombre de fichiers ouverts
★	0099	153	Appareil d'entrée (normal = 0)
★	009A	154	Appareil de sortie (CMD) (normal = 3)
	009B	155	Octet paritaire de la bande
	009C	156	Recevoir indicateur pour octet
	009D	157	Contrôle de sortie (80 = direct, 00 = RUN)
	009E	158	Erreur de la bande/tampon-caractères
	009F	159	Erreur de la bande corrigée
★	00A0 – 00A2	160 – 162	Horloge interne (HML)
	00A3	163	Compteur de bits sériel indicateur pour EOI
	00A4	164	Compteur de cycles
	00A5	165	Compteur descendant lors de l'écriture sur cassette
	00A6	166	Repère tampon-cassette
	00A7 – 00AB	167 – 171	Indicateurs pour l'écriture et la lecture de cassettes
	00AC – 00AD	172 – 173	Repère de lancement du programme
	00AE – 00AF	174 – 175	Repère de fin du programme
	00B0 – 00B1	176 – 177	Constante de temps pour bande
★	00B2 – 00B3	178 – 179	Repère du début du tampon-cassette
	00B4	180	Timer bande (1 = positionné), compteur de bits
	00B5	181	Bande EOT/RS 232 prochain bit à émettre
	00B6	182	★★★

★	00B7	183	Nombre de caractères dans le nom du fichier
★	00B8	184	Numéro actuel logique du fichier
★	00B9	185	Adresse actuelle secondaire
★	00BA	186	Appareil actuel
★	00BB – 00BC	187 – 188	Repère sur le nom du fichier
	00BD	189	★★★
	00BE	190	Nombre pour la lecture/écriture de blocs restants
	00BF	191	Tampon-mots sériel
	00C0	192	Indicateur moteur-cassette
	00C1 – 00C2	193 – 194	Adresse de lancement Entrée/Sortie
	00C3 – 00C4	195 – 196	Repère sur adresses vectorielles KERNAL
★	00C6	198	Nombre de caractères dans le tampon-clavier
★	00C7	199	Indicateur RVS pour écran
	00C8	200	Repère sur la fin de la ligne pour entrée
	00C9 – 00CA	201 – 202	Position du curseur d'entrée (ligne, colonne)
★	00CB	203	Touche appuyée (64=aucune touche)
	00CC	204	Curseur allumé/éteint (0=curseur clignote)
	00CD	205	Compteur pour curseur clignotant
	00CE	206	Caractère sur position du curseur
	00CF	207	Curseur dans la phase clignotante
	00D0	208	Entrée écran/clavier
★	00D1 – 00D2	209 – 210	Repère sur la ligne de l'écran
★	00D3	211	Repère sur la colonne de l'écran
	00D4	212	0=curseur direct, sinon programmé (QUOTE MODE)
★	00D5	213	Longueur de l'actuelle ligne de l' écran (40/80)
★	00D6	214	Ligne dans laquelle se trouve le curseur
	00D7	215	Dernière touche/total de contrôle/ tampon
★	00D8	216	Nombre d'insertions non entrées
★	00D9 – 00F0	217 – 240	Table d'enchaînement lignes d'écran
	00F1	241	Ligne d'écran factice

	00F2	242	Marque de lignes d'écran
★	00F3 – 00F4	242 – 244	Repère sur la couleur d'écran
	00F5 – 00F6	245 – 246	Repère sur table de décodage clavier
	00F7 – 00F8	247 – 248	RS232 Repère de réception
	00F9 – 00FA	249 – 250	RS232 Repère de report
★	00FB – 00FE	251 – 254	Espace vide à la page 0 pour système d'exploitation
	00FF	255	Mémoire BASIC
	0100 – 010A	256 – 266	Secteur de travail pour la conversion de la virgule flottante en ASCII
	0100 – 013E	256 – 318	Erreur de bande
	0100 – 01FF	256 – 511	Secteur de la pile du processeur
★	0200 – 0258	512 – 600	Tampon d'entrée BASIC
★	0259 – 0262	601 – 610	Table des fichiers logiques
★	0263 – 026C	611 – 620	Table des numéros d'appareil
★	026D – 0276	621 – 630	Table des adresses secondaires
★	0277 – 0280	631 – 640	Tampon-clavier
★	0281 – 0282	641 – 642	Adresse de lancement de la RAM pour le système d'exploitation
★	0283 – 0284	642 – 644	Fin de la RAM pour le système d'exploitation
	0285	645	Indicateur pour dépassement du temps sur bus sériel
★	0286	646	L'actuel code de couleur
	0287	647	Couleur sous curseur
★	0288	648	Adresse de mémoire-écran (page)
★	0289	649	Volume maximal du tampon-clavier
★	028A	650	Répétition des touches (128=toutes les touches)
★	028B	651	Compteur de vitesse de répétition
	028C	652	Compteur de retards de répétition
★	028D	653	Indicateur pour SHIFT/CNTRL
	028E	654	Dernière référence SHIFT du clavier
	028F – 0290	655 – 656	Repère sur table de décodage clavier
★	0291	657	Mode SHIFT (0=positionné, 128=bloqué)
	0292	658	Déroulement automatique descendant (0=allumé, #0=éteint)
	0293	659	RS232 Registre de contrôle
	0294	660	RS232 Registre de commande

	0285 – 0296	661 – 662	non standard (temps bit)
			★★★
	0297	663	RS232 Registre d'état
	0298	664	Nombre de bits à émettre
	0299 – 029A	665 – 666	Baud rate
	029B	667	RS232 Repère de réception — recevoir
	029C	668	RS232 Repère d'entrée
	029D	669	RS232 Repère de report — émettre
	029E	670	RS232 Repère de sortie
	029F – 02A0	271 – 672	Contient vecteur IRQ pendant exploitation par cassette
	02A1 – 02FF	673 – 767	★★★
★	0300 – 0301	768 – 769	Vecteur pour messages d'erreurs
	0302 – 0303	770 – 771	Vecteur pour lancement à chaud BASIC
	0304 – 0305	772 – 773	Conversion de mots-clés en signes
	0306 – 0307	774 – 775	Conversion de signes en mots-clés
	0308 – 0309	776 – 777	Exécuter nouvelle commande BASIC
	030A – 030B	778 – 779	Aller chercher élément arithmétique
	030C	780	Mémoire pour 6502 registre ,,A
	030D	781	Mémoire pour 6502 registre ,,X
	030E	782	Mémoire pour 6502 registre ,,Y
	030F	783	Mémoire pour 6502 registre ,,P
	0310 – 0313	784 – 787	Saut USR
	0314 – 0315	788 – 789	Arrêt-machine (IRQ) (EA31)
	0316 – 0317	790 – 791	Interruption (FE66) (break interrupt)
	0318 – 0319	792 – 793	Interruption sans masquage (NMI) (FE47)
	031A – 031B	794 – 795	OPEN (F40A) (F34A)
	031C – 031D	796 – 797	CLOSE (F291)
	031E – 031F	798 – 799	Canal d'entrée (F2C7) (F209)
	0320 – 0321	800 – 801	Canal de sortie (F250)
	0322 – 0323	802 – 803	Reconstitution de l'éntrée/sortie (Effacement de tous les canaux ouverts (F333)
	0324 – 0325	804 – 805	INPUT (F157)
	0326 – 0327	806 – 807	OUTPUT (FICA)
	0328 – 0329	808 – 809	Vérifier touche STOP (F770) (F6ED)

032A – 032B	810 – 811	GET (F13E)
032C – 032D	812 – 813	Fermer tous les canaux (F32F)
032E – 032F	814 – 815	IRQ-utilisateur (FE66)
0330 – 0331	816 – 817	Charger RAM (F4A5)
0332 – 0333	818 – 819	Mémoriser RAM (F5ED)
0334 – 033B	820 – 827	★★★
033C – 03FB	828 – 1019	Tampon-cassette
0400 – 07FF	1024 – 2047	Mémoire d'écran 1 K
(0400 – 07E7	1024 – 2023	Matrice (vidéo)
(07F8 – 07FF	2040 – 2047	Repère pour sprites (lutins)
0800 – 9FFF	2048 – 40959	Mémoire d'utilisateur BASIC
A000 – BFFF	40960 – 49151	ROM 3 K BASIC
C000 – CFFF	49152 – 53247	RAM 4 K

INDEX

M

N

O

P

R

S

T

U

V

W

X

CARTE DE REFERENCE RAPIDE COMMODORE 64

VARIABLES SIMPLES

Type	*Nom*	*Limites*
Réel	XY	±1.70141183E+38 ±2.93873588E−39
Nombre entier	XY%	±32767
Chaîne	XY$	de 0 à 255 caractères

X est une lettre (A-Z), Y est une lettre ou un chiffre (0 - 9). Les noms des variables peuvent comporter plus de 2 caractères, mais les deux premiers sont identifiés.

VARIABLES DES TABLEAUX

Type	*Nom*
A une entrée	XY(5)
A deux entrées	XY(5,5)
A trois entrées	XY(5,5,5)

Les tableaux jusqu'à 11 éléments (indice 0 - 10) peuvent être utilisés si nécessaire. Les tableaux comportant plus de onze éléments doivent être DIMensionnés.

DIM A(X,Y,Z)	Définit les indices maximum pour A: Réserve de l'espace pour (X+1)*(Y+1)*(Z+1) Eléments démarrant sur A (0,0,0)

OPÉRATEURS ALGÉBRIQUES

=	Assigne une valeur à une variable
–	Négation ou soustraction
↑	Elévation à une puissance
*	Multiplication
/	Division
+	Addition

OPÉRATEURS LOGIQUES

=	Egal
<>	Différent de
<	Inférieur à
>	Supérieur à
< =	Inférieur ou égal à
> =	Supérieur ou égal à
NOT	Logique «PAS»
AND	Logique «ET»
OR	Logique «OU» (pas exclu)

COMMANDES DU SYSTÈME

LOAD «NOM»	Charge un programme de la bande
SAVE «NOM»	Charge un programme sur la bande
LOAD «NOM»,8	Charge un programme du disque
SAVE «NOM»,8	Sauvegarde un programme sur disque
VERIFY «NOM»	Vérifie que le programme a été sauvegardé correctement
RUN	Exécute un programme
RUN xxx	Exécute un programme en partant de la ligne xxx
STOP	Arrête l'exécution
END	Achève l'exécution
CONT	Poursuite de l'exécution du programme à partir de la ligne où le programme a été arrêté
POKE X,Y	Modifie les contenus de l'emplacement X à la valeur Y
SYS xxxxx	Saute à l'exécution d'un programme en langage machine en commençant par xxxxx
WAIT X,Y,Z	Le programme attend jusqu'à ce que les contenus de l'emplacement X soient différents de zéro, lorsque EOR vient avec Y et AND avec Z

FONCTIONS DU SYSTÈME

PEEK(X)	Fournit les contenus de l'emplacement de la mémoire X
USR(X)	Passe la valeur de X à une sous-routine en langage machine
RND(X)	chiffre aléatoire
FRE(X)	indique la mémoire libre

COMMANDES D'ÉDITION ET DE FORMATAGE

LIST	Sort la liste d'un programme complet
LIST A-B	Listes de la ligne A à la ligne B
REM Commentaire	Le commentaire du message peut être listé, mais
CLR/HOME	Positionne le curseur en haut à gauche de l'écran
SHIFT CLR/HOME	Efface l'écran
SHIFT INST/DEL	Insère un espace devant l'actuelle position du curseur.
INST/DEL	Efface le caractère devant la position du curseur.
CTRL	Lorsqu'elle est utilisée avec une touche numérique, sélectionne la couleur. Peut être utilisée dans l'instruction PRINT.
CRSR touches	Déplacent le curseur dans la direction indiquée sur l'écran.
Touche COMMODORE	Lorsqu'elle est utilisée avec SHIFT, sélectionne le mode minuscule et le mode d'affichage graphique. Lorsqu'elle est utilisée avec une touche numérique, sélectionne la couleur.

FONCTIONS DE FORMATAGE

TAB(X)	Espace les positions X sur l'écran
SPC(X)	X espaces
POS(X)	Position du prochain espace sur la ligne

TABLEAUX ET CHAÎNES

LEN (X$)	Fournit le nombre de caractères dans X$
STR$(X)	Fournit la valeur numérique de X convertit en une chaîne
VAL(X$)	Fournit la valeur numérique de X$ jusqu'au premier caratère non numérique
CHR$(X)	Fournit le caractère du code ASCII X
ASC(X$)	Fournit le code ASCII pour le premier caractère de X$
LEFT$(A$,X)	Fournit les premiers caractères X$ de A$
RIGHT$(A$,X)	Fournit les derniers caractères X$ de A$
MID$(A$,X,Y)	Fournit les caractères Y de A$ en commençant par le caractère X

COMMANDES D'ENTRÉE/SORTIE

INPUT A$ ou A	Affiche «?» sur l'écran et attend que l'utilisateur entre une chaîne ou une valeur
INPUT «ABC»;A	Affiche un message et attend que l'utilisateur entre une valeur.
GET A$ ou A	Attend que l'utilisateur frappe une valeur de caractère; inutile de frapper RETURN
DATA A, «B», C	Initialise un jeu de valeurs qui peut être utilisé par l'instruction READ
READ A$ ou A	Assigne la valeur DATA suivante à A$ ou A
RESTORE	Réinitialise le pointeur de données pour redémarrer la lecture de la liste de données
PRINT «A=»; A	Affiche 'A=' et valeur de A.';' supprime les espaces, ',' tabule les données sur la zone suivante
OPEN 1,4	ouvre le fichier 1 pour l'imprimante
INPUT #2, A$	lit la variable A$ du fichier 2
GET #2, A$	lit un signe dans la variable A$ du fichier 2
PRINT #1, A	Imprime A sur l'imprimante
CLOSE 1	ferme le fichier 1

AVANCEMENT DU PROGRAMME

GOTO X	Branche sur la ligne X
IF A = 3 THEN 10	Si l'insertion IF est confirmée, THEN (alors) réalise l'exécution de la partie suivante de l'instruction. Si IF n'est pas confirmé on passe à l'exécution du numéro de ligne suivant.
FOR A = 1 TO 10	Exécute deux instructions entre FOR et l'instruction
STEP 2: NEXT	NEXT correspondante avec A passant de 1 à 10, par pas de 2. Sauf spécification contraire, le pas est 1.
NEXT A	Définit une fin de boucle. A est optionnel.
GOSUB 2000	Branche la sous-routine en commençant sur la ligne 2000
RETURN	Marque de fin de sous-routine. Retourne à l'instruction suivant l'instruction GOSUB la plus récente.
ON X GOT A,B,C	Se branche sur le Xème numéro de ligne de la liste. Si X = 1, se branche sur A etc.
ON X GOSUB A,B,C	Se branche sur la sous-routine du Xème numéro de ligne de la liste.

Commodore Business
Machines Limited
3370 Pharmacy Avenue
Agincourt
Ontario M1W 2K4

Commodore AG
Aeschenvorstadt 57B
4010 Bâle

Commodore SPRL
Av. des Becassines 30
1160 Bruxelles