AMIGA® HARDWARE REFERENCE MANUAL

REVISED & UPDATED

COMMODORE-AMIGA, INCORPORATED

TABLE DES MATIERES

Chapitre 1 INTRODUCTION
Les Composants de l' Amiga 2
LE MC68000 ET LES PUCES SPECIALISEES DE L'AMIGA 2
INTERFACE DIRECTE POUR CAMERA VIDEO ET MAGNETOSCOPE.....5
PERIPHERIQUES.....5
EXTENSION ET POSSIBILITES D'ADAPTATION DU SYSTEME.....6
Au Sujet des Exemples.....7
Quelques Mises en Garde destinées aux Programmeurs à Niveau 'Hardware'...9

Chapitre 2 LE 'HARDWARE' DU CO-PROCESSEUR.....13
Introduction.....13
AU SUJET DE CE CHAPITRE.....14
Définition des Instruction du Copper.....14
L'Instruction MOVE.....15
L'Instruction WAIT.....17
POSITION HORIZONTALE DU FAISCEAU.....18
POSITION VERTICALE DU FAISCEAU.....18
LES BITS DE VALIDATION DE COMPARAISON.....19
Utilisation des registres du Copper.....20
REGISTRES D'ADRESSE.....20
ADRESSE 'STROBE' DE SAUT.....21
LE REGISTRE DE CONTROLE.....21
Réalisation d'une Liste d'Instructions du Copper.....22
EXEMPLE DE LISTE COPPER COMPLETE.....24
BOUCLES ET BRANCHEMENTS.....25
Démarrage et Arrêt du Copper.....25
DEMARRAGE DU COPPER APRES UNE REINITIALISATION 25
ARRET DU COPPER.....26
Sujets Avancés.....27
L'INSTRUCTION SKIP.....27
BOUCLES ET BRANCHEMENTS DU COPPER ET VALIDATION D'UNE COMPARAISON....28
UTILISATION DU COPPER EN MODE ENTRELACE..... 30
UTILISATION DU COPPER AVEC LE 'BLITTER'.....31
LE COPPER ET LE 68000.....31
Récapitulation des Instructions du Copper.....32

Chapitre 3 LE 'HARDWARE' DES AIRES DE JEU.....33
Introduction.....33
AU SUJET DE CE CHAPITRE.....34
CARACTERISTIQUES DES AIRES DE JEU.....34
Construction d'une Aire de Jeu Simple.....38
HAUTEUR ET LARGEUR D'UNE AIRE DE JEU.....39
PLANS DE BITS ET COULEUR.....39
SELECTION DU NOMBRE DE PLANS DE BITS.....43
RESERVATION DE MEMOIRE POUR DES PLANS DE BITS.....46
CODAGE DES PLANS DE BITS POUR OBTENIR UN BON COLORIAGE.....49
DEFINITION DES DIMENSIONS DE LA FENETRE D'AFFICHAGE.....50
INFORMER LE SYSTEME SUR LA MANIERE DE PRELEVER ET D'AFFICHER LES DONNEES.....53
AFFICHAGE ET RE-AFFICHAGE D'UNE AIRE DE JEU.....56
VALIDATION DE L'AFFICHAGE DES COULEURS.....56
RECAPITULATION SUR L'AIRE DE JEU SIMPLE.....57
EXEMPLES DE CONSTRUCTION D'AIRES DE JEU SIMPLES.....59
Création d'Affichage d'une Double Aire de Jeu.....62
Assignation d'un Plan de Bits en Mode Double Aire de Jeu.....62
REGISTRES DE COULEUR EN MODE DOUBLE AIRE DE JEU.....65
CONTROLE ET PRIORITE D'UNE DOUBLE AIRE DE JEU.....66
ACTIVATION DU MODE DOUBLE AIRE DE JEU.....67
RECAPITULATION SUR LES AIRES DE JEU DOUBLES.....67
Plans de Bits et Fenêtres d'Affichage de Toutes les Dimensions.....68
QUAND LA GRANDE IMAGE A UNE TAILLE PLUS GRANDE QUE LA FENETRE D'AFFICHAGE.....68
DIMENSIONS MAXIMALES DE LA FENETRE D'AFFICHAGE.....74
Déplacement (Déroulement) des Aires de Jeu.....75
DEROULEMENT VERTICAL.....75
DEROULEMENT HORIZONTAL.....77
RECAPITULATION SUR LES AIRES DE JEU QU'ON VEUT FAIRE DEFILER.....80
Sujets Avancés.....81
INTERACTIONS ENTRE LES AIRES DE JEU ET LES AUTRES OBJETS.....81
MODE 'HOLD-AND-MODIFY'.....81
CREATION D'UN AFFICHAGE AVEC PLUSIEURS AIRES DE JEU DIFFERENTES.....84
UTILISATION D'UNE SOURCE VIDEO EXTERNE.....84
RECAPITULATION DES REGISTRES DES AIRES DE JEU.....84
Récapitulation sur la Sélection des Couleurs.....87
CONTENU DES REGISTRES DE COULEUR.....87
UNE SELECTION DU CONTENU DES REGISTRES DE COULEUR.....88
SELECTION DE COULEUR EN BASSE RESOLUTION.....88
SELECTION DE COULEUR EN MODE 'HOLD-AND-MODIFY'.....90
SELECTION DE COULEUR EN HAUTE RESOLUTION..... 90

Chapitre 4 LE 'HARDWARE' DES SPRITES.....93
Introduction.....93
AU SUJET DE CE CHAPITRE.....94
Création d'un Sprite.....94

POSITION A L'ECRAN.....94
DIMENSIONS DES SPRITES.....97
FORME DES SPRITES.....97
COULEUR DES SPRITES.....98
CONCEPTION D'UN SPRITE.....101
CONSTRUCTION DE LA STRUCTURE DE DONNEES.....101
Affichage d'un Sprite.....106
SELECTION D'UN CANAL DMA ET MISE EN PLACE DES POINTEURS.....107
REINITIALISATION DES POINTEURS D'ADRESSE.....107
EXEMPLE D'AFFICHAGE D'UN SPRITE.....108
Déplacement d'un Sprite.....110
Création de Sprites Supplémentaires.....111
PRIORITE DES SPRITES.....112
Réemploi des Canaux DMA des Sprites.....113
Sprites Recouverts.....115
Sprites Rattachés.....117
Mode Manuel.....120
Particularités du 'Hardware' des Sprites.....121
Récapitulation des Registres des Sprites.....124
POINTEURS.....124
REGISTRES DE CONTROLE.....124
REGISTRES DE DONNEES.....126
Récapitulation des Registres de Couleur des Sprites.....126
INTERACTION ENTRE SPRITES ET AUTRES OBJETS.....128

Chapitre 5 LE 'HARDWARE' AUDIO.....129
Introduction.....129
INTRODUCING A LA GENERATION DES SONS.....130
LE 'HARDWARE' SON DE L'AMIGA.....133
Création et Ecoute d'un Son.....134
CHOIX DU CANAL A UTILISER.....134
CREATION DES DONNEES DE LA FORME D'ONDE.....134
INFORMER LE SYSTEME SUR LES DONNEES.....136
SELECTION DU VOLUME.....136
SELECTION DU TAUX DE SORTIE DES DONNEES.....137
ENTENDRE LA FORME D'ONDE.....140
ARRET DU DMA AUDIO.....141
RECAPITULATION.....142
EXEMPLE.....142
Production de Sons Complexes.....143
RACCORDER LES SONS.....143
ENTENDRE PLUSIEURS SONS EN MEME TEMPS.....145
MODULATION DU SON.....145
Production de Sons de Bonne Qualité.....148
REALISATION DES TRANSITIONS D'UNE FORME D'ONDE.....148
TAUX D'ECHANTILLONNAGE.....148
RENDEMENT.....149
REDUCTION DU BRUIT.....150

LA DISTORSION.....150
FILTRE PASSE-BAS.....152
Utilisation Directe de la Sortie Audio (sans DMA).....153
La Gamme Musicale Tempérée.....154
Valeurs en Décibels des Intevalles de Volume.....159
Les Etats Audio Machine.....160

Chapitre 6 LE 'HARDWARE' DU 'BLITTER'.....163
Introduction.....163
Organisatio de la Mémoire.....164
Les Canaux DMA.....164
Generateur de Fonctions.....168
ETUDE DE L'OCTET DE CONTROLE LF AVEC LES 'MINTERMS'.....169
ETUDE DE L'OCTET DE CONTROLE LF AVEC LES DIAGRAMMES DE VENN.....172
Décalages et Masques.....173
Adressage par Decrémentation.....176
Copie de Sections Arbitraires.....177
Mode Remplissage de Surfaces.....178
Le Drapeau 'Done' du 'Blitter'.....180
LE MULTITACHE ET LE 'BLITTER'.....181
Drapeau d'Interruption.....181
Le Drapeau Zéro.....182
Le Registre Pipeline.....182
Mode Ligne.....184
RECAPITULATION DU REGISTRE EN MODE LIGNE.....186
Vitesse d'Exécution du 'Blitter'.....188
Opérations du 'Blitter' et DMA du Système.....189
Schéma Fonctionnel par Blocs du 'Blitter'.....193
Les points clé du 'Blitter'.....195
EXEMPLE : ClearMem.....195
EXEMPLE : Ligne Simple.....197
EXEMPLE : Rotation de Bits.....199

Chapitre 7 LE 'HARDWARE' DE CONTROLE DU SYSTEME.....201
Introduction.....201
Les priorités Vidéo.....202
LES PRIORITES DES SPRITES DEFINIES A L'AVANCE.....202
COMMENT SONT REGROUPES LES SPRITES.....203
COMPRENDRE LES PRIORITES VIDEO.....203
MISE EN PLACE DU REGISTRE DE CONTROLE DES PRIORITES.....204
Détection d'une Collision.....207
COMMENT AVOIR CONNAISSANCE D'UNE COLLISION.....207
COMMENT INTERPRETER LES DONNEES D'UNE COLLISION.....208
COMMENT EST CONTROLEE LA DETECTION D'UNE COLLISION.....209
Détection de la Position du Faisceau.....210
UTILISATION DU COMPTEUR DE POSITION DU FAISCEAU.....210
Les Interruptions 211
INTERRUPTIONS NON MASQUABLES.....212

INTERRUPTIONS MASQUABLES.....212
INTERFACE UTILISATEUR VERS LE SYSTEME D'INTERRUPTIONS.....212
REGISTRES DE CONTROLE DES INTERRUPTIONS.....212
METTRE A 1 ET A 0 LES BITS.....213
Contrôle des DMA.....217
Accès du Processeur à la Mémoire 'Chip'.....217
Réinitialisation et Opération de Mise sous Tension Rapide.....219

Chapitre 8 LE 'HARDWARE' DES INTERFACES.....221
Introduction.....221
Interface du Port Contrôleur.....222
REGISTRES UTILISES AVEC LE PORT DU CONTROLEUR.....223
Controleur du Lecteur de Disquettes.....,.....235
REGISTRES UTILISES PAR LE SOUS-SYSTEME DU LECTEUR DISQUETTES.....236
LES INTERRUPTIONS DU LECTEUR DE DISQUETTES.....244
Le Clavier.....245
COMMENT ON RECEPTIONNE LES DONNEES DE CLAVIER.....245
LE TYPE DE DONNEES QU'ON RECOIT.....245
LIMITATIONS DU CLAVIER.....247
Interface d'Entrée/Sortie Parallèle.....250
Interface Série.....250
INTRODUCTION A L'ENSEMBLE DES CIRCUITS SERIE.....250
MISE EN PLACE DE LA VITESSE DE TRANSMISSION DE L'INFORMATION.....250
MISE EN PLACE DU MODE RECEPTION.....251
CONTENU DU REGISTRE DE RECEPTION DES DONNEES.....251
COMMENT SONT TRANSMISES LES DONNEES EN SORTIE.....253
SPECIFICATION DU CONTENU DU REGISTRE.....254
Connexions de Sortie pour l'Affichage.....255

Appendice A Récapitulation des Registres - Ordre Alphabétique.....257

Appendice B Récapitulation des Registres par Ordre d'Adresse.....281

Appendice C Liste d'Implantation des Broches des Puces Spécialisées.....289

Appendice D Mappe Mémoire du Système.....293

Appendice E Les Interfaces.....295

Appendice F Adaptateurs d'Interface Complexes.....317
Puces 8520 : Adaptateurs d'Interface Complexes (CIA).....317
Mappe des Registres de la Puce 319
Description Fonctionnelle des Registres.....320
PORTS D'E/S (PRA, PRB, DDRA, DDRB).....320
TRANSFERT DE DONNEES.....320
METRONOMES (MINUTERIE A, MINUTERIE B).....320
MODES D'ENTREE.....322
NOM DES BITS du Registre de LECTURE.....322

NOM DES BITS du Registre d'ECRITURE.....322
Horloge Temps Réel.....323
NOM DES BITS d'écriture TEMPS REEL/ALARME ou de lecture TEMPS REEL.....323
Registre à Décalage Série (SDR).....324
MODE ENTREE [INPUT MODE].....324
MODE SORTIE [OUTPUT MODE].....324
LE COTE BI-DIRECTIONNEL.....325
Registre de Contrôle d'Interruption (ICR).....325
LECTURE DU REGISTRE DE CONTROLE D'INTERRUPTION.....326
ECRITURE DU MASQUE DE CONTROLE D'INTERRUPTION.....326
Registres de Contrôle.....327
REGISTRE DE CONTROLE A.....327
MAPPE DE BITS DU REGISTRE CRA.....328
MAPPE DE BITS DU REGISTRE CRB.....329
Assignation des Signaux au Port.....329
Détail sur les Connexions 'Hardware'.....332
SIGNAUX D'INTERFACE.....332

Appendice G AUTOCONFIG (MARQUE DEPOSEE).....335
Debogage des Cartes AUTOCONFIG.....336
Tableau de Spécification d'Adresses.....337

Appendice H Le Clavier.....343
La Communication dans le Clavier.....344
Code des Touches.....345
La Touche 'CAPS LOCK'.....345
La Condition "Perte de Synchro".....346
Séquence à la Mise sous Tension.....346
Averissement de Réinitialisation.....348
Réinitialisation Matérielle.....348
Codes Spéciaux.....349
Table de la Matrice.....350

Appendice I Spécifications sur l'Interface du Connecteur de l'Unité de Disquette Externe.....353
Généralités.....353
Tableau Récapitulatif.....354
Signaux en Rapport avec le Pilotage de l'Unité de Disquettes.....355
I.D. du Périphérique.... 357

Appendice J Fichier 'Include' pour les Exemples 'Hardware'.....359

Lexique.....365

Table des Matières.....373

INDEX DES PLANCHES

Planche 1-1 Organigramme de la Famille d'Ordinateurs Amiga.....11
Planche 2-1 Plan de Bits Entrelacé en RAM.....30
Planche 3-1 Création de l'Image Vidéo d'un Affichage.....34
Planche 3-2 Ce qu'est un Pixel.....35
Planche 3-3 Comment les Plans de Bits sélectionnent une Couleur.....37
Planche 3-4 Poids des Données des Plans de Bits lors d'une Sélection de couleurs.....38
Planche 3-5 Entrelacement.....44
Planche 3-6 Effet produit par le Mode Entrelacé sur les Arêtes des Objets.....44
Planche 3-7 Organisation de la Mémoire pour un Plan de Bits Simple.....48
Planche 3-8 Combinaison de Plans de Bits.....50
Planche 3-9 Placement de l'Affichage à l'Ecran.....51
Planche 3-10 Acquisition des Données de la Première Ligne pour Modulo = 0.....54
Planche 3-11 Acquisition des Données de la Deuxième Ligne pour Modulo = 0.....55
Planche 3-12 Affichage d'une Double Aire de Jeu 63
Planche 3-13 Nombre de Plans de Bits Assignés dans une Double Aire de Jeu.....64
Planche 3-14 Image Mémoire plus Grande que l'Affichage.....69
Planche 3-15 Acquisition des Données de la Première Ligne pour Modulo = 40.....69
Planche 3-16 Acquisition des Données de la Deuxième Ligne pour Modulo = 40.....70
Planche 3-17 Organisation des Données de la Moitié Droite de la Première Ligne de la Grande Image.....70
Planche 3-18 Organisation des Données de la Moitié Droite de la Deuxième Ligne de la Grande Image.....70
Planche 3-19 Position du Point d'Origine Horizontal de la Fenêtre d'Affichage.....72
Planche 3-20 Position du Point d'Origine Vertical de la Fenêtre d'Affichage.....72
Planche 3-21 Position du Point d'Arrêt Horizontal de la Fenêtre d'Affichage.....73
Planche 3-22 Position du Point d'Arrêt Vertical de la Fenêtre d'Affichage.....74
Planche 3-23 Déroulement Vertical.....76
Planche 3-24 Déroulement Horizontal.....78
Planche 3-25 Image en Mémoire plus Grande que la Fenêtre d'Affichage.....79
Planche 3-26 Données de la Ligne n.o 1 - Déroulement Horizontal.....79
Planche 3-27 Données de la Ligne n.o 2 - Déroulement Horizontal.....79
Planche 4-1 Définition de la Position du Sprite à l'Ecran.....95
Planche 4-2 Position de Sprites.....96
Planche 4-3 Forme d'Engin Spatial.....97
Planche 4-4 Sprite avec une Forme d'Engin Spatial Définie.....98
Planche 4-5 Définition de Couleur d'un Sprite.....99
Planche 4-6 Attributions des Regsitres de Couleur.....100

Planche 4-7 Organisation de la Structure de Données.....103
Planche 4-8 Priorité des Sprites.....112
Planche 4-9 Exemple Typique de Réemploi d'un Sprite.....113
Planche 4-10 Structure de Données Typique pour le Réemploi d'un Sprite....114
Planche 4-11 Chevauchement de Sprites (non Rattachés).....116
Planche 4-12 Mise de deux Sprites Côte-à-côte.....117
Planche 4-13 Circuits de Contrôle des Sprites.....122
Planche 5-1 Forme d'Onde Sinusoïdale.....131
Planche 5-2 Valeurs d'Amplitude Numérisées.....133
Planche 5-3 Exemple d'Onde Sinusoïdale.....139
Planche 5-4 Forme d'Onde avec Plusieurs Cycles.....149
Planche 5-5 Tracé dans le Domaine de Fréquences du Filtre Passe-bas.....151
Planche 5-6 Sortie sans Bruit (Sans Distorsion).....151
Planche 5-7 Distorsion.....152
Planche 5-8 Diagramme des Etats Audio.....162
Planche 6-1 De quelle Façon sont Stockées les Images en Mémoire.....165
Planche 6-2 Calculs pour BLTxP et BLTxMOD.....167
Planche 6-3 Diagramme de Venn pour les 'Minterms' du Blitter.....172
Planche 6-4 Extraction d'une Suite de Colonnes.....175
Planche 6-5 Utilisation du Bit FCI - le Bit est à 0.....179
Planche 6-6 Utilisation du Bit FCI - le Bit est à 1.....179
Planche 6-7 Exemple de Sommet à un Seul Point.....180
Planche 6-8 Octants pour le Tracé d'une Ligne.....184
Planche 6-9 Réservation de la Plage de temps pour les DMA.....190
Planche 6-10 Cycle Normal du 68000.....191
Planche 6-11 Plages de Temps Utilisées par un Affichage avec six Plans de Bits.....192
Planche 6-12 Plages de Temps Utilisées par un Affichage en Haute Résolution.....192
Planche 6-13 Schéma Fonctionnel par Blocs du 'Blitter'.....194
Planche 7-1 Priorités entre Sprites Fixées d'Avance.....202
Planche 7-2 Analogie en Rapport avec les Priorités Vidéo.....203
Planche 7-3 Priorité Sprite/Aire de jeu.....206
Planche 7 4 Priorité des Interruptions..... 216
Planche 8-1 Fiche du Contrôleur et Connecteur de l'Ordinateur.....222
Planche 8-2 Quadrature pour la Souris.....224
Planche 8-3 Connexions de la Manette de Jeu au Compteur.....227
Planche 8-4 Diagramme de Branchement Typique d'un Manche à Balai.....229
Planche 8-5 Effets de la Résistance sur le Taux de Charge.....230
Planche 8-6 Circuit de Charge du Potentiomètre.....231
Planche 8-7 Diagramme de Synchronisation de Chinon.....236
Planche 8-8 Diagramme de Synchronisation de Chinon (suite).....237
Planche 8-9 Le Clavier de l'A1000, où on peut voir le Code des Touches en Hexa.....249
Planche 8-10 Le Clavier de l'A500/2000, où on peut voir le Code des Touches en Hexa.....249
Planche 8-11 Aspect Initial du Registre SERDAT et du Registre à Décalage.....254
Planche 8-12 Aspect Final du Registre à Décalage.....254
Planche G-1 Comment Lire le Tableau de Spécification d'Adresses.....338

INDEX DES TABLEAUX

Tableau 2-1 Pour Interrompre le 68000.....31
Tableau 2-2 Récapitulation des Instructions du Copper.....32
Tableau 3-1 Les Couleurs d'une Aire de Jeu Simple.....39
Tableau 3-2 Une Partie de la Table de Couleurs.....40
Tableau 3-3 Contenu des Registres de Couleur.....41
Tableau 3-4 Extrait de Contenu des Registres de Couleur.....41
Tableau 3-5 Réglage du Nombre de Plans de Bits.....42
Tableau 3-6 Lignes d'une Aire de Jeu Normale.....43
Tableau 3-7 Besoins en Mémoire de l'Aire de Jeu, en NTSC.....46
Tableau 3-8 Besoins en Mémoire de l'Aire de jeu, en PAL.....47
Tableau 3-9 Récapitulation sur DIWSTRT et DIWSTOP.....53
Tableau 3-10 Registre de Couleur de l'Aire de Jeu n.o 1 en Mode Basse résolution.....65
Tableau 3-11 Registre de Couleur de l'Aire de Jeu n.o 2 en Mode Basse résolution.....65
Tableau 3-12 Registres de Couleur des Aires de Jau n.o 1 et n.o 2 en Mode Haute Résolution.....66
Tableau 3-13 Zone Verticale Maximale Disponible sur l'Ecran Vidéo.....74
Tableau 3-14 Zone Horizontale Maximale Disponible sur l'Ecran Vidéo.....75
Tableau 3-15 Contenu des Registres de Couleur.....87
Tableau 3-16 Quelques Valeurs des Registres et Couleurs Résultantes.....88
Tableau 3-17 Selection de Couleur en Basse résolution.....89
Tableau 3-18 Selection de Couleur en Mode 'Hold-and-modify'.....90
Tableau 3-19 Selection de Couleur en Haute Résolution.....91
Tableau 4-1 Structure de Données de Sprite.....102
Tableau 4-2 Registres de couleur du Sprite..... 105
Tableau 4-3 Paires de Registres de Couleur du Sprite 112
Tableau 4-4 Mots de Données de la Première Ligne du Sprite Engin Spatial.....118
Tableau 4-5 Registres de Couleur de Sprites Rattachés.....119
Tableau 4-6 Registres de Couleur d'un Sprite Simple.....127
Tableau 4-7 Registres de Couleur des Sprites Rattachés.....128
Tableau 5-1 Suite de Données d'Echantillons Audio du Canal 0.....135
Tableau 5-2 Valeurs de Volume.....137
Tableau 5-3 Bits de Validation du Canal Audio et des DMA.....141
Tableau 5-4 Interprétation des Données en Mode "Relié".....146
Tableau 5-5 Connexion de Canaux pour une Modulation.....147
Tableau 5-6 Relation entre Fréquence et Echantillonnage.....153
Tableau 5-7 Octave Tempérée d'un Echantillon de 16 Octets.....154

Tableau 5-8 Gamme Tempérée sur cinq Octaves.....156
Tableau 5-9 Valeurs en Décibels des Intervalles de Volume.....159
Tableau 6-1 Tableau des Valeurs Minterms Courantes.....171
Tableau 6-2 Séquence typique d'un Cycle du 'Blitter'.....183
Tableau 6-3 Bits de Code BLTCON1 pour Tracer les Octants d'une Ligne....184
Tableau 7-1 Bits dans BPLCON2.....204
Tableau 7-2 Priorité des Aires de Jeu Basées sur la valeur de
Bits PF1P2-PF1P0.....205
Tableau 7-3 Bits de CLXDAT.....208
Tableau 7-4 Bits de CLXCON.....209
Tableau 7-5 Contenu du Compteur de Position du Faisceau.....211
Tableau 7-6 Contenu du Registre.....218
Tableau 8-1 Connexions Typiques du Contrôleur.....223
Tableau 8-2 Déterminer la Direction de la Souris.....226
Tableau 8-3 Interprétation des Données dans JOY0DAT etJOY1DAT.....228
Tableau 8-4 Registres POTGO ($DFF034) et POTINP ($DFF016).....234
Tableau 8-5 Sous-Système du Lecteur de Disquettes.....237
Tableau 8-6 Registre DSKLEN ($DFF024).....240
Tableau 8-7 Registre DSKBYTR.....242
Tableau 8-8 Registres ADKCON et ADKCONR.....243
Tableau 8-9 Registres SERDATR / ADKCON.....251
Tableau G-1 Tableau de Spécification d'Adresses.....338

CHAPITRE 1

INTRODUCTION

La famille d'ordinateurs Amiga comprend différents modèles : chacun de ceux-ci a été conçu sur les mêmes principes pour fournir à l'utilisateur un ordinateur bon marché offrant des performances de haut niveau. L'Amiga atteint ce but par l'emploi d'un matériel adapté à base de silicium qui est à l'origine de ses spécificités graphiques et sonores avancées.

Il existe trois modèles différents qui composent la famille d'ordinateurs Amiga : l'A500, l'A1000 et l'A2000. Bien que ces modèles diffèrent entre eux pour leur prix et leurs particularités, ils ont en commun un noyau matériel qui rend leurs logiciels compatibles. Ce chapitre décrit les composants physiques des différents modèles Amiga et donne un bref aperçu de leurs caractéristiques graphiques et sonores.

LES COMPOSANTS DE L'AMIGA

Les composants physiques de l'Amiga sont les suivants :

- o Processeur principal Motorola MC68000 16/32 bits. En option, l'Amiga accepte aussi les processeurs 68010, 68020 et 68030.
- o 512 Ko de RAM en interne, extensible à un 1 Mo sur les A500 et les A2000.
- o 256 Ko de ROM contenant un système d'exploitation en temps réel avec des routines de support pour l'animation, le graphisme et le son.
- o Lecteur de disquettes interne de 3 pouces et demi, double face.
- o Port d'extension pour disquettes pour connecter jusqu'à trois lecteurs de disquettes supplémentaires; ceux-ci pourront être de 3 pouces et demi ou de 5 pouces et un quart, double face.
- o Port série RS-232-C entièrement programmable.
- o Port parallèle entièrement programmable.
- o Souris opto-mécanique avec deux boutons.
- o Deux ports contrôleur reconfigurables (pour souris, manette de jeux, crayon optique, manche à balai ou contrôleurs adaptés).
- o Un clavier professionnel avec pavé numérique, 10 touches de fonction et touches curseur. L'Amiga supporte aussi différents claviers internationaux.
- o Ports pour vidéo composite simultanée et sortie RVB numérique ou analogique.
- o Ports audio stéréo gauche et droite en provenance de quatre canaux audio spécialement adaptés.
- o Options pour des extensions permettant d'ajouter de la RAM, des lecteurs de disquettes supplémentaires (disquette ou disque dur), des périphériques ou des co-processeurs.

LE MC6X000 ET LES PUCES SPECIALISEES DE L'AMIGA

Le 68000 de Motorola est un micro-processeur 16/32 bits. La vitesse de l'horloge du système, pour les Amiga au standard NTSC, est de 7.15909 mégahertz (pour les Amiga au standard PAL, elle est de 7.09379 mégahertz). Ces vitesses peuvent varier si l'on emploie une horloge de système externe, comme celle d'un 'genlock'. Le 68000 possède un espace adressable de 16 méga-octets de mémoire vive non morcelée (RAM).

En plus du 68000, l'Amiga possède un 'hardware' spécifique, connu sous le nom de "puces spécialisées" ("custom chips") qui améliorent grandement le fonctionnement du système. L'expression "puces spécialisées" fait référence à trois circuits intégrés qui ont été conçus spécifiquement pour l'Amiga. Ces trois puces spécialisées (c'est-à-dire, Agnus, Paula et Denise) contiennent chacune la logique nécessaire à la gestion d'un ensemble de tâches spécifiques, comme la vidéo, le son, l'accès direct en mémoire (DMA) ou le graphisme.

Parmi d'autres, les puces spécialisées ont les tâches suivantes :

o Graphisme en haute résolution généré par plans de bits pouvant accepter à la fois les deux standards vidéo PAL et NTSC.

- Sur les systèmes NTSC l'Amiga produit typiquement un affichage non entrelacé de 320 sur 200 ou un affichage entrelacé de 320 sur 400 en 32 couleurs et un affichage non entrelacé de 640 sur 200 ou un affichage entrelacé de 640 sur 400 en 16 couleurs.

- Sur les systèmes PAL, l'Amiga produit typiquement un affichage non entrelacé de 320 sur 256 ou un affichage entrelacé de 320 sur 512 en 32 couleurs et un affichage non entrelacé de 640 sur 256 ou un affichage entrelacé de 640 sur 512 en 16 couleurs.

Des modes vidéo supplémentaires permettent un affichage à l'écran de 4096 couleurs simultanément (mode 'hold-and-modify') ou des affichages de hauteur et de largeur qui dépassent l'écran ('overscan').

o Un co-processeur spécialisé pour l'affichage qui permet d'effectuer des modifications dans la plupart des registres spécialisés en synchronisation avec la position du faisceau vidéo. C'est ce processeur qui permet des effets tels que la modification du milieu de l'écran, en accord avec la palette des couleurs, la subdivision de l'écran en plusieurs bandes horizontales, chacune avec un résolution et une intensité de couleur différentes, la génération d'une interruption du 68000 en synchronisation avec le faisceau vidéo, et plusieurs autres effets. Ce co-processeur peut se déclencher plusieurs fois par écran, au milieu des lignes et au début et à la fin de l'intervalle de suppression horizontale du faisceau vidéo. Le co-processeur lui-même peut affecter directement le contenu de plusieurs registres des autres puces spécialisés, libérant ainsi le 68000 pour les tâches de calcul courantes.

o 32 registres couleur destinés au système, chacun possédant un nombre sur douze bits, avec quatre bits d'information sur l'intensité du ROUGE, quatre sur celle du VERT et quatre sur celle du BLEU. Ceci permet au système d'avoir une palette de couleurs avec un choix différent de 4096 couleurs pour chaque registre.

o Huit sprites ('sprites') ré-utilisables, d'une largeur de 16 bits, avec un choix de 15 couleurs pour chaque pixel du sprite (lorsque les sprites sont mis par paires). Un sprite est un objet graphique qu'on peut déplacer facilement et dont l'affichage est totalement indépendant de l'arrière-plan où il se trouve (qui s'appelle une aire de jeu ('playfield')); les sprites peuvent s'afficher au-dessus ou au-dessous de l'arrière-plan où ils se trouvent. En basse résolution, un sprite a une largeur de 16 pixels et une hauteur d'un nombre de lignes arbitraire. Après avoir produit la dernière ligne d'un sprite à l'écran, on pourra utiliser un canal DMA de sprites pour créer une nouvelle image de sprite à un endroit quelconque de l'écran (en laissant au moins une ligne horizontale entre chaque nouvelle utilisation du processeur de sprites). Ainsi, on pourra créer plusieurs petits sprites en réemployant simplement les processeurs de sprites de manière appropriée.

o Priorité entre objets contrôlable de manière dynamique, avec détection de collision. Ce qui signifie que le système a la possibilité de contrôler d'une manière dynamique la priorité vidéo entre objets et arrière-plan, c'est-à-dire entre les sprites et les plans de bits (aires de jeu). Vous avez la possibilité de vérifier à tout moment quel objet ou quels objets apparaissent au-dessus ou au-dessous de l'arrière-plan.

En outre, vous aurez la possibilité d'utiliser le 'hardware' du système pour détecter des collisions entre objets, ainsi que pour faire réagir votre programme en conséquence.

o Processeur graphique spécialisé ('blitter'), employé pour déplacer des données rapidement, adaptable à l'animation des plans de bits. Ce processeur graphique a été conçu pour prendre des données à partir de trois sources de manière efficace, pour les assembler en une des 256 façons possibles et différentes et, en option, pour les stocker, une fois assemblées, dans une zone mémoire cible. Celui-ci est un des cas où le 68000 consacre des cycles de mémoire à un canal DMA qui peut effectuer ce travail de façon plus efficace (voir ci-dessous). Plus particulièrement, ce processeur graphique trace des lignes qui serviront de modèle dans des zones de mémoire organisées en rectangle à une vitesse d'environ un million de points par seconde; et il peut gérer avec efficacité le remplissage d'une zone mémoire.

o Audio représentée par quatre canaux numériques avec volume et taux d'échantillonnage programmables séparément. Les canaux audio prennent leur contrôle et leurs données en accédant directement à la mémoire. Une fois mis en route, chaque canal peut jouer une forme d'onde donnée automatiquement sans autre interaction avec le processeur. Deux canaux sont envoyés dans chacune des deux sorties audio stéréo. Les canaux audio peuvent être reliés ensemble pour faire de l'amplitude ou de la modulation en fréquence ou les deux formes de modulation en même temps.

o Lecture et écriture de disquette contrôlée par DMA sur toute une piste. Ce qui signifie que le lecteur de disquettes interne pourra lire jusqu'à 5600 octets de données en un seul tour de disquette (11 secteurs de 512 octets chacun).

La mémoire interne -que se partagent les puces spécialisées et le 68000- s'appelle aussi mémoire 'chip'. A l'origine, les puces spécialisées de l'Amiga ont été conçues pour pouvoir accéder physiquement jusqu'à 512 Ko de cette mémoire qu'ils se partagent. La nouvelle version de la puce spécialisée Agnus a été créée pour permettre au 'hardware' destiné au graphisme et à l'audio d'avoir accès à un mégaoctet de mémoire.

Les modèles Amiga 500 et 2000 ont été conçus pour pouvoir accepter la nouvelle puce spécialisée Agnus, qui s'appelle "Fat Agnus" à cause de sa forme carrée. Dorénavant, l'A500 et l'A2000 pourront se réserver un espace de mémoire 'chip' d'un mégaoctet. Tout cet espace dépendra de la logique d'arbitrage qui contrôle les accès au CPU et aux puces spécialisées. Sur l'Amiga 1000, seuls les premiers 512 Ko d'espace mémoire sont constitués de mémoire partagée, c'est-à-dire de mémoire 'chip'.

Les puces spécialisées et le 68000 se partagent la mémoire de manière totalement imbriquée. Etant donné que seul le 68000 aura besoin d'accéder au bus mémoire lors d'un cycle d'horloge sur deux, pour pouvoir fonctionner à pleine vitesse, le reste du temps le bus mémoire sera libre pour d'autres activités. Les puces spécialisées utilisent le bus mémoire pendant ces cycles inoccupés et permettent effectivement au 68000 de fonctionner à sa vitesse nominale la plus élevée la plupart du temps. Nous disons "la plupart du temps" car il y a des cas où les parties 'hardware' spécialisées volent des cycles de mémoire au 68000 pour des raisons d'efficacité. D'une façon plus spécifique, le co-processeur et le canal DMA qui déplace les données et qui a appelé le 'blitter' peuvent, chacun, voler du temps de travail au 68000 pour des actions qu'ils peuvent effectuer avec plus de rendement que le 68000 lui-même. Pour cette raison, les canaux DMA du système ont été conçus en pensant à l'efficacité de leur fonctionnement. Le travail qu'ils doivent accomplir est exécuté par les composants 'hardware' les plus efficaces se trouvant sur le marché. Même quand ces canaux volent des cycles de temps de travail, cette action se limite à bloquer l'accès du 68000 seulement pour ce qui concerne la mémoire partagée, c'est-à-dire interne. Quand les canaux DMA utilisent la ROM ou la mémoire externe, le 68000 continue à fonctionner à pleine vitesse.

Une autre caractéristique importante du 'hardware' de l'Amiga est la capacité qu'il a de contrôler de façon dynamique quelle partie de mémoire 'chip' on est en train d'utiliser pour l'affichage de l'arrière-plan, pour l'audio et pour les sprites. L'Amiga ne se limite pas à une petite portion spécifique de RAM pour un tampon de trame. Au contraire, le système permet d'allouer une partie quelconque de la mémoire 'chip' pour l'affichage des plans de bits, pour les listes de contrôle du processeur de sprites, pour les listes d'instructions du co-processeur ou celles des canaux audio.

Le 'blitter' peut aussi accéder à cette même portion de mémoire. Ce qui signifie, par exemple, que l'utilisateur pourra stocker des parties d'image dans des zones choisies en mémoire 'chip' et les employer pour des effets d'animation en les remplaçant rapidement à l'écran pendant qu'il sauvegarde et restaure les images d'arrière-plan. En effet, l'Amiga comprend un support programmé pour la définition de l'affichage ainsi qu'un contrôle et un support pour les objets animés qui se trouveraient imbriqués dans des aires de jeux.

INTERFACE DIRECTE POUR CAMERA VIDEO ET POUR MAGNETOSCOPE

En plus des connecteurs destinés aux moniteurs RVB numériques ou analogiques et aux moniteurs composites monochromes, l'Amiga peut être élargi pour inclure une interface pour une caméra vidéo ou un magnétoscope. Ce système peut se synchroniser avec une source vidéo externe et peut remplacer la couleur d'arrière-plan du système avec une image venant de l'extérieur. Cette caractéristique permet le développement d'images vidéo entièrement intégrées aux graphismes générés par l'ordinateur. De la même manière, l'Amiga accepte une entrée CD.

PERIPHERIQUES

Le stockage sur disquette se fait par un lecteur de disquettes de 3 pouces et demi en interne. Les disquettes ont 80 pistes, sont double face, et ont 11 secteurs formatés par piste, 512 octets par secteur (plus de 900000 octets par disquette). Le contrôleur de disquettes peut lire et écrire des disquettes 5 pouces et un quart et 3 pouces et demi au format IBM PCTM de 320/360 Ko, ainsi que des disquettes 3 pouces et demi au format IBM PC (MS-DOS) de 640/720 Ko. On peut ajouter des lecteurs de disquettes externes de 5 pouces et un quart et de 3 pouces et demi par l'intermédiaire d'un connecteur d'extension. L'ensemble des circuits de ces périphériques se trouve dans Paula. D'autres puces gèrent différents signaux qui n'ont pas été confiés d'une manière spécifique aux puces spécialisées, y compris les contrôles concernant le modem, les tests sur l'état de la disquette, les contrôles concernant le déplacement de la tête et ceux du moteur du lecteur de disquette, la mise en service de la ROM, l'interface parallèle d'E/S et l'interface du clavier.

L'Amiga comprend en standard un port série RS-232-C pour des périphériques externes d'entrée/sortie.

Un clavier avec pavé numérique, touches de contrôle du curseur et 10 touches de fonction est compris dans le système de base. Pour plus de flexibilité, le système envoie soit le signal de la touche qu'on enfonce soit celui de la touche qu'on relâche. L'Amiga accepte en outre différents claviers internationaux. On pourra connecter à l'unité de base plusieurs autres types de contrôleurs par l'intermédiaire des deux ports contrôleur. On pourra employer un contrôleur de souris, de manette de jeux, de clavier numérique, de boule roulante, de crayon optique ou de volant de pilotage dans l'un des deux ports.

EXTENSION ET POSSIBILITE D'ADAPTATION DU SYSTEME.

Dans tous les modèles Amiga, on peut ajouter facilement des nouveaux périphériques externes. Ceux-ci seront reconnus automatiquement et utilisés par la partie logicielle du système par l'intermédiaire d'AUTOCONFIG, une procédure de transition bien définie et bien documentée.

Sur les modèles A500 et A1000, on peut ajouter des périphériques externes par l'intermédiaire du connecteur d'extension 86 broches de l'Amiga, ainsi que de la RAM externe supplémentaire. On peut ajouter des unités de disquettes supplémentaires à partir du connecteur qui se trouve à l'arrière de cette même unité.

Le modèle A2000 offre à l'utilisateur les mêmes caractéristiques que l'A500 ou l'A1000, mais offre en plus un confort pour ce qui est d'une extension simple et importante du système. Sur l'A2000 on n'a pas accès au connecteur externe 86 broches de l'A1000 et de l'A500. Par contre, l'A2000 contient sept emplacements internes qui permettent d'ajouter facilement et rapidement plusieurs type de cartes d'extension à l'ordinateur. Ces cartes pourront contenir des co-processeurs, une extension RAM, des contrôleurs de disque dur, des ports d'E/S ou des ports vidéo. Il y a aussi de la place pour installer à l'intérieur le disque dur et les lecteurs de disquettes. L'A2000 accepte aussi la carte co-processur spéciale Bridgeboard. Celle-ci offre un IBM PC complet sur une seule carte et permet à l'Amiga d'utiliser des logiciels compatibles MS-DOS, tout en utilisant simultanément ses logiciels propres.

AU SUJET DES EXEMPLES

Les exemples de ce livre montrent tous des manipulations directes du 'hardware' de l'Amiga. Cependant, en règle générale, il est interdit d'accéder directement au 'hardware' soit pendant que le 'software' contrôle le système soit pendant que le 'software' n'a pas fait un arbitrage -par l'intermédiaire du SE sur un accès exclusif- sur les différentes parties du 'hardware' qu'on souhaite pouvoir contrôler.

Presque tout le 'hardware', dont nous parlerons dans ce manuel, et plus particulièrement le 'Blitter', le 'Copper', l'aire de jeux, les sprites, le CIA, le 'trackdisk' et le 'hardware' qui contrôle le système, est utilisé en mode exclusif ou se trouve sous l'arbitrage de certaines parties du SE de l'Amiga dans tout système Amiga en fonctionnement. Le reste du 'hardware', c'est-à-dire celui qui s'occupe de l'audio, des ports série et parallèle, pourra se trouver en état de fonctionnement lors d'applications qui auront prévu son utilisation par l'intermédiaire du 'software' du système.

Avant d'essayer de manipuler directement une partie du 'hardware', dans l'environnement multitâche de l'Amiga, votre application devra se garantir l'accès exclusif à ce même 'hardware' par l'intermédiaire de la bibliothèque du système d'exploitation, du périphérique ou de la ressource logique qui fait les arbitrages sur la maîtrise de l'accès. Les fonctions du système d'exploitation pour demander et obtenir le contrôle du 'hardware' de l'Amiga sont variées et leur analyse n'entre pas dans le cadre de ce manuel. D'une manière générale, ces fonctions, lorsqu'elles sont disponibles, se trouvent dans la bibliothèque, le périphérique ou la ressource logique qui s'occupe de la gestion de la partie 'hardware' concernée dans un environnement multitâche. La liste qui suit vous aidera à trouver les fonctions du système d'exploitation appropriées ou les mécanismes pouvant exister pour accéder, après arbitrage, au 'hardware' dont nous parlerons dans ce manuel.

Copper, Playfield, Sprite, Blitter - graphics.library
Audio - audio.device
Trackdisk - trackdisk.device, disk.resource
Serial - serial.device, misc.resource
Parallel - parallel.device, cia.resource, misc.resource
Gameport - input.device, gameport.device, potgo.resource
Keyboard - input.device, keyboard.device
System Control - graphics.library, exec.library (interruptions)

La plupart des exemples de ce livre utilise le fichier hw_examples.i (cf. Appendice J) pour définir le nom des registres des puces. Hw_examples.i utilise le fichier 'include' du système hardware/custom.i pour définir les structures des puces et leurs adresses relatives. Les valeurs définies dans hardware/custom.i et hw_examples.i sont des 'offsets' établis à partir de la base de l'espace d'adressage du registre de la puce. En général, cette valeur de base est définie en tant que _custom (utilisateur) et est résolue lors de l'édition de liens par l'amiga.lib (_ciaa et _ciab sont résolues de manière identique).

Normalement, l'adresse de base est chargée dans un registre d'adresse et les 'offsets' fournis par hardware/custom.i et hw_examples.i sont employés par la suite pour adresser le bon registre.

NOTE

Les valeurs d' 'offset' des registres sont les adresses que le Copper doit employer pour parler aux registres.

par exemple, en assembleur :

```
        INCLUDE "exec/types.i"
        INCLUDE "hardware/custom.i"

        XREF custom                 ; Référence externe

Start:
        lea    _custom,a0           ; Utilisez a0 comme registre de base
        move.w #$7FFF,intena(a0)    ; Inhibe toutes les interruptions
```

En C, vous emploieriez les définitions de structure de hardware/custom.h.
Par exemple :

```
#lnclude        "exec/types.h"
#include        "hardware/custom.h"

extern  struct  Custom  custom;

/*  Vous pourriez avoir besoin de définir la référence externe ci-dessus
**  en tant que 'extern struct Custom far custom;'
**  Consultez le manuel de votre compilateur.
*/

main()
{
custom.intena = 0x7FFF;         /* Inhibe toutes les interruptions */
}
```

Les fichiers 'include' qui concernent le 'hardware' de l'Amiga sont fournis en général avec le compilateur ou l'assembleur. Le listage de ces fichiers se trouve aussi dans l' "Amiga ROM Kernel Manual : Includes and Autodocs", édité par Addison-Wesley. En général, les noms d'étiquette des fichiers 'include' sont très similaires à ceux de la liste des registres 'hardware' équivalents avec, cependant, ces quelques différences caractéristiques.

- Les registres d'adresse comprenant un mot de poids fort et un mot de poids faible sont généralement listés comme des registres ayant une capacité de deux mots dans la liste des registres 'hardware' et chaque nom de registre contient soit un suffixe soit il inclut un "H" ou un "L" pour fort ('high') et faible ('low'). L'étiquette du fichier 'include' de ce même registre traitera généralement le registre tout entier en tant que mot long (32 bits) et, par conséquent, ne contiendra pas la distinction entre "L" ou "H".

- Les registres séquentiels associés à qui l'on donne des noms individuels ayant des nombres comme suffixe, dans la liste des registres 'hardware', sont généralement référencés, dans les fichiers 'include', à partir d'une unique définition du registre de base. Par exemple, les registres pour la couleur de la liste 'hardware' (COLOR00, COLOR01, etc.) sont référencés à partir de l'étiquette "color" telle qu'elle définie dans "hardware/custom.i" (color+0, color+2, etc.).

- On pourra trouver des exemples sur la manière de définir correctement l' 'offset' d'un registre dans le fichier hw_examples.i dont nous donnons le listage dans l'Appendice J.

QUELQUES MISES EN GARDES DESTINEES AUX PROGRAMMEURS AU NIVEAU DU 'HARDWARE'

L'Amiga est disponible en plusieurs modèles et différentes configurations et se diversifie en outre à cause de l'abondance des périphériques d'extension additionnels et de la possibilité de remplacer son processeur. De plus, même le 'hardware' standard de l'Amiga, comme le clavier ou les lecteurs de disquettes, est fourni par des nombreux constructeurs différents et peut changer imperceptiblement soit pour ce qui est de sa synchronisation, soit pour ce qui est des possibilités qu'il offre de fonctionner au-delà des performances annoncées.

Le système d'exploitation de l'Amiga a été conçu pour faire fonctionner le 'hardware' de l'Amiga dans la limite de ses spécifications, pour pouvoir s'adapter à différents composants 'hardware' et à différentes configurations de RAM et, d'une manière générale, pour offrir une compatibilité ascendante avec les mises à niveau 'hardware' futures ou avec les produits additionnels envisagés par des concepteurs. Pour une compatibilité ascendante maximale, nous suggérons vivement aux programmeurs de travailler sur le 'hardware' par l'intermédiaire des commandes et des fonctions qu'offre le système d'exploitation de l'Amiga.

S'il vous paraît nécessaire de devoir programmer le 'hardware' directement, vous devrez alors écrire un code qui devra fonctionner proprement sur des modèles et des configurations différents. Faites attention à demander et à obtenir correctement le contrôle du 'hardware' que vous manipulez, et portez une attention toute particulière à ce qui suit :

Ne faites jamais des appels en ROM. Prenez garde à tout code d'exemple qui appelle des routines se situant dans l'intervalle qui va de $F80000 à $FFFFFF. Il s'agit d'adresses en ROM et ces routines CHANGERONT d'adresse à chaque révision du SE. Le seul code ROM accepté comme interface au système est celui qu'on obtient en faisant appel à la bibliothèque, au périphérique ou à la ressource logique fournis.

Ne modifiez jamais le format des structures privées du système et n'en soyez pas dépendant. Ce format comprend l'écriture en mémoire (le "pokage") des listes du 'copper', celui des listes mémoire ainsi que les adresses de base des bibliothèques de fonctions.

Ne soyez jamais dépendant d'une adresse qui contiendrait une structure propre au système ou un type de mémoire. Les modules du système réservent leur espace mémoire de manière dynamique lors de l'initialisation. Les adresses de structures du système diffèrent avec chaque nouveau SE, chaque modèle et chaque configuration, tout comme diffère la quantité de mémoire disponible et l'utilisation de la pile du système. N'oubliez pas que toutes les données se rapportant à un accès direct des puces spécialisées doivent se trouver en CHIP RAM. Ces données comprennent les images (plans de bits, sprites, etc.), l'échantillonnage de sons, les tampons du 'trackdisk' et les listes du 'copper'.

N'écrivez pas des données inexactes ou alors interprétez les données non définies à partir des bits actuellement non utilisés ou des adresses qui se trouvent dans l'espace des puces spécialisées. Tous les bits non définis devront être mis à zéro lors d'une écriture et ignorés lors d'une lecture.

N'écrivez pas de données au-delà de l'espace des puces spécialisées. Ces puces pourraient être modifiées ou améliorées pour fournir des registres supplémentaires ou pour pouvoir utiliser les bits non définis dans les registres qui existent déjà.

Tous les registres des puces spécialisées sont destinés seulement à la lecture OU à l'écriture. N'essayez pas de lire des registres destinés à l'écriture et n'essayez pas d'écrire dans les registres de lecture.

Ne lisez pas dans l'espace mémoire actuellement non défini, n'y écrivez rien ou, mieux encore, ne vous en servez pas. Commodore se réserve l'utilisation actuelle et future de cet espace mémoire qui sera sujet à des changements.

Si vous utilisez les bibliothèques du système, celles des périphériques et des ressources logiques, vous devrez employer l'interface telle qu'elle a été définie. Les programmeurs en assembleur (et ceux qui écrivent des compilateurs) devront entrer leurs fonctions par l'intermédiaire des tables de saut de base des bibliothèques, en leur passant les arguments en tant que mots longs et l'adresse de base de la bibliothèque dans A6. Le résultat obtenu en retour dans D0 devra être testé, et le contenu de D0-D1/A0-A1 devra être considéré comme valable après un appel au système.

NOTE

On ne devra pas employer l'instruction assembleur TAS dans aucun programme destiné à l'Amiga. Cette instruction choisit une action 'read-modify-write' (lecture-modification-écriture) de manière implicite, mais ce choix pourrait être mis en échec par les DMA du système. Employez à sa place BSET et BCLR. Ces instructions exécutent un test et déclenchent une action qui ne pourra être interrompue.

L'instruction TAS est nécessaire seulement sur un système avec plusieurs processeurs. Sur un système à processeur unique, les instructions BSET et BCLR sont identiques à TAS, étant donné que le 68000 n'interrompt pas les instructions au milieu. BSET et BCLR effectuent d'abord un test, puis valident les bits.

N'employez pas des instructions privilégiées appartenant à un autre processeur de la famille du 68000, surtout MOVE SR,<ea> qui est privilégiée sur les processeurs 68010/20/30. Employez la fonction GetCC() d'Exec au lieu de MOVE SR, ou employez alors l'instruction appropriée non privilégiée, comme nous l'indiquons ci-dessous :

CPU	User Mode	Super Mode
68000	MOVE SR,<ea>	MOVE SR,<ea>
68010/20/30	MOVE CCR,<ea>	MOVE SR,<ea>

Toutes les adresses doivent avoir 32 bits. N'employez pas les huit bits de poids fort pour d'autres données et ne vous servez pas de variables signées ou de signes mathématiques pour les adresses.

Ne faites pas exécuter du code dans votre pile - ou alors utilisez du code auto-modifiable - car ce type de code pourra être mis en échec à cause de sa mise en anté-mémoire, propre à certains processeurs de la famille 68xxx. Et n'utilisez jamais des boucles de code dépendant de la vitesse d'horloge ou du processeur pour synchroniser les temps d'attente. Pour d'autres informations sur l'utilisation du 'timer' du 8520 pour les temps d'attente, cf. l'Appendice F.

NOTE

Pendant l'échantillonnage d'un registre qui répond à un ordre de lecture ou à un ordre d'écriture (par exemple 'copjmp2'), assurez-vous que vous utilisez un MOVE.W #$00, et non un CLR.W. L'instruction CLR provoque sur un 68000 un ordre de lecture et une remise à zéro (deux accès), mais un seul accès sur un 68020 et d'autres processeurs plus récents. Ceci donnera des résultats différents avec des processeurs différents.

Si vous programmez au niveau du 'hardware', vous devrez suivre les spécifications d'interfaçage le concernant. Tout le 'hardware' N'est PAS identique. Ne partez pas du principe que des 'hacks' à bas niveau pour la vitesse et la protection contre la copie fonctionnent avec tous les lecteurs de disquettes ou avec tous les claviers ou sur tous les systèmes ou sur des systèmes à venir. Testez vos programmes sur plusieurs systèmes différents, avec des processeurs, des SE, du 'hardware' différents et des configurations de RAM différentes.

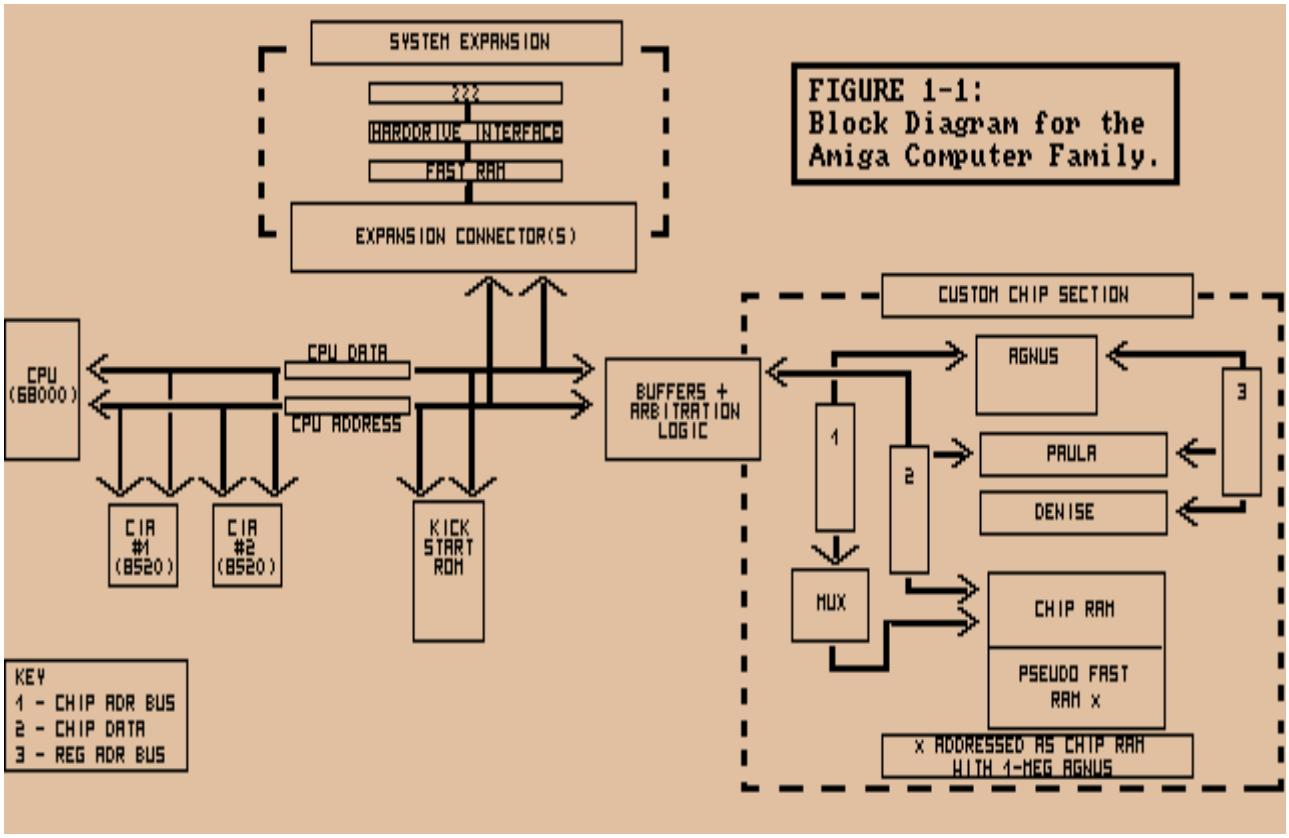


PLANCHE 1-1 : Organigramme de la Famille d'Ordinateurs Amiga.

Chapitre 2

LE 'HARDWARE' DU COPROCESSEUR

INTRODUCTION

Le Copper est un co-processeur polyvalent sur commande qui se trouve dans l'une des puces spécialisées de l'Amiga. Celui-ci prend ses informations en accédant directement à la mémoire (DMA). Le Copper peut contrôler presque tout le système graphique, libérant ainsi le 68000 qui peut se consacrer à la partie logique du programme; il peut aussi affecter directement le contenu de la plupart des registres de controle des puces. Il s'agit d'un outil vraiment puissant pour gérer les modifications du milieu de l'écran lors de l'affichage d'un graphisme et pour gérer les changements intervenant dans les registres, changements qui doivent se produire pendant les périodes de suppression verticale. Parmi d'autres choses, ce co-processeur a la possibilité de contrôler la remise à jour des registres, le replacement des sprites, le changement de la palette des couleurs, la remise à jour des canaux sonores et le contrôle du 'blitter'.

L'un des traits caractéristiques du Copper est sa faculté d'ATTENDRE (WAIT) une position bien précise du faisceau vidéo, puis de TRANSFERER (MOVE) les données dans un registre du système. Pendant la période d'attente (WAIT), le Copper examine directement le compteur de position du faisceau vidéo. Ce qui signifie que -pendant que le Copper attend pour que le faisceau atteigne une position déterminée- celui-ci n'utilise aucunement le bus mémoire. Par conséquent, ce bus est libre et peut être employé par les autres canaux DMA ou par le 68000.

Lorsque la condition WAIT a été satisfaite, le Copper vole des cycles de mémoire au 'blitter' ou au 68000 pour déplacer des données déterminées dans un registre spécialisé qui aura été sélectionné.

Le Copper est un processeur à deux cycles qui demande l'accès au bus seulement pendant les cycles de mémoire impairs. Cette caractéristique prévient toute forme de collision avec l'audio, la disquette, le rafraîchissement, les sprites et la plupart des accès DMA pour l'affichage en en basse résolution, avec tout ce qui utilise seulement des cycles mémoire impairs. Le Copper, par conséquent, a besoin d'une priorité seulement sur le 68000 et le 'blitter' (les canaux DMA qui gèrent l'animation, le tracé de lignes et le remplissage des polygones).

Comme tous les autres canaux DMA du système Amiga, le Copper peut prendre ses instructions seulement dans une zone de RAM 'chip' de la mémoire du système.

AU SUJET DE CE CHAPITRE

Dans ce chapitre, nous apprendrons comment utiliser le jeu d'instructions propres au Copper pour organiser la modification de la valeur d'un registre de milieu d'écran et la préparation du registre pointeur pendant l'intervalle de suppression verticale. Le chapitre montre comment organiser les instructions du Copper à l'intérieur de listes Copper, comment utiliser les listes Copper en mode entrelacé et comment utiliser le Copper avec le 'blitter'. Dans ce chapitre, nous traiterons du Copper d'une manière générique. Les chapitres qui traitent des aires de jeu, des sprites, de l'audio et du 'blitter' contiennent des propositions plus spécifiques pour l'utilisation du Copper.

DEFINITION DES INSTRUCTIONS DU COPPER

En tant que co-processeur, le Copper ajoute son propre jeu d'instructions aux instructions déjà fournies par le 68000. Le Copper ne dispose que de trois instructions, mais, avec celles-ci, on a la possibilité de faire un tas de choses :

- o WAIT attend une position précise de l'écran indiquée par ses coordonnés x et y.

- o MOVE transfère une valeur de données immédiate n à l'intérieur des registres spécialisés.

- o SKIP saute l'instruction suivante si le faisceau vidéo a déjà atteint une position déterminée de l'écran.

Toutes les instructions du Copper sont composées de deux mots de 16 bits situés à des emplacements mémoire de manière séquentielle. Chaque fois que le Copper prélève une instruction, il prélève les deux mots. Les instructions MOVE et SKIP nécessitent deux cycles mémoire et deux mots pour l'instruction elle-même. Étant donné que le Copper n'exige que des cycles de mémoire pairs, chaque instruction aura besoin d'un temps égal à quatre cycles de mémoire. L'instruction WAIT nécessite trois cycles de mémoire et a besoin d'un temps égal à six cycles mémoire; elle prend un cycle de mémoire supplémentaire pour être "réveillée".

Bien que le Copper ne puisse affecter que les registres machine, il peut aussi affecter la mémoire en mettant en route une opération en provenance du 'blitter'. Vous pourrez trouver plus de précisions sur la manière d'utiliser le Copper pour contrôler le 'blitter' dans la section "Registre de Contrôle" et dans la section "Utilisation du Copper avec le 'Blitter'".

Nous décrivons ci-dessous les instructions WAIT et MOVE. L'instruction SKIP sera décrite dans la section "Sujets Avancés".

L'INSTRUCTION MOVE

L'instruction MOVE transfère des données de la RAM à un registre cible. Les données transférées se trouvent dans le second mot de l'instruction MOVE; le premier mot contient l'adresse du registre cible. Nous montrons cette procédure en détail dans la section "Sommaire des Instructions du Copper".

PREMIER MOT DE L'INSTRUCTION (IR1)

Bit 0	Toujours mis à 0.
Bits 8 - 1	Adresse du registre cible (DA8-1).
Bits 15 - 9	Inutilisé, mais devra être mis à 0.

SECOND MOT DE L'INSTRUCTION (IR2)

Bits 15 - 0	Les 16 bits de données qui devront être transférées (déplacées) dans le registre cible.

Le Copper peut stocker des données dans les registres suivants :

- Tout registre dont l'adresse est égale ou supérieure à $20 [1].

- Tout registre dont l'adresse se trouve dans un intervalle qui va de $10 à $20, si le bit de "danger" du Copper est présent dans le registre de contrôle COPCON du Copper, décrit dans la section "Registre de Contrôle".

- Le Copper ne peut écrire dans un registre dont l'adresse est inférieure à $10.

Vous trouverez dans l'Appendice B toutes les adresses des registres machine.

Les instructions MOVE de l'exemple qui suit montrent le pointeur du plan de bits numéro 1 à l'adresse $21000 et le pointeur du plan de bits numéro 2 à l'adresse S25000 [2].

```
DC.W   $00E0,$0002    ;Transfère $0002 dans le registre $0E0 (BPL1PTH)
DC.W   $00E2,$1000    ;Transfère $1000 dans le registre $0E2 (BPL1PTL)
DC.W   $00E4,$0002    ;Transfère $0002 dans le registre $0E4 (BPL2PTH)
DC.W   $00E6,$5000    ;Transfère $5000 dans le registre $0E6 (BPL2PTL)
```

Normalement, les fichiers ".i" assembleur sont inclus pour qu'on puisse employer l es noms plutôt que les adresses pour référencer les registres 'hardware'.
Nous recommandons vivement de référencer toutes les adresses 'hardware' par leurs noms tels qu'ils sont définis dans les fichiers 'include' du système. Ceci permet d'adapter plus facilement les programmes à tirer avantage du 'hardware' futur ou des améliorations qu'on pourrait apporter à l'avenir. Par exemple :

```
INCLUDE "hardware/custom.i"

DC.W   bplpt+$00,$0002      ;Transfère $0002 dans le registre $0E0 (BPLlPTH)
DC.W   bplpt+$02,$1000      ;Transfère $1000 dans le registre $0E2 (BPLlPTL)
DC.W   bplpt+$04,$0002      ;Transfère $0002 dans le registre $0E4 (BPL2PTH)
DC.W   bplpt+$06,$5000      ;Transfère $5000 dans le registre $0E6 (BPL2PTL)
```

Pour qu'on puisse utiliser les exemples de ce manuel, nous avons créé un fichier 'include' spécial (cf. Appendice J) qui définit tous les noms des registres 'hardware' à partir du fichier "hardware/custom.i". Nous avons fait ceci pour rendre les exemples plus lisibles du point de vue du 'hardware'. Plusieurs exemples de ce manuel se trouvent ici pour aider à expliquer le 'hardware' et, dans plusieurs cas, ils ne sont pas utiles sans modifications et une bonne quantité de code supplémentaire.

[1] Les nombres hexadécimaux se distinguent des nombres décimaux par la présence du préfixe $.
[2] Tous les extraits de segments de code sont en langage assembleur.

L'INSTRUCTION WAIT

L'instruction WAIT oblige le Copper à attendre jusqu'à ce que les compteurs de faisceau vidéo soient égaux (ou supérieurs) aux coordonnés indiquées dans l'instruction. Pendant qu'il attend, le Copper n'utilise pas le bus et ne consomme pas de cycles de mémoire.

Le premier mot de l'instruction contient les coordonnés verticale et horizontale de la position où se trouve le faisceau vidéo. Le deuxième mot contient les bits de validation utilisés pour former un "masque" qui dit au système quels bits de position du faisceau vidéo il faudra employer lors d'une comparaison.

PREMIER MOT DE L'INSTRUCTION (IR1)

Bit 0	Toujours mis à 1.
Bits 15 - 8	Position verticale du faisceau (qu'on appelle VP).
Bits 7 - 1	Position horizontale du faisceau (qu'on appelle HP).

SECOND MOT DE L'INSTRUCTION (IR2)

Bit 0	Toujours mis à 0.
Bit 15	Le bit de mise hors service définitive du blitter. Normalement, ce bit est mis à 1. (Cf. ci-dessous la section "Sujets Avancés".)
Bits 14 - 8	Bits de mise en service pour comparaison avec la position verticale (qu'on appelle VE).
Bits 7 - 1	Bits de mise en service pour comparaison avec la position horizontale (qu'on appelle HE).

L'instruction WAIT de l'exemple qui suit attend la ligne de balayage 150 ($96) pendant qu'on a masqué la position horizontale.

```
DC.W    $9601,$FF00     ; Attend la ligne 150,
                        ; ignore le compteur horizontal.
```

L'instruction WAIT de l'exemple qui suit attend la ligne de balayage 255 et la position horizontale 254. Cet événement ne se produira jamais, ainsi le Copper s'arrête jusqu'à l'intervalle de suppression verticale.

```
DC.W    $FFFF,$FFFE     ; Attend la ligne 255,
                        ; H = 254 (met fin à la liste du Copper).
```

Pour comprendre pourquoi la position VP=$FF HP=$FE n'interviendra jamais, vous devez regarder l'opération de comparaison du Copper et les restrictions de taille concernant l'information sur la position.
La ligne numéro 255 est une ligne valide à attendre, il s'agit en effet de la valeur maximale pouvant convenir à ce champ.
Étant donné que 255 est un chiffre maximal, la ligne qui suit sera mise à zéro (la ligne 256 apparaît en tant que zéro lors de la comparaison).
Le numéro de ligne ne devra jamais être supérieur à $FF. La valeur maximale de la position horizontale est égale à $E2. Ce qui signifie que le chiffre plus élevé pouvant jamais apparaître lors d'une comparaison sera $FFE2.
En attendant $FFE2, on atteindra la ligne $FF, mais la position horizontale $FE n'interviendra jamais. Ainsi, la position n'atteindra jamais la valeur $FFFE.
Nous pourrions être tentés d'attendre la position horizontale $FE (du moment qu'elle n'intervient jamais) et de mettre un chiffre plus bas dans le champ qui indique la position verticale. Mais ceci ne conduira pas au résultat escompté.
L'opération de comparaison attend que la position du faisceau vidéo soit supérieure ou égale à celle de la valeur rentrée.
Si la position verticale n'est pas égale à $FF, alors, dès que le numéro de ligne devient supérieur à celui que nous avons rentré, la comparaison l'estimera valide (vrai) et l'attente prendra fin.

Ces notes sur la position horizontale et verticale du faisceau s'appliquent à l'instruction WAIT autant qu'à l'instruction SKIP. Nous la décrivons ci-dessous, dans la section "Sujets Avancés".

POSITION HORIZONTALE DU FAISCEAU

La position horizontale du faisceau a une valeur comprise entre $0 et $E2. Le bit de poids faible n'est pas utilisé dans la comparaison, ainsi il y a 113 positions disponibles pour les opérations du Copper. Ce qui correspond à 4 pixels en basse résolution et à 8 en haute résolution. La suppression horizontale se situe dans une fourchette comprise entre $0F et $35. L'écran standard (d'une largeur de 320 pixels) possède une partie horizontale, comprise entre $04 et $47, qui n'est pas occupée (pendant la suppression, seule s'affiche la couleur d'arrière-plan).

En NTSC, toutes les lignes n'ont pas la même longueur. Seule une ligne sur deux sera une ligne paire (avec une fréquence de couleur égale à 228, 0-$E3), tandis que les autres lignes auront une longueur de fréquence de couleur égale à 227. En PAL, les lignes ont toutes une longueur de 227. L'affichage considère ces lignes comme ayant une longueur de fréquence de couleur égale à 227 et 1/2, tandis que le Copper les voit alternativement comme des lignes paires et des lignes impaires.

POSITION VERTICALE DU FAISCEAU

La position verticale du faisceau peut se résoudre en une seule ligne, avec une valeur maximale de 255. Actuellement, en NTSC, il existe 262 positions verticales possibles (en PAL, il y en a 312). Des complications d'ordre mineur peuvent intervenir si vous souhaitez que quelque chose se produise à l'intérieur des dernières six ou sept lignes de balayage. Comme il n'y a que huit bits de résolution pour la position verticale du faisceau (ce qui permet 256 positions différentes), nous montrons ci-dessous une des manières les plus simples pour pouvoir gérer tout cela.

INSTRUCTION	EXPLICATION
[... autres instructions ...]	
WAIT la position (0,255)	Ici, le compteur vertical semble se mettre à zéro car la comparaison fonctionne sur les bits de poids faible du comptage vertical.
WAIT une position horizontale quelconque avec une position verticale de 0 à 5 qui couvre les six dernières lignes du balayage avant que n'intervienne la suppression verticale.	Ainsi, la totalité des 256 + 6 = 262 lignes du faisceau vidéo continuent à fonctionner pendant que les instructions du Copper sont exécutées.

NOTE

Le vertical est identique à l'horizontal - étant donné qu'il y a alternance de lignes paires et de lignes impaires, il y aura des champs pairs et des champs impairs (seulement en mode entrelacé). En NTSC, nous avons 262 champs, donc 263 lignes; en PAL, 312 champs, donc 313 lignes.

Cette alternance de lignes et de champs produit le modèle répétitif 4 champs du standard NTSC suivant :

fin du champ impair sur une ligne impaire
fin du champ pair sur une ligne paire
fin du champ impair sur une ligne paire
fin du champ pair sur une ligne impaire
et retour au début...

Un calcul horizontal prend un cycle d'horloge au système. (Le processeur prend deux fois ce temps.)

NTSC - 3,579,545 Hertz
PAL - 3,546,895 Hertz
"genlocké" - fréquence de base de l'horloge plus ou moins 2% environ.

LES BITS DE VALIDATION DE COMPARAISON

Normalement les bits 14-1 sont tous mis à 1. L'emploi des bits de validation d'une comparaison sera décrit plus loin dans la section "Sujets Avancés".

UTILISATION DES REGISTRES DU COPPER

Le Copper possède plusieurs registres machines et plusieurs adresses 'strobe' spécialisées :

- o Registres d'adresse
- o 'Strobes' [1] pour les adresses de saut
- o Registre de contrôle

REGISTRES D'ADRESSE

Le Copper possède deux jeux de registres d'adresse :

COP1LCH Les 3 bits de poids fort de l'adresse de la première liste Copper.
COP1LCL Les 15 bits de poids faible de l'adresse de la première liste Copper.
COP2LCH Les 3 bits de poids fort de l'adresse de la deuxième liste Copper.
COP2LCL Les 15 bits de poids faible de l'adresse de la deuxième liste Copper.

En accédant directement au 'hardware', vous aurez souvent besoin d'écrire dans une paire de registres contenant l'adresse des données. Le registre d'adresse inférieur a toujours un nom qui se termine par "H" et contient les données de poids fort, c'est-à-dire les trois bits de poids fort de l'adresse. Le registre d'adresse supérieur a un nom qui se termine par "L" et contient les données de poids faible, c'est-à-dire les 15 bits de poids faible de l'adresse. Par conséquent, vous écrirez une adresse de 18 bits en transférant un mot long dans le registre dont le nom se termine par "H". On procède ainsi parce que -quand on écrit des mots longs avec le 68000-, le mot le plus significatif va dans un mot d'adresse inférieure.

Dans le cas des registres d'adresse du Copper, vous écrirez l'adresse dans COP1LCH. Par la suite, pour simplifier, nous ferons référence à ces adresses en les appelant COP1LC ou COP2LC.

Les registres d'adresse du Copper contiennent les deux adresses indirectes de saut utilisées par le Copper lui-même. Celui-ci prélève ses propres instructions en employant son compteur de programme et augmente ce dernier d'un pas après chaque prélèvement. Dès qu'un 'strobe' pour l'adresse de saut est écrit, le registre d'adresse correspondant est chargé à son tour dans le compteur de programme du Copper. Ceci oblige le Copper à sauter à une nouvelle adresse, d'où il prélèvera l'instruction suivante. Ce prélèvement continuera séquentiellement jusqu'à ce que le Copper soit interrompu par un autre 'strobe' pour l'adresse de saut.

[1] Il s'agit de valeurs qui ne seront pas écrites dans un registre, mais qui serviront à déclencher une action particulière (NdT).

NOTE

Au début de chaque intervalle de suppression verticale, COP1LC est utilisé automatiquement pour faire démarrer le compteur de programme. Ceci se produit parce que -sans égard pour ce que le Copper est en train de faire- à la fin de la suppression verticale, le Copper est forcé automatiquement à recommencer ses opérations à l'adresse contenue dans COP1LC.

ADRESSE 'STROBE' DE SAUT

Quand on écrit dans une adresse 'strobe' du Copper, ce dernier recharge son compteur de programme depuis le registre d'adresse correspondant. Le Copper a la possibilité d'écrire dans ses propres registres d'adresse et dans ses adresses 'strobe' pour exécuter ensuite des sauts programmés. Par exemple, on pourrait TRANSFERER (MOVE) une adresse indirecte dans le registre d'adresse COP2LC. Ensuite, toute instruction MOVE -qui adresserait COPJMP2- enverrait cette adresse indirecte dans le compteur de programme.

Il y a deux deux adresses 'strobe' de saut :

COPJMP1 Relance le Copper à partir de l'adresse contenue dans COP1LC.
COPJMP2 Relance le Copper à partir de l'adresse contenue dans COP2LC.

LE REGISTRE DE CONTROLE

Le Copper peut avoir accès à certains registres spécialisés à n'importe quel moment à condition d'avoir mis à 1 le bit spécial de contrôle et seulement à certains registres, non à tous. Ces registres qui peuvent toujours être affectés par le Copper sont numérotés de $20 à $FF inclus. (Cf. l'Appendice B pour la liste des registres par ordre d'adresse.) Le registre de contrôle du Copper fait partie de ce groupe (de $00 à $0F). Ainsi, celui-ci entreprend une action délibérée, avec la participation du 68000, pour permettre au Copper d'écrire dans un intervalle déterminé des registres spécialisés.

Le registre de contrôle du Copper, COPCON, contient un seul bit. Celui-ci s'appelle CDANG (ce qui signifie Bit de Danger du Copper) et protège tous les registres compris entre $10 à $1F inclus. Nous trouvons dans cet intervalle les registres de contrôle du 'blitter'. Lorsque CDANG est mis à 0, le Copper n'a pas le droit d'écrire dans ces registres. Lorsque CDANG est mis à 1, le Copper peut y écrire. En empêchant le Copper d'avoir accès aux registres de contrôle du 'blitter', ce bit empêche qu'un Copper "emballé" (à cause d'une liste d'instructions mal construite) affecte accidentellement la mémoire du système.

NOTE

Le bit CDANG est remis à zéro après une réinitialisation.

REALISATION D'UNE LISTE D'INSTRUCTIONS DU COPPER

La liste d'instructions du Copper comprend la remise à zéro de tous les registres, remise à zéro qui s'effectue pendant l'intervalle de suppression verticale, ainsi que les modifications des registres nécessaires pour réaliser des altérations au milieu de l'écran. Dès que vous faites des projets sur ce qui pourra intervenir dans une zone d'affichage, vous pourriez trouver qu'il est plus simple de s'imaginer chaque partie d'un affichage comme un sous-système indépendant, tel que des aires de jeu, des sprites, de l'audio, des interruptions et ainsi de suite. Vous pourriez ensuite écrire une liste séparée des choses à faire individuellement pour chaque position du faisceau vidéo dans chaque sous-système.

Dès que vous aurez créé toutes ces listes intermédiaires de choses à faire, vous devrez les fusionner ensemble dans une seule liste d'instructions que le Copper exécutera une seule fois à chaque trame de l'affichage. Une solution alternative est de créer cette liste qui inclut tout ce que nous venons d'énumérer, sans devoir passer par des étapes intermédiaires.

Par exemple, les pointeurs des plans de bits utilisés dans les affichages d'aires de jeu ('playfield') et les pointeurs des sprites ('sprite') devront être redessinés pendant l'intervalle de suppression verticale pour que les données puissent être saisies correctement lorsque l'affichage de l'écran reprendra. Ceci peut être réalisé par une liste d'instructions Copper qui fait ce qui suit :

```
WAIT [ATTENDRE] jusqu'à la première ligne de l'affichage
MOVE [TRANSFERER] les données vers le pointeur du 1er plan de bits
MOVE [TRANSFERER] les données vers le pointeur du 2e plan de bits
MOVE [TRANSFERER] les données vers le pointeur du 1er sprite
et ainsi de suite
```

Comme autre exemple, les canaux DMA des sprites -qui créent des objets qu'on peut déplacer- peuvent être réemployés plusieurs fois dans le même champ d'affichage. Vous avez la possibilité de modifier la taille et la forme d'un sprite à chaque nouvelle utilisation; cependant, normalement, chaque nouvelle utilisation multiple utilise le même jeu de couleurs pendant tout un champ d'affichage. Vous pouvez changer les couleurs du sprite au milieu de l'écran par une liste d'instructions Copper qui attend jusqu'à la dernière ligne de la première utilisation du processeur de sprites et change les couleurs avant la prochaine utilisation de ce même processeur :

```
WAIT [ATTENDRE] la première ligne de l'affichage
MOVE [TRANSFERER] PremièreCouleurNo1 dans COLOR 17
MOVE [TRANSFERER] PremièreCouleurNo2 dans COLOR 18
MOVE [TRANSFERER] PremièreCouleurNo3 dans COLOR 19
WAIT [ATTENDRE] la derniere ligne+1 du premier emploi du sprite
MOVE [TRANSFERER] DeuxièmeCouleurNo1 dans COLOR 17
MOVE [TRANSFERER] DeuxièmeCouleurNo2 dans COLOR 18
MOVE [TRANSFERER] DeuxièmeCouleurNo3 dans COLOR 19
et ainsi de suite
```

En créant les listes d'instructions Copper, remarquez que la liste finale doit se trouver dans le même ordre que celui utilisé par le faisceau vidéo pour créer l'affichage. Le faisceau vidéo traverse l'écran de la position (0,0) dans le coin supérieur gauche de l'écran à la fin de l'affichage (226,262) en NTSC (ou (226,312) en PAL) dans le coin inférieur droit. Le premier 0 de (0,0) représente la position des x. Le deuxième 0 représente la position des y. Par exemple, une instruction qui ne fait rien à la position (0,100) devra venir après une instruction qui affecte l'affichage à la position (0,60).

NOTE

A cause de la forme de l'instruction WAIT, vous pourrez parfois "sortir" sans arriver à mettre en place une liste exactement dans l'ordre du faisceau vidéo. L'instruction WAIT oblige le Copper à attendre tant que la valeur du compteur de faisceau n'est pas égale ou supérieure à celle de la valeur de l'instruction.

Ceci signifie que, par exemple, si vous avez des instructions qui se suivent les unes après les autres comme ci-dessous :

```
WAIT  la position (64,64)
MOVE  les données
WAIT  la position (60,60)
MOVE  les données
```

Le Copper exécutera les deux transferts de données, même si les instructions se trouvent en dehors de la séquence. La spécification "supérieure à" évite que le Copper se bloque si le faisceau a déjà dépassé la position indiquée. Un effet secondaire est constitué par le fait que le second MOVE ci-dessous est exécuté :

```
WAIT  la position (60,60)
MOVE  les données
WAIT  la position (60,60)
MOVE  les données
```

Au moment du second WAIT de la séquence, les compteurs de faisceau auront une valeur supérieure à celle de la position que nous indiquons dans les instructions. Par conséquent, le deuxième MOVE sera exécuté à son tour.

Notez aussi que la séquence d'instructions ci-dessus pourrait facilement s'écrire ainsi

```
WAIT  la position (60,60)
MOVE  les données
MOVE  les données
```

car plusieurs MOVE peuvent se trouver après un seul WAIT.

EXEMPLE DE LISTE COPPER COMPLETE

L'exemple qui suit montre une liste Copper complète. La liste est construite pour deux plans de bits, l'un situé à $21000 et l'autre situé à $25000. En haut de l'écran, les registres de couleur seront chargés avec les valeurs suivantes :

REGISTRE	COULEUR
COLOR00	blanc
COLOR01	rouge
COLOR02	vert
COLOR03	bleu

A la ligne 150 de l'écran, les registres de couleur seront rechargés :

REGISTRE	COULEUR
COLOR00	noir
COLOR01	jaune
COLOR02	cyan
COLOR03	magenta

Suit la liste Copper complète.

```
;
; Notes:
;       1. Les listes Copper doivent se trouver en mémoire CHIP.
;       2. Les adresses des plans de bits employées dans cet exemple sont
;          arbitraires.
;       3. Les adresses des registres de destination des instructions 'move'
;          du Copper prennent leurs 'offsets' dans l'adresse de base des
;          puces spécialisées.
;       4. Comme toujours, les exemples 'hardware' de ce manuel présument
;          qu'une application a pris le plein contrôle du 'hardware' et
;          qu'elle n'est pas en conflit avec l'usage que fait le système
;          d'exploitation de ce même 'hardware'.
;       5. Beaucoup d'exemples se limitent à choisir des adresses en mémoire
;          pour s'en servir. Normalement, vous auriez besoin de réserver le
;          type de mémoire dont a besoin le système par AllocMem().
;       6. Comme nous l'avons affirmé précédemment, les exemples sont
;          principalement destinés à vous aider à clarifier de quelle façon
;          fonctionne le 'hardware'.
;       7. Tous les exemples de ce chapitre ont besoin des fichiers INCLUDE
;          suivants.
;

          INCLUDE "exec/types.i"
          INCLUDE "hardware/custom.i"
          INCLUDE "hardware/dmabits.i"
          INCLUDE "hardware/hw_examples.i"
```

```
COPPERLIST:
;
; Mettre en place les pointeurs sur les deux plans de bits
;
    DC.W    BPL1PTH,$0002      ;Transfère S0002 dans le registre $0E0 (BPL1PTH)
    DC.W    BPL1PTL,$1000      ;Transfère $1000 dans le registre $0E2 (BPL1PTL)
    DC.W    BPL2PTH,$0002      ;Transfère $0002 dans le registre $0E4 (BPL2PTH)
    DC.W    BPL2PTL,$5000      ;Transfère $5000 dans le registre $0E6  (BPL2PTL)
;
; Charger les registres de couleur
;
    DC.W    COLOR00,$0FFF      ;Transfère le blanc dans le registre $180 (COLOR00)
    DC.W    COLOR01,$0F00      ;Transfère le rouge dans le registre $182 (COLOR01)
    DC.W    COLOR02,$00F0      ;Transfère le vert dans le registre $189 (COLOR02)
    DC.W    COLOR03,$000F      ;Transfère le bleu dans le registre $186 (COLOR03)
;
; Indiquer 2 plans de bits en basse résolution
;
    DC.W    BPLCON0,$2200      ;2 plans en basse résolution, en couleur
;
; Attendre la ligne 150
;
    DC.W    $9601,$FF00            ;Attendre la ligne 150, ignorer la position horizontale
;
; Modifier les registres de couleur au milieu de l'écran
;
    DC.W    COLOR00,$0000      ;Transfère le noir dans le registre $0180 (COLOR00)
    DC.W    COLOR01,$0FF0      ;Transfère le jaune dans le registre $0182 (COLOR01)
    DC.W    COLOR02,$00FF      ;Transfère le cyan dans le registre $0184 (COLOR02)
    DC.W    COLOR03,$0F0F      ;Transfère le magenta dans le registre $0186 (COLOR03)
;
;  Fin de la liste du Copper en attendant l'impossible
;
    DC.W    $FFFF,$FFFE            ;Attendre la ligne 255, H = 254 (ce qui ne se produit jamais)
```

Pour plus de renseignements sur les registres de couleur, voir le Chapitre3, "Le 'Hardware' des Aires de Jeu".

BOUCLES ET BRANCHEMENTS

Les boucles et les branchements dans une liste Copper seront traités plus loin, dans la section "Sujets Avancés".

DEMARRAGE ET ARRET DU COPPER

DEMARRAGE DU COPPER APRES UNE REINITIALISATION

A l'allumage ou lors d'une réinitialisation, vous devez initialiser un des registres d'adresse du Copper (COP1LC ou COP2LC) et écrire dans ses adresses 'strobe' avant que le DMA du Copper se mette en route. Cette action permet d'avoir une adresse de démarrage et un état connus d'avance.
Habituellement, on utilise COP1LC parce que ce registre particulier est réemployé pendant chaque intervalle de suppression verticale.
La séquence d'instructions qui suit indique comment initialiser un registre d'adresse. Nous présumons que l'utilisateur a déjà créé une liste d'instructions Copper correcte à l'adresse "MaListeCop".

```
;  Installer la liste Copper
;
    LEA    CUSTOM,a1                ; a1 = adresse des puces spécialisées
    LEA    MALISTECOP (pc),a0   ; Adresse de notre liste Copper
    MOVE.L a0,COP1LC(a1)        ; Ecrire l'adresse en mots longs
    MOVE.W COPJMP1(a1),d0      ; Oblige le Copper à charger le PC à partir de COP1LC
;
; Puis mettre en service le Copper et le 'dma' de trame
;
    MOVE.W  #(DMAF SETCLR!DMAF_COPPER!DMAF_RASTER!DMAF_MASTER),DMACON(a1)
;
```

Maintenant, si le contenu de COP1LC n'a pas été modifié, à chaque fois qu'interviendra la suppression verticale, le Copper recommencera à partir de la même adresse pour chaque écran vidéo ultérieur. Ceci créera une boucle répétitive qui -si on a construit la liste correctement- permettra d'avoir un écran d'affichage stable.

ARRET DU COPPER

Le Copper n'a pas d'instruction pour l'arrêt. Pour garantir qu'il s'arrête et qu'il ne fasse plus rien jusqu'à la fin de l'affichage de l'écran et jusqu'à ce que le compteur du programme démarre à nouveau au début de la liste d'instructions, la dernière instruction devrait ATTENDRE (WAIT) un événement qui ne pourra se produire. Une instruction typique est celle d'ATTENDRE (WAIT) VP = $FF et HP = $FE. Il n'est pas possible d'avoir un HP supérieur à $E2. Lorsque l'affichage de l'écran prend fin et démarre la suppression verticale, le Copper sera automatiquement pointé vers le début de sa liste d'instructions et cette instruction WAIT finale n'en permettra jamais l'arrêt.

Vous avez aussi la possibilité d'arrêter le Copper en l'empêchant de se servir des DMA pour recevoir des instructions ou pour déplacer des données. Le registre DMACON contrôle tous les canaux DMA. Le bit 7, COPEN, lorsqu'il est à 1, permet au Copper d'accéder aux DMA.

Pour des renseignements sur le contrôle des DMA, voir le Chapitre 7, "Le 'Hardware' de Contrôle du Système".

L'INSTRUCTION SKIP

L'instruction SKIP oblige le Copper à sauter à l'adresse suivante si les compteurs de faisceau vidéo sont égaux ou supérieurs à la valeur donnée dans l'instruction.

Nous donnons ci-dessous le contenu des mots de l'instruction SKIP. Ce contenu est identique à celui de l'instruction WAIT, si l'on excepte que le bit 0 du deuxième mot de l'instruction est mis à 1 pour pouvoir reconnaître cette instruction en tant qu'instruction SKIP.

PREMIER MOT DE L'INSTRUCTION (IR1)

Bit 0	Toujours mis à 1.
Bits 15 - 8	Position verticale (qu'on appelle VP).
Bits 7 - 1	Position horizontale (qu'on appelle HP). Sauter si le compteur de faisceau vidéo est égal ou supérieur à ces deux séries de bits ensemble (bits 15 à 1).

DEUXIEME MOT DE L'INSTRUCTION (IR2)

Bit 0	Toujours mis à 1.
Bit 15	Le bit de mise hors service définitive du 'blitter'. (Cf. ci-dessous "Utilisation du Copper avec le 'Blitter'").
Bits 14 - 8	Bits de mise en service pour comparaison avec la position verticale (qu'on appelle VE).
Bits 7 - 1	Bits de mise en service pour comparaison avec la position horizontale (qu'on appelle HE).

Les notes sur la position horizontale et verticale du faisceau, qu'on trouve dans la partie qui traite l'instruction WAIT, s'appliquent aussi à l'instruction SKIP.

L'instruction SKIP de l'exemple qui suit saute l'instruction suivante si VE (position verticale du faisceau) est supérieur ou égal à 100 ($64).

```
DC.W   $6401,$FF01    ; Si VP >= 100, sauter l'instruction suivante (ignorer HP)
```

BOUCLES ET BRANCHEMENTS DU COPPER ET VALIDATION D'UNE COMPARAISON

Vous avez la possibilité de modifier à chaque instant la valeur contenue dans les registres d'adresse et d'utiliser cette valeur pour construire des boucles dans la liste des instructions. Cependant, avant l'intervalle suivant de suppression verticale, on devra pointer à nouveau les registres COP1LC au début de la liste Copper appropriée. La valeur des registres d'adresse COP1LC sera restaurée dans le compteur de programme du Copper au début de l'intervalle de suppression verticale.

Les bits 14-1 du deuxième mot d'instruction des instructions SKIP et WAIT indiquent quels bits de position verticale et horizontale devront être employés pour la comparaison avec le compteur de faisceau. La position dans le premier mot et les bits de validation présents dans le deuxième mot de l'instruction sont testés avec la valeur réelle des compteurs de faisceau avant d'entreprendre une action quelconque. Lors de la comparaison des positions avec les valeurs réelles des compteurs, on utilise un bit de position du premier mot de l'instruction, si, et seulement si, le bit de validation correspondant du deuxième mot de l'instruction est mis à 1. Si le bit de validation correspondant est à 0, la comparaison sera toujours vraie. Par exemple, si vous vous intéressez seulement à la valeur des quatre derniers bits de la position verticale, vous ne devrez employer que les quatre derniers bits de validation de comparaison, c'est-à-dire les bits (11-8) du deuxième mot de l'instruction.

Les bits du compteur de faisceau ne peuvent pas tous être masqués. Si vous regardez la description du IR2 (deuxième mot de l'instruction), vous vous apercevrez que le bit 15 est le bit de mise hors service définitive du 'blitter'. Ce bit ne fait pas partie du masque de comparaison du compteur de faisceau, il a sa propre signification à l'intérieur de l'instruction WAIT du Copper. Ainsi, vous n'avez pas la possibilité de masquer le bit de poids fort des instructions SKIP ou WAIT. Dans la plupart des cas, cette restriction n'entre pas en jeu, cependant, l'exemple suivant vous montrera le traitement qu'on on devra appliquer à ce bit.

Cet exemple apprend au Copper à provoquer une interruption toutes les 16 lignes de balayage. On pourrait penser qu'une manière de provoquer cette interruption serait celle d'utiliser un masque d'une valeur $0F et de comparer ensuite le résultat avec $0F. Le résultat de la comparaison serait "vrai" pour $1F, $2F, $3F, etc.. Etant donné que le test a lieu pour une valeur supérieure ou égale, cette manière de faire semblerait pouvoir permettre une vérification toutes les seize lignes de balayage. Cependant, le bit prioritaire ne peut être masqué, par conséquent, celui-ci apparaîtra toujours dans les comparaisons. Quand le Copper attend $0F et que la position verticale a dépassé 128 ($80, en hexadécimal), ce type de test sera toujours vrai. Dans ce cas, la valeur minimale, lors de la comparaison, sera $80 valeur qui est toujours supérieure à $0F et donc l'interruption aura lieu à chaque ligne de balayage. Souvenez-vous-en, le Copper ne vérifie que pour des valeurs supérieures ou égales.

Dans l'exemple qui suit, les listes Copper ont été faites pour boucler. Les valeurs de COP1LC et de COP2LC sont mises en route par le CPU ou dans une liste Copper avant cette section de code du Copper. De plus, nous présumons que vous avez installé correctement un serveur d'interruption pour l'interruption Copper qui sera générée toutes les seize lignes. Remarquez que ces lignes de balayage sont en mode non entrelacé.

MODE DE FOCTIONNEMENT :

En gros, les deux boucles sont exactement identiques. Dans l'une, le Copper attend jusqu'à ce que le registre de position verticale contienne une valeur $?F (? est un chiffre hexadécimal quelconque), à ce moment nous provoquons une interruption Copper dans le 'hardware' de l'Amiga. Pour s'assurer que le Copper ne boucle pas en arrière avant que la position verticale n'ait changé et qu'il ne provoque pas une autre interruption sur la même ligne de balayage, attendez que la position horizontale soit $E2 après chaque interruption.
Pour le Copper, la position $E2 est la position horizontale 113 et celle-ci est aussi l a dernière position horizontale effectivement disponible. Cette action forcera le Copper à aller à la ligne suivante dans l'attente de l'instruction WAIT suivante.
Ce qui oblige le Copper à sauter à l'adresse qu'on a initialisée dans COP1LC.

Le problème de masque, décrit plus haut, fera que le programme échoue après chaque position verticale 127. Une boucle séparée devra être exécutée quand la position verticale sera supérieure ou égale à 127. Lorsque cette position verticale deviendra supérieure ou égale à 127, alors la première instruction de boucle sera sautée, en envoyant le Copper dans la deuxième boucle. Celle-ci est, en gros, identique à la première, si l'on excepte qu'elle attend $?F avec le bit prioritaire mis à 1 (1xxx1111, en binaire).
Ce bit sera vrai pour les deux instructions WAIT, pour celle verticale et pour celle horizontale.
Pour déclencher la deuxième boucle, écrivez dans le registre COPJMP2. La liste sera mise en attente à l'infini quand VP >= 255, par conséquent, celle-ci prendra fin avant une suppression verticale. A la fin de la période de suppression verticale, le système d'exploitation écrira à son tour dans COP1LC, en provoquant à nouveau l'exécution de la première boucle.

NOTE

C'est un gestionnaire d'interruption graphique - qui se trouve dans la chaîne des serveurs d'interruption pendant la période de suppression verticale – qui écrit dans le registre COP1LC à la fin de l'intervalle de suppression verticale.
Tant que ce serveur restera intact COP1LC se déclenchera de façon correcte lors de chaque suppression verticale.

```
;
; Celles-ci sont les données de la liste Copper.
;
; Nous présumons que COPPERL1 a été chargé dans COP1LC et que
; COPPERL2 a été chargé dans COP2LC par une autre partie du programme.
;
COPPERL1:
    DC.W    $0F01,$8F00       ; Attendre VP=0xxxllll
    DC.W    INTREQ,$8010      ; Mettre à 1 le bit d'interruption du Copper

    DC.W    $00E3,$80FE       ; Attendre la position horizontale $E2
                              ; Nous mettons ceci au cas où la ligne
                              ; prendrait fin avant que l'on ne puisse
                              ; vérifier que nous y sommes déjà (le 'wait' ci-dessus)

    DC.W    $7F01,$7F01       ; Sauter si VP>=127
    DC.W    COPJMP1,$0        ; Forcer le saut dans COP1LC

COPPERL2:
    DC.W    $8F01,$8F00       ; Attendre Vp=1xxx1111
    DC.W    INTREQ,$8010      ; Mettre à 1 le bit d'interruption du Copper...

    DC.W    $80E3,$80FE       ; Attendre la position horizontale $E2
                              ; Nous mettons ceci au cas où la ligne
                              ; prendrait fin avant que l'on ne puisse
                              ; vérifier que nous y sommes déjà (le 'wait' ci-dessus)

    DC.W    $FF01, $FE01      ; Sauter si VP>=255

    DC.W    COPJMP2,$0        ; Forcer le saut dans COP2LC
```



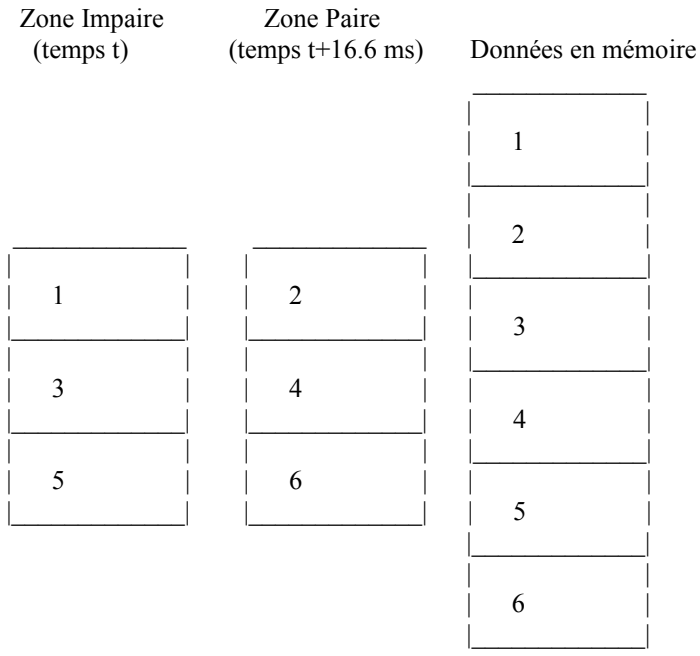
```
; Mettre ici tout ce qui serait nécessaire pour nettoyer le programme Copper
```

```
; Comme nous avons 262 lignes en NTSC, et que nous nous sommes arrêtés à la
; ligne 255, nous disposons ici d'un bit de temps

    DC.W    $FFFF,$FFFE     ; Fin de la liste Copper
;
```

UTILISATION DU COPPER EN MODE ENTRELACE

A l'écran, l'affichage d'un plan de bits en mode entrelacé a un nombre de lignes verticales double par rapport à un plan normal. Là où un affichage normal en NTSC possède 262 lignes, un affichage en NTSC en mode entrelacé en aura 524. Le PAL a normalement 312 lignes et 625 en mode entrelacé. Dans ce mode, le faisceau vidéo balaie l'écran deux fois de haut en bas, en affichant, dans le cas du NTSC, 262 lignes à chaque fois. Lors du premier balayage, seront affichées les lignes de numéro impair. Lors du deuxième, seront affichées les lignes de numéro pair, puis ces dernières seront entrelacées avec les lignes de numéro impair. Ainsi, les circuits de balayage traitent un affichage entrelacé comme deux zones d'affichage, la première contenant les lignes paires et la seconde les lignes impaires. Le DESSIN 2-1 montre comment est stocké en mémoire un affichage en mode entrelacé.



DESSIN 2-1 : (Plan de Bits entrelacé en RAM)

Le système interroge les données des plans de bits en mémoire en utilisant des pointeurs sur l'adresse initiale des données. Comme vous pouvez le voir, l'adresse initiale pour les zones de numéro pair a une ligne de plus que celle des zones de numéro impair. Cependant, le pointeur du plan de bits doit contenir une valeur différente dans le cas des zones alternées d'un affichage entrelacé.

Tout simplement, l'organisation des données en mémoire concorde avec l'organisation évidente des données à l'écran (par ex., les lignes impaires et les lignes paires sont entrelacées les unes avec les autres). Il est possible de faire cela en ayant une liste d'instructions Copper séparée pour chaque zone pour pouvoir gérer l'affichage des données.

Pour pousser le Copper à traiter la liste correcte, vous mettez en place une interruption dans le 68000 juste après la première ligne de balayage. Quand l'interruption se déclenche, vous modifiez le contenu du registre d'adresse COP1LC pour qu'il pointe sur la deuxième liste. Ensuite, pendant la période de suppression verticale, COP1LC sera initialisé automatiquement pour pointer sur la liste originale.

Pour plus de renseignements sur les affichages entrelacés, voyez le Chapitre 3, "Le 'Hardware' des Aires de Jeu".

UTILISATION DU COPPER AVEC LE BLITTER

Si on utilise le Copper pour lancer une séquence d'opérations du 'blitter', le Copper devra attendre l'interruption définitive du 'blitter' avant de démarrer une nouvelle opération sur le 'blitter'. Si l'on modifie les registres du 'blitter', pendant que celui-ci est en train de travailler, on peut provoquer des effets imprévisibles. C'est pour cela que l'instruction WAIT possède un bit de contrôle supplémentaire, BFD (ce qui signifie mise hors service du 'blitter'). Normalement ce bit est à 1 et seules les comparaisons du compteur de faisceau contrôlent le WAIT.

Quand le bit BFD se trouve à 0, la logique de l'instruction WAIT du Copper est modifiée. Le Copper ATTENDRA (WAIT) jusqu'à ce que la comparaison du compteur de faisceau soit vraie et que le 'blitter' ait fini. Le 'blitter' a fini lorsque le drapeau de mise hors service est activé. Ce bit devrait être réinitialisé avec précaution. Il pourrait éventuellement empêcher certains affichages d'écran ou empêcher que les objets soient affichés correctement.

Pour plus de renseignements sur l'emploi du 'blitter', voyez le Chapitre 6, "Le 'Hardware' du Blitter".

LE COPPER ET LE 68000

Dans les occasions où les instructions du Copper ne sont pas suffisantes, vous avez la possibilité d'interrompre le 68000 et d'utiliser, à la place, son jeu d'instructions. Le 68000 peut interroger les 'flags' d'interruption présents dans le registre INTREQ par l'intermédiaire de plusieurs périphériques. Pour interrompre le 68000, employez l'instruction MOVE du Copper pour mettre un 1 dans les bits d'INTREQ ci-dessous :

Tableau 2-1 : Pour interrompre le 68000

NUMERO DE BIT	NOM	FONCTION
15	SET/CLR	Bit de contrôle Set/Clear. Détermine si les bits écrits avec un 1 seront validés ou remis à zéro.
4	COPEN	Co-processeur pour interrompre le 68000.

Voir le Chapitre 7, "'Hardware' de Contrôle du Système", pour plus de renseignements sur les interruptions.

RECAPITULATION DES INSTRUCTIONS DU COPPER

Le tableau ci-dessous montre une récapitulation des positions des bits pour chacune des instructions du Copper. Cf. l'Appendice A pour la récapitulation de tous les registres.

Tableau 2-2 : Récapitulation des Instructions du Copper

	Move		**Wait**		**Skip**	
Bit#	IR1	IR2	IR1	IR2	IR1	IR2
15	x	RD15	VP7	BFD	VP7	BFD
14	x	RD14	VP6	VE6	VP6	VE6
13	x	RD13	VP5	VE5	VP5	VE5
12	x	RD12	VP4	VE4	VP4	VE4
11	x	RD11	VP3	VE3	VP3	VE3
10	x	RD10	VP2	VE2	VP2	VE2
09	x	RD09	VP1	VE1	VP1	VE1
08	DA8	RD08	VP0	VE0	VP0	VE0
07	DA7	RD07	HP8	HE8	HP8	HE8
06	DA6	RD06	HP7	HE7	HP7	HE7
05	DA5	RD05	HP6	HE6	HP6	HE6
04	DA4	RD04	HP5	HE5	HP5	HE5
03	DA3	RD03	HP4	HE4	HP4	HE4
02	DA2	RD02	HP3	HE3	HP3	HE3
01	DA1	RD01	HP2	HE2	HP2	HE2
00	0	RD00	1	0	1	1

x = ne vous en occupez pas, mais devrait être égal à 0 pour une compatibilité ascendante
IR1 = premier mot de l'instruction
IR2 = deuxième mot de l'instruction
DA = adresse de destination
RD = données en RAM à transférer dans un registre de destination
VP = bit de position verticale du faisceau
HP = bit de position horizontale du faisceau
VE = valide la comparaison (bit de masque)
HE = valide la comparaison (bit de masque)
BFD = mise hors service définitive du 'blitter'

Chapitre 3

LES 'PLAYFIELDS HARDWARE'

INTRODUCTION

L'affichage d'un écran est composé de deux parties principales : les aires de jeu, appelées parfois arrière-plans, et les sprites, qui sont des objets graphiques qu'on peut facilement déplacer. Ce chapitre décrit de quelle manière on peut accéder directement aux registres 'hardware' pour construire des aires de jeu.

AU SUJET DE CE CHAPITRE

Ce chapitre commence par un bref aperçu des caractéristiques des aires de jeu, y compris la définition de certains termes fondamentaux, et continue avec les sujets principaux suivants :

- Construction d'une seule aire de jeu "simple" : il s'agit d'une aire ayant des dimensions égales à celles de l'écran d'affichage. Cette section traite aussi de concepts qui sont fondamentaux lors de la construction d'une quelconque aire de jeu.

- Construction d'un affichage ayant une aire de jeu ['dual-playfield'] double, où chaque aire de jeu se superpose à l'autre. Cette procédure diffère -dans quelques détails- de celle qui permet de construire une aire de jeu de simple.

- Construction d'aires de jeu ayant des dimensions différentes, n'affichant qu'une partie d'une aire de jeu plus importante.

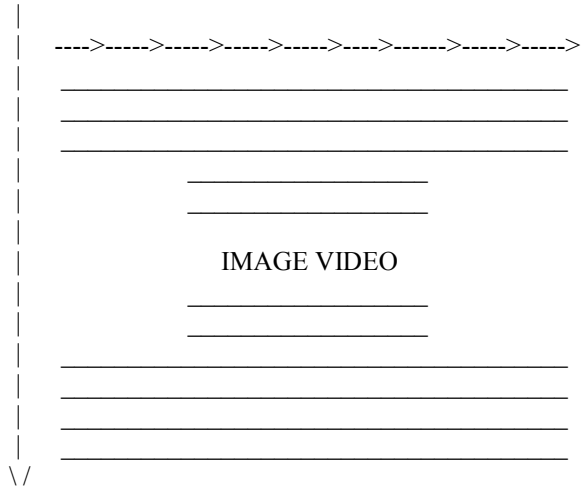
- Déplacement des aires de jeu par défilement vertical.

- Sujets avancés pour vous aider à utiliser les aires de jeu dans certaines situations particulières.

Pour des renseignements sur les objets sprites mobiles, voyez le Chapitre 4, "Le 'Hardware' des sprites". Il existe aussi des objets aires de jeu mobiles qui sont des sous-sections de l'aire de jeu. Pour déplacer des parties d'une aire de jeu, on emploie un technique particulière, l'animation des aires de jeu, que nous décrivons au Chapitre 6, "Le 'Hardware' du Blitter".

CARACTERISTIQUES DES AIRES DE JEU

L'Amiga construit ses affichages vidéo en faisant appel aux techniques de l'affichage par trame. L'image qu'on voit à l'écran est faite d'une suite de lignes vidéo horizontales qui s'affichent l'une après l'autre. Chaque ligne horizontale est faite d'une suite de pixels. On crée un affichage graphique en définissant un ou plusieurs plans de bits en mémoire et en les remplissant avec des "1" et des "0". La combinaison des "1" avec celle des "0" déterminera les couleurs de l'affichage.



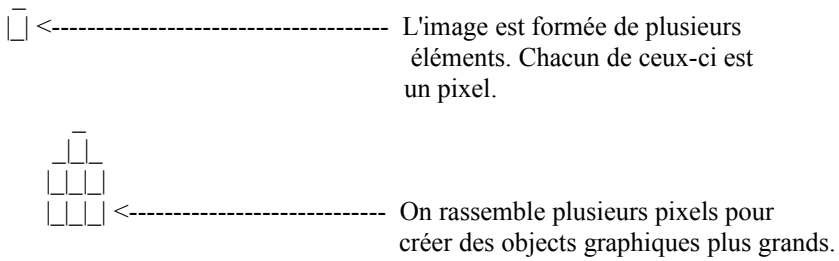
Chaque ligne représente un balayage du rayon électronique qui est en train de "peindre" l'image au fur et à mesure que celle-ci progresse.

Le faisceau vidéo crée chaque ligne en balayant de gauche à droite. Cette action construit tout l'écran, par balayage du faisceau de haut en bas, une seule ligne à la fois.

PLANCHE 3-1 : Création de l'Image Vidéo d'un Affichage

De haut en bas, le faisceau vidéo construit 262 lignes environ, dont 200 sont, normalement, dans un système NTSC, visibles à l'écran. Avec un système PAL, le faisceau construit 312 lignes, dont, normalement, 256 sont visibles. On appelle champ d'affichage, chaque jeu de lignes complet (262/NTSC ou 312/PAL). Le temps d'affichage, c'est-à-dire le temps nécessaire pour que le système produise un champ d'affichage, est approximativement égal à 1/60e de seconde dans un système NTSC et, approximativement, égal à 1/50e de seconde dans un système PAL. Entre les champs d'affichage, le faisceau vidéo traverse les lignes qui ne sont pas visibles à l'écran pour pouvoir créer un nouveau champ d'affichage.

On définit la zone d'affichage comme grille de pixels. Un pixel est un élément d'image unique, la plus petite partie adressable d'un affichage d'écran. Les dessins ci-dessous montrent ce qu'est un pixel et de quelle façon les pixels forment un écran.



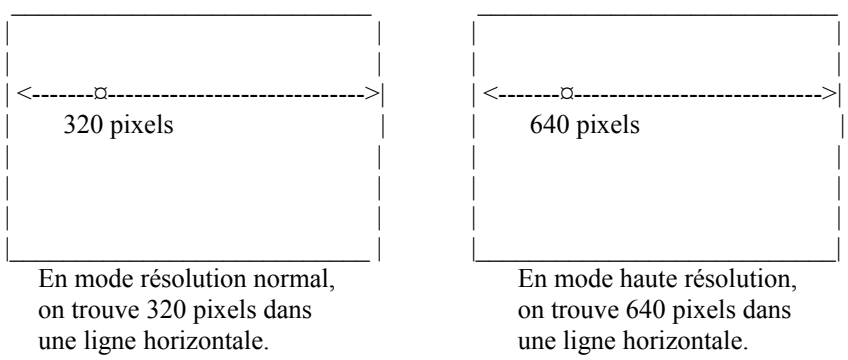


PLANCHE 3-2 : Ce qu'est un Pixel.

L'Amiga offre un choix, soit en résolution horizontale, soit en résolution verticale. On peut régler la définition horizontale pour travailler en mode basse ou haute résolution. On peut régler la définition verticale pour travailler en mode entrelacé ou non entrelacé.

- En mode basse résolution, la largeur d'une aire de jeu normale est égale à 320 pixels.

- Le mode haute résolution offre une résolution horizontale plus fine de 640 pixels pour une zone d'affichage physiquement identique.

- En mode non entrelacé, la hauteur d'une aire de jeu NTSC normale est égale à 200 lignes vidéo. La hauteur d'un écran PAL normal est de 256 lignes vidéo.

- Le mode entrelacé offre une résolution verticale plus fine de 400 lignes pour une zone d'affichage physiquement identique en NTSC et de 512 lignes en PAL.

Ces modes peuvent être combinés, ainsi on peut avoir, par exemple, un affichage en haute résolution, entrelacé.

Remarquez que les dimensions que nous mentionnons comme "normales", dans le paragraphe précèdent, sont des dimensions nominales et qu'elles représentent les valeurs normales auxquelles on peut s'attendre. En réalité, vous aurez la possibilité d'afficher des aires de jeu plus importantes; nous donnons les dimensions maximales dans la section "Plans de bits et Aires de Jeu de Toutes les Dimensions". En outre, en mémoire, les dimensions d'une aire de jeu sont souvent supérieures à celles de l'aire de jeu qui s'affiche à l'écran. On choisit la partie d'image en mémoire -image qui est plus grande- qu'on veut afficher en indiquant des dimensions différentes pour la fenêtre où se fera l'affichage.

On peut faire défiler ou déplacer lentement vers le haut ou vers le bas une aire de jeu plus grande que l'écran. On peut faire défiler horizontalement, de gauche à droite ou de droite à gauche une aire de jeu plus large que l'écran. Nous décrirons le déroulement ['scrolling'] dans la section "Déplacement (Déroulement) des Aires de Jeu".

Dans le système graphique de l'Amiga, quand on utilise les méthodes d'affichage normales, on a la possibilité d'avoir jusqu'à 32 couleurs différentes dans une seule aire de jeu. On pourra aussi contrôler individuellement la couleur de chaque pixel de l'affichage d'une aire de jeu en mettant à 1 le bit ou les bits qui contrôlent chaque pixel. Un affichage construit de cette façon est un affichage adressable par bit.

Par exemple, dans un affichage en deux couleurs, la couleur de chaque pixel est déterminée par le fait qu'un pixel est ou n'est pas activé. Si le bit est à 0, le pixel aura la couleur définie par l'utilisateur, si le bit est à 1, le pixel aura une autre couleur. Pour un affichage en quatre couleurs, on devra construire en mémoire deux plans de bits, en d'autres termes, chaque pixel aura alors une profondeur de deux bits. Vous pouvez combiner, de cette façon, jusqu'à cinq plans de bits. Les plans de bits constitués de trois, quatre ou cinq plans de bits permettront respectivement un choix de huit, seize ou trente-deux couleurs.

La couleur d'un pixel est déterminée toujours par une combinaison de bits binaire qui la définit. Lorsque le système combine les plans de bits d'un affichage, la combinaison de bits qu'on aura attribuée à chaque pixel correspondra au numéro d'un registre de couleur. Cette façon de procéder pour colorier les pixels s'appelle une indirection de couleur. L'Amiga a trente-deux registres de couleur, chacun contenant des bits qui définissent une couleur sélectionnée par l'utilisateur (sur les 4096 possibles).

La Planche 3-3 montre comment une combinaison pouvant aller jusqu'à cinq plans de bits composera un code qui sélectionne le registre à utiliser pour afficher la couleur d'un pixel d'une aire de jeu, sur les trente-deux possibles.

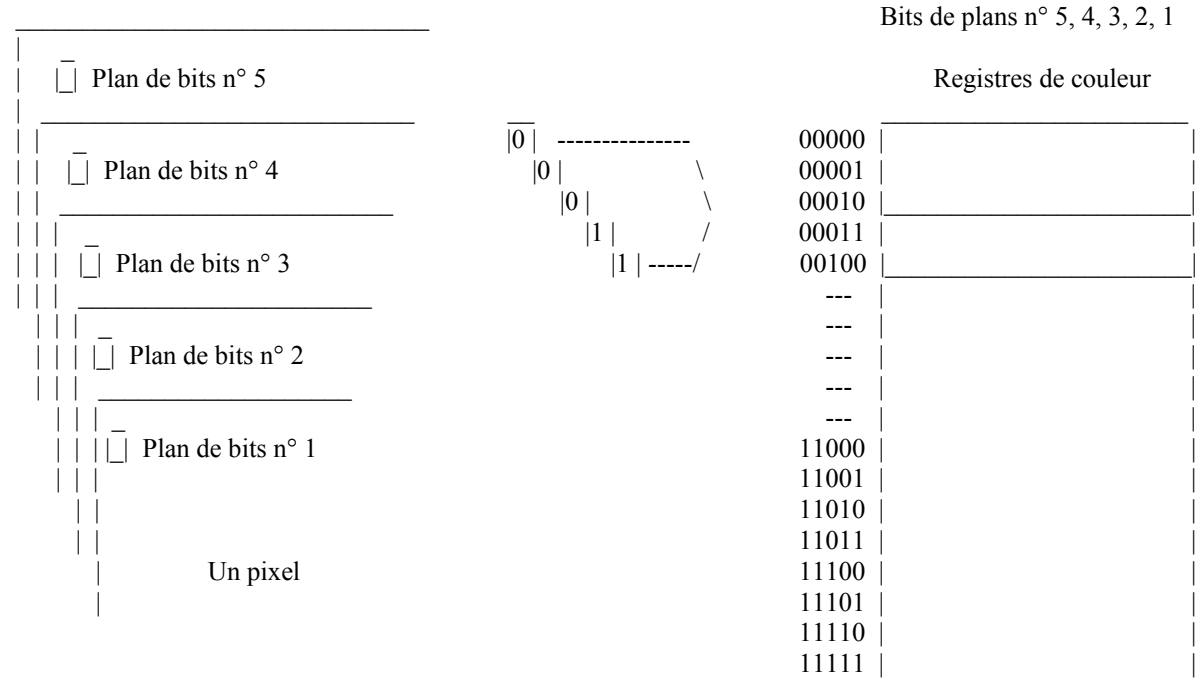


PLANCHE 3-3 : Comment les Plans de Bits Sélectionnent une Couleur

Les valeurs du plan de bits dont le numéro d'ordre est le plus important constitueront les valeurs de poids fort du nombre binaire. Comme nous le montrons dans la Planche 3-4, la valeur de chaque pixel du plan de bits dont le numéro d'ordre est le plus important constitue le chiffre le plus à gauche du nombre. La valeur du plan de bits suivant dont le numéro d'ordre sera le plus important constituera le bit suivant, et ainsi de suite.

Données d'exemple pour 4 pixels

```
A:  B:  C:  D:
|   |   |   |

1   1   0   0  ----  Données du Plan de Bits n° 5  (Poids Fort)
1   0   1   0  ----  Données du Plan de Bits n° 4
1   0   0   1  ----  Données du Plan de Bits n° 3
0   1   1   1  ----  Données du Plan de Bits n° 2
0   0   1   0  ----  Données du Plan de Bits n° 1  (Poids Faible)
```

A: Valeur n° 6 (COLOR6)
B: Valeur n° 11 (COLOR11)
C: Valeur n° 18 (COLOR18)
D: Valeur n° 28 (COLOR28)

PLANCHE 3-4 : Poids des Données des Plans de Bits lors d'une sélection de Couleur

En outre, vous avez la possibilité de choisir de définir deux aires de jeu séparément, chacune pouvant avoir jusqu'à trois plans de bits. Chacune de ces deux aires de jeu utilisera un jeu de huit couleurs différentes. C'est ce qu'on appelle le mode double aire de jeu ['dual-playfield'].

CONSTRUCTION D'UNE AIRE DE JEU SIMPLE

Pour commencer, cette section décrit de quelle manière on accède directement aux registres 'hardware' pour composer une seule aire de jeu simple ayant les mêmes dimensions que l'écran. Ici, l'expression "mêmes dimensions" veut dire que l'aire de jeu aura les mêmes dimensions que la fenêtre vidéo d'affichage courante. Cette composition laissera un petit filet entre l'aire de jeu et le bord de l'écran vidéo. Habituellement, une aire de jeu ne s'étend pas de toutes les façons jusqu'au bord de l'affichage physique.

Pour construire une aire de jeu, il faudra définir les caractéristiques suivantes :

- Hauteur et largeur de l'aire de jeu et dimensions de la fenêtre vidéo (autrement dit, quelle est la portion d'aire de jeu s'affichant réellement à l'écran).
- Couleur de chaque pixel de l'aire de jeu.
- Résolution horizontale.
- Résolution verticale, ou entrelacement.
- Accès aux données et à la valeur modulo, pour informer le système du nombre de données qu'il faudra placer dans une ligne horizontale et du nombre de données qu'il faudra prélever de la mémoire pour les envoyer à l'écran.

De plus, il faudra réserver de la mémoire pour stocker l'aire de jeu, mettre en place les pointeurs pour informer le système de l'endroit où se trouvent les données en mémoire et (en option) écrire une routine pour le Copper pour que celui-ci puisse gérer le nouveau dessin de l'aire de jeu.

HAUTEUR ET LARGEUR D'UNE AIRE DE JEU

Pour créer une aire de jeu ayant les mêmes dimensions que l'écran, on peut utiliser une largeur de 320 ou de 640 pixels en relation avec la résolution choisie. En NTSC, on peut avoir une hauteur de 200 ou de 400 lignes, en PAL de 256 ou 512, selon que l'on choisisse un mode entrelacé ou non.

PLANS DE BITS ET COULEUR

On définit la couleur d'une aire de jeu de la manière suivante :

1. En décidant du nombre de couleurs dont on aura besoin et en choisissant la couleur qu'on souhaite donner à chaque pixel.

2. En chargeant les couleurs dans les registres de couleur.

3. En réservant de la mémoire pour le nombre de plans de bits nécessaires et en mettant en place un pointeur pour chaque plan de bits.

4. En écrivant les instructions pour mettre une valeur dans chaque bit du plan en vue d'obtenir la bonne couleur.

Le Tableau 3-1 montre combien de plans de bits sont nécessaires pour une sélection de couleurs donnée.

Nombre de Couleurs	Nombre de Plans de Bits
1- 2	1
3- 4	2
5- 8	3
9-16	4
17-32	5

Tableau 3-1 : Les Couleurs d'une Aire de Jeu Simple.

LA TABLE DE COULEURS

La table de couleurs contient 32 registres et, dans chacun de ceux-ci, on peut charger une couleur différente. Voici une vue condensée de la table de couleurs :

Tableau 3-2 : Une Partie de la Table de Couleurs.

Nom du Registre	Contenu	Signification
COLOR00	12 bits	Couleur définie par l'utilisateur pour les bords et la section d'arrière-plan.
COLOR01	12 bits	Couleur n° 1 définie par l'utilisateur (par exemple, la sélection alternée de couleurs pour une aire de jeu avec deux couleurs).
COLOR02	12 bits	Couleur n° 2 définie par l'utilisateur.
.		
.		
.		
.		
.		
COLOR31	12 bits	Couleur n° 31 définie par l'utilisateur.

COLOR00 est toujours réservée à la couleur d'arrière-plan. Cette couleur est visible à l'affichage là où d'autres objets ne sont pas présents et s'affiche aussi à l'extérieur de la fenêtre d'affichage définie, sur la partie réservée aux bords.

NOTE

Si vous employez une carte pour le 'genlock', en option, pour des entrées vidéo à partir d'une caméra , d'un magnétoscope ou d'un vidéodisque, la couleur d'arrière-plan sera remplacée en entrée par celle de l'affichage vidéo.

Douze bits de sélection de couleur permettent de définir, pour chacun des 32 registres, une des 4096 couleurs possibles, comme indiqué dans le Tableau 3-3.

Table 3-3 : Contenu des Registres de Couleur

Bits	
Bits 15 -12	Inutilisés
Bits 11 - 8	Rouge
Bits 7 - 4	Vert
Bits 3 - 0	Bleu

Le Tableau 3-4 montre quelques affectations de bits dans les registres de couleur et les couleurs qui en résultent. Vous trouverez une liste plus importante a la fin du chapitre.

Tableau 3-4 : Extrait de Contenu des Registres de Couleur

Contenu du Registre de Couleur	Couleur Résultante
$fff	Blanc
$6fe	Bleu ciel
$db9	Ocre
$000	Noir

Nous donnons ci-dessous quelques instructions pour charger les registres de couleur :

```
LEA    CUSTOM,a0                ; Sauvegarder l'adresse de base des puces
                                ; spécialisées
MOVE.W #$FFF,COLOR00(a0)        ; Charger le blanc dans le registre de
                                ; couleur 0
MOVE.W #$6FE,COLOR01(a0)        ; Charger le bleu ciel dans le registre
                                ; de couleur 1
```

NOTE

Les registres de couleur s'emploient seulement en écriture. C'est seulement en regardant l'écran que vous pourrez voir le contenu de chaque registre de couleur. Donc, par une habitude routinière, pour ces registres et d'autres destinés seulement à l'écriture, on pourrait vouloir en garder une copie en RAM. Étant donné qu'on écrit dans les registres de couleur eux-mêmes, on devra penser à remettre à jour leur copie en mémoire. En faisant ainsi, on connaîtra toujours la valeur présente dans chaque registre.

SELECTION DU NOMBRE DE PLANS DE BITS

Après avoir décidé du nombre de couleurs qu'on souhaite avoir et du nombre de plans de bits nécessaires pour obtenir ces couleurs, on informera le système du nombre de plans de bits à utiliser.

On sélectionne le nombre de plans de bits en écrivant ce nombre dans le registre BPLCON0 (c'est-à-dire Registre de Contrôle des Plans de Bits n°0). Les bits pertinents sont les bits 14, 13 et 12, c'est-à-dire BPU2, BPU1, et BPU0 (ce qui signifie "Plans de Bits Utilisés").
Le Tableau 3-5 indique les valeurs à donner à ces bits et la manière dont le système assigne un numéro aux plans de bits.

Tableau 3-5 : Réglage du Nombre de Plans de Bits

Valeur	Nombre de Plans de Bits	Nom(s) des Plans de Bits
000	Aucun *	
001	1	PLAN 1
010	2	PLANS 1 et 2
011	3	PLANS 1 - 3
100	4	PLANS 1 - 4
101	5	PLANS 1 - 5
110	6	PLANS 1 - 6 **
111	7	Valeur inutilisée.

* Indique seulement la couleur d'arrière-plan; on ne verra aucun plan de bits.

** On utilise six plans de bits seulement en mode double aire de jeu ['dual-playfield'] et en mode 'hold-and-modify' (décrit dans la section "Sujets Avancés").

NOTE

On ne peut valider les bits du registre BPLCON0 individuellement. Pour activer un bit quelconque, on doit les recharger tous.

L'exemple qui suit montre comment informer le système pour qu'il utilise deux plans de bits en basse résolution.

```
MOVE.W #$2200,BPLCON0+CUSTOM ; Ecrire dans ce registre
```

Comme le registre BPLCON0 est utilisé pour régler d'autres aspects de l'affichage et que les bits ne peuvent être validés individuellement, l'exemple ci-dessus met aussi en place d'autres paramètres (nous donnerons une description de tous ces paramètres plus loin dans ce même chapitre).

- o Le mode 'hold-and-modify' est invalidé.
- o Le mode "aire de jeu simple" ['single-playfield'] est invalidé.
- o La couleur composite pour la vidéo est validée. (Ceci n'est pas possible dans tous les modes.)
- o L'audio pour le 'genlock' est invalidée.
- o Le crayon optique est invalidé.
- o Le mode entrelacé est invalidé.
- o Le re-synchronisation externe est invalidée (concerne le 'genlock').

SELECTION DE LA RESOLUTION HORIZONTALE ET VERTICALE

Les écrans standard des téléviseurs domestiques sont mieux adaptés à un affichage en basse résolution. Ce mode permet d'avoir 320 pixels pour chaque ligne horizontale. Les moniteurs en noir et blanc haute définition et les moniteurs RVB peuvent produire des affichages en mode haute définition, mode qui permet d'avoir 640 pixels pour chaque ligne horizontale. Si l'on définit un objet en basse résolution et on l'affiche ensuite en haute résolution, celui-ci sera moins large de la moitié.

Pour mettre en place le mode de résolution horizontale, on écrit dans le bit 15, HIRES, dans le registre BPLCON0 :

Mode haute résolution ? écrire 1 dans le bit 15.
Mode basse résolution ? écrire 0 dans le bit 15.

Remarquez que, en mode haute résolution, on peut avoir jusqu'à quatre plans de bits par aire de jeu et, par conséquent, jusqu'à 16 couleurs.

Par rapport au mode non entrelacé, le mode entrelacé permet d'afficher une double quantité de données dans une même surface verticale. Ceci est rendu possible en doublant le nombre de lignes s'affichant à l'écran. Le Tableau qui suit indique le nombre de lignes nécessaires pour remplir un écran normal qui ne soit pas en 'overscan'.

Tableau 3-6 : Lignes d'une Aire de Jeu Normale.

	NTSC	PAL
Non entrelacées	200	256
Entrelacées	400	512

En mode entrelacé, les circuits de balayage vertical décalent le début de tous les autres champs au milieu d'une ligne de balayage.

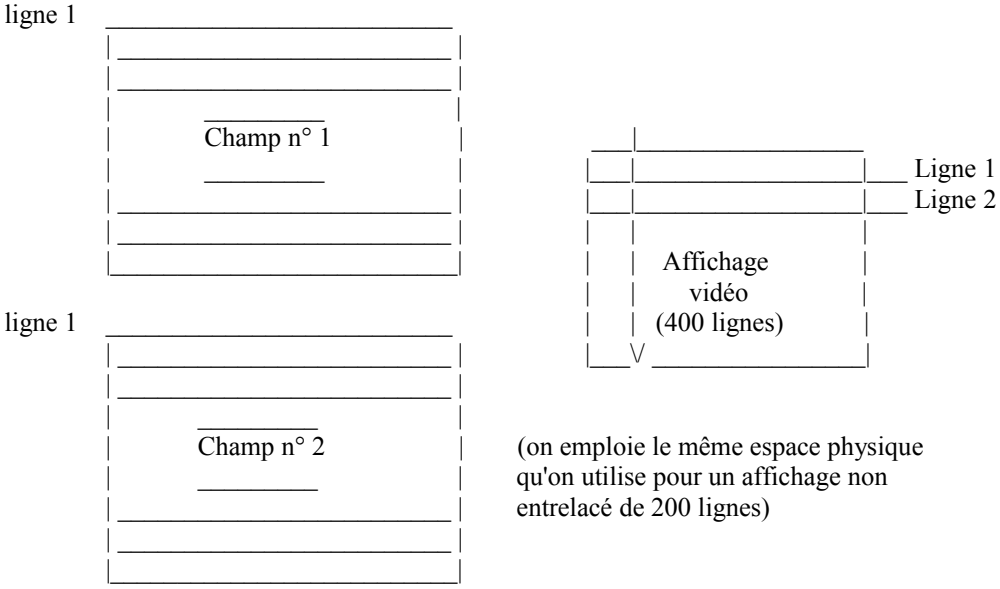


PLANCHE 3-5 : Entrelacement

Bien que le mode entrelacé nécessite une modeste quantité de travail supplémentaire (comme nous le verrons plus loin dans cette section), il permet un réglage fin qui est indispensable pour certains effets graphiques. Regardez la diagonale de la Planche 3-6 telle qu'elle s'affiche en mode non entrelacé et en mode entrelacé. L'entrelacement élimine les irrégularités ou "effet d'escalier" des arêtes de la ligne elle-même.

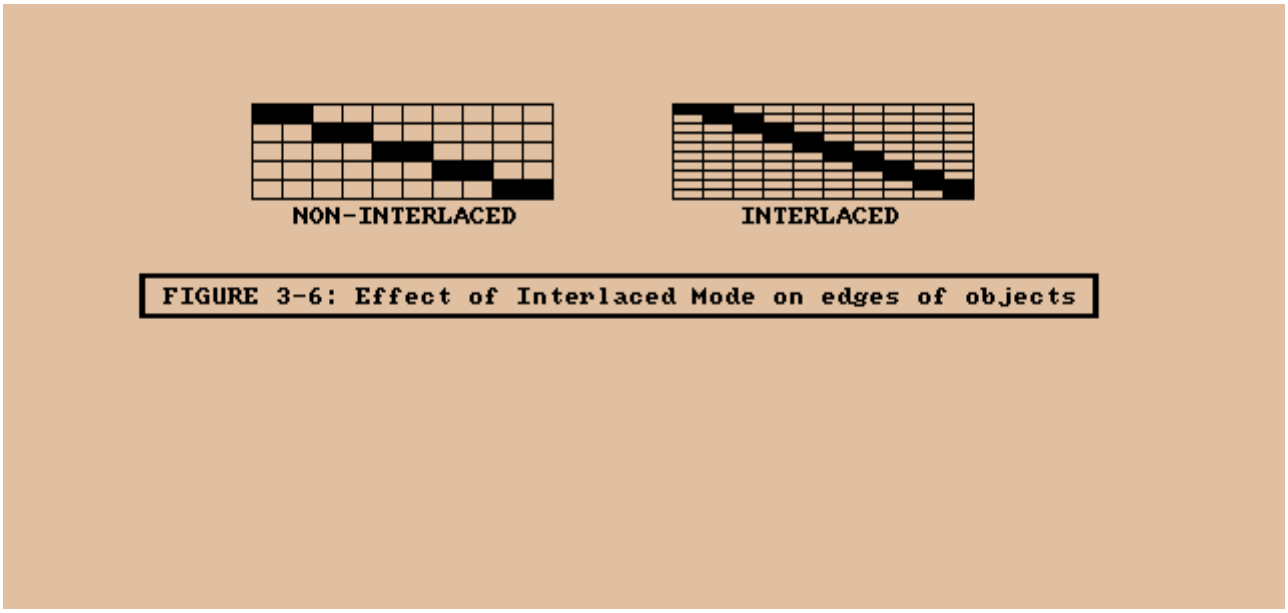


PLANCHE 3-6 : Effet produit par le Mode Entrelacé sur les arêtes des Objets.

Quand on utilise le canal DMA spécial du 'blitter' pour tracer des lignes ou des polygones dans une aire de jeu entrelacée, celle-ci est traitée bien plus comme un affichage unique que comme une suite de champs pairs et impairs. Cependant, grâce à l'entrelacement, vous aurez des arêtes plus douces.

Pour mettre en fonction le mode entrelacé ou le mode non entrelacé, on écrit dans le bit 2, LACE, du registre BPLCON0 :

Pour le mode entrelacé --> écrire 1 dans le bit 2.
Pour le mode non entrelacé --> écrire 0 dans le bit 2.

Comme nous l'avons expliqué ci-dessus dans "Sélection du Nombre de Plans de Bits", il n'est pas possible de valider individuellement les bits dans BPLCON0.

L'exemple suivant montre comment spécifier les modes haute résolution et entrelacé.

```
MOVE.W #$A204,BPLCON0+CUSTOM            ; Ecrire dans ce registre.
```

L'exemple ci-dessus valide aussi d'autres paramètres qu'on contrôle par l'intermédiaire du registre BPLCON0. Voici ces paramètres :

- Le mode haute résolution est validé.
- Utilisation de deux plans de bits.
- Le mode 'hold-and-modify' est invalidé.
- Le mode "aire de jeu simple" ['single-playfield'] est invalidé.
- La couleur composite pour la vidéo est validée.
- L'audio pour le 'genlock' est invalidée.
- Le crayon optique est invalidé.
- Le mode entrelacé est validé.
- Le re-synchronisation externe est invalidée.

La quantité de mémoire à réserver, nécessaire à chaque plan de bits, dépend des modes de résolution qu'on aura sélectionnés car les aires de jeu entrelacées et en haute résolution possèdent plus de données et nécessitent des plans de bits plus importants.

RESERVATION DE MEMOIRE POUR DES PLANS DE BITS

Après avoir fixé le nombre de plans de bits et indiqué les modes de résolution, nous sommes prêts pour réserver la mémoire. Un plan de bits est constitué d'une séquence de mots mis bout à bout dans des emplacements mémoire consécutifs. Lorsqu'on travaille sous le système d'exploitation de l'Amiga, on utilise un appel au système comme AllocMem() pour déplacer un bloc de mémoire de la liste libre et le mettre à la disposition d'un programme. Du moment que la machine a pris la relève du SE, réservons simplement une zone de mémoire pour les plans de bits. Ensuite, mettons en place les registres qui pointent sur les plans de bits (BPLxPTH/BPLxPTL) pour qu'ils pointent sur l'adresse où commence chacun des plans de bits utilisé. L'adresse initiale est celle du mot mémoire qui contient les bits du coin supérieur gauche du plan de bits.

Le Tableau 3-7 indique la quantité de mémoire nécessaire pour des aires de jeu simples. Il vous faudra équilibrer les besoins en résolution et en couleur avec la quantité de mémoire dont dispose le système.

Tableau 3-7 : Besoins en Mémoire de l'Aire de Jeu, en NTSC

Taille de l'Image	Modes	Nombre d'Octets par Plan de Bits
320 X 200	Basse résolution, non entrelacée	8000
320 X 400	Basse résolution, entrelacée	16000
640 X 200	Haute résolution, non entrelacée	16000
640 X 400	Haute résolution, entrelacée	32000

Tableau 3-7 : Besoins en Mémoire de l'Aire de Jeu, en PAL

Taille de l'Image	Modes	Nombre d'Octets par Plan de Bits
320 X 256	Basse résolution, non entrelacée	8192
320 X 512	Basse résolution, entrelacée	16384
640 X 256	Haute résolution, non entrelacée	16384
640 X5120	Haute résolution, entrelacée	32768

EXEMPLE DE TAILLE D'UN PLAN DE BITS EN NTSC

Par exemple, pour employer un plan de bits normal, en basse résolution, avec affichage non entrelacé, avec 320 pixels par ligne d'affichage et pour un total de 200 lignes, chaque ligne du plan aura besoin de 40 octets (320 bits divisé par 8 bits par octet = 40). Multiplions les 200 lignes par 40 octets par ligne pour obtenir 8000 octets par plan de bits, comme nous le disions ci-dessus.

Une aire de jeu en basse résolution, en mode non entrelacé, composée de deux plans de bits aura besoin d'une zone de mémoire de 16000 octets. Pour chaque plan de bits, la mémoire devra être continue, on aura donc besoin de deux blocs de mémoire disponibles de 8000 octets. La Planche 3-7 montre une zone de mémoire de 8000 octets organisée en 200 lignes de 40 octets chacune, fournissant un bit pour chaque position de pixel dans le plan d'affichage.

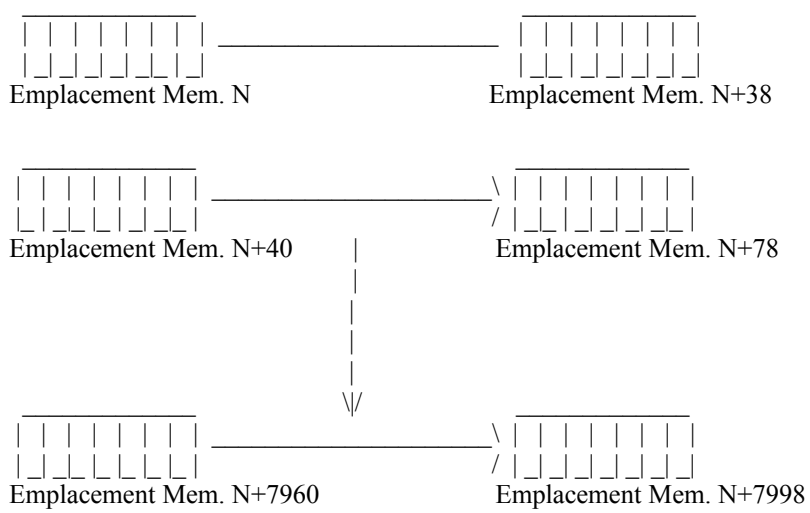


PLANCHE 3-7 : Organisation de la Mémoire pour un Plan de Bits Simple.

On accède aux plans de bits en mémoire par deux registres d'adresse BPLxPTH et BPLxPTL, un pour chaque plan (en tout, 12 registres). L'emplacement "x" du nom du registre contient le numéro du plan de bits; par exemple, BPL1PTH et BPL1PTL contiennent l'adresse où commence le PLAN n° 1. Les paires de registres ayant un nom se terminant par PTH et PTL contiennent des adresses de 19 bits. Les programmeurs du 68000 peuvent traiter ces adresses comme des adresses de 32 bits et y écrire un mot long. On écrit dans le mot prioritaire, c'est-à-dire dans le registre dont le nom se termine par "PTH".

L'exemple ci-dessous montre comment mettre en place les pointeurs de plans de bits. En supposant d'avoir deux plans, le premier à l'adresse $21000 et l'autre à l'adresse $25000, le processeur mettra BPL1PT à $21000 et BPL2PT à $25000. Remarquez qu'habituellement cette tâche est confiée au Copper.

```
;
; Etant donné que les registres pointeurs des plans de bits sont organisés
; exclusivement en données de mots longs, nous avons la possibilité de
; stocker les adresses par un 'move' sur 32 bits...
;

   LEA    CUSTOM,a0                ; Sauvegarder l'adresse de base des puces
                                   ; spécialisées...
   MOVE.L $21000,BPL1PTH(a0)       ; Ecrire le pointeur sur le plan n° 1
   MOVE.L $25000,BPL2PTH(a0)       ; Ecrire le pointeur sur le plan n° 2
```

Remarquez que la quantité de mémoire que nous donnons ici ne concerne que l'aire de jeu. Vous pourriez avoir besoin de réserver de la mémoire supplémentaire pour d'autres éléments de l'affichage, pour les sprites, l'audio, l'animation, ainsi que pour des programmes d'application. Nous parlerons de la réservation de mémoire pour d'autres éléments de l'affichage dans les chapitres qui traitent de ces sujets.

CODAGE DES PLANS DE BITS POUR OBTENIR UN BON COLORIAGE

Après avoir spécifié le nombre de plans de bits et avoir mis en place leurs pointeurs, on pourra réellement écrire les codes des registres de couleur dans les plans concernés.

UNE AIRE DE JEU AVEC UNE OU DEUX COULEURS

Pour une aire avec une seule couleur, il suffira d'écrire des "0" dans tous les bits du plan comme nous le montrons dans l'exemple ci-dessous. Cette partie de programme remplit un plan de bits en basse résolution avec la couleur d'arrière-plan (COLOR00) en écrivant des "0" dans la zone mémoire concernée. Ce plan de bits commence à l'adresse $21000 et a une taille de 8000 octets.

```
        LEA       $21000,a0    ; Pointer sur le plan de bits
        MOVE.W    #2000,d0     ; Ecrire 2000 mots longs = 8000 octets
LOOP:   MOVE.L    #0,(a0)+     ; Recopier un zéro
        DBRA      d0,LOOP      ; Décrementer le compteur d'une unité et
                               ; boucler jusqu'à ce que ce soit fini
```

Pour une aire de jeu de deux couleurs, on définit un plan de bits qui aura des "0" là où on souhaitera avoir la couleur d'arrière-plan et des "1" là où on souhaitera la couleur du registre n° 1. La partie de programme d'exemple qui suit est identique à celle de l'exemple précédent, si l'on excepte que le plan de bits sera rempli avec $FF00FF00 au lieu d'être totalement rempli avec des 0. Ce choix produira deux couleurs.

```
        LEA       $21000,a0          ; Pointer sur le plan de bits
        MOVE.W    #2000,d0           ; Ecrire 2000 mots longs = 8000 octets
LOOP:   MOVE.L    #$FF00FF00,(a0)+   ; Recopier $FF00FF00
        DBRA      d0,LOOP            ; Décrementer le compteur d'une unité et
                                     ; boucler jusqu'à ce que ce soit fini
```

UNE AIRE DE JEU AVEC TROIS COULEURS OU PLUS

Pour trois couleurs ou plus, on aura besoin de plus d'un plan de bits. Le travail, ici, est de définir chaque plan de bits de telle façon que -quand il sera combiné avec d'autres plans pour l'affichage- chaque pixel contienne la bonne combinaison de bits. C'est un peu plus compliqué qu'avec une aire de jeu avec un seul plan de bits. Les exemples suivants montreront une aire de jeu avec quatre couleurs, mais l'idée de départ et la procédure à suivre sont identiques à celles employées pour des aires de jeu pouvant avoir jusqu'à 32 couleurs.

La Planche 3-8 montre deux plans de bits composant une aire de jeu avec quatre couleurs :

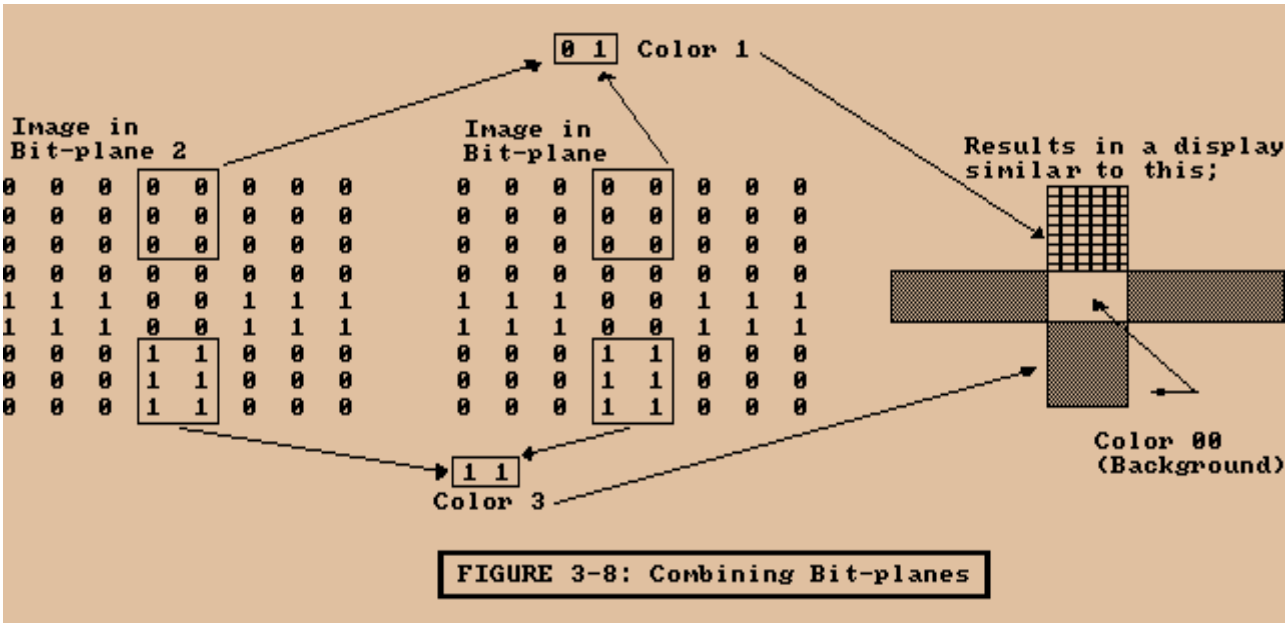


FIGURE 3-8: Combining Bit-planes

PLANCHE 3-8 : Combinaison de Plans de Bits

Pour attribuer la bonne couleur à chaque pixel de l'image ci-dessus, on mettra des "1" et des "0" dans les deux plans de bits de façon correcte.

De cette façon, dans une aire de jeu simple, on pourra combiner jusqu'à cinq plans de bits. L'utilisation de cinq plans permettra d'avoir un choix de 32 couleurs différentes pour chaque pixel. Les tableaux de sélection de couleur pour les aires de jeu à la fin de ce chapitre récapitulent les combinaisons de bits pour les aires de jeu constituées de quatre ou cinq plans de bits.

DEFINITION DES DIMENSIONS DE LA FENETRE D'AFFICHAGE

Après avoir défini l'aire de jeu intégralement, il nous faudra définir les dimensions de la fenêtre d'affichage qui seront les dimensions véritables de l'affichage à l'écran. Le réglage des dimensions de la fenêtre d'affichage affecte la zone d'affichage dans sa totalité, y compris le cadre et les sprites, et non uniquement l'aire d'affichage. Il ne sera pas possible d'afficher des objets en dehors de la fenêtre d'affichage telle qu'on l'aura définie. De plus, l'épaisseur du cadre autour de l'aire d'affichage dépendra des dimensions de la fenêtre elle-même.

L'aire de jeu simple décrite dans cette section a les mêmes dimensions que la zone d'affichage de l'écran et aussi les mêmes dimensions que la fenêtre d'affichage. Ce n'est pas toujours la cas; souvent la fenêtre d'affichage est plus petite que la véritable "grande image" de l'aire de jeu telle qu'elle définie en mémoire (la trame). Une fenêtre d'affichage plus petite que l'aire de jeu permet d'afficher des segments d'une aire de jeu plus grande ou de faire défiler l'aire de jeu dans la fenêtre.
On peut aussi définir des fenêtres d'affichage plus grandes que l'aire de jeu simple. Nous décrirons ce type d'aires de jeu plus grandes, ainsi que les fenêtres d'affichage de formats différents dans la section "Plans de Bits et Fenêtres d'Affichage de tous les Formats" ci-dessous.

On détermine les dimensions d'une fenêtre d'affichage en indiquant les positions verticale et horizontale de son point d'origine et de son point d'arrivée et en écrivant ces positions dans les registres de la fenêtre d'affichage. La résolution du point vertical d'origine et celle du point d'arrivée est égale à celle d'une ligne de balayage. La résolution du point horizontal d'origine et celle du point d'arrivée est égale à celle d'un pixel en basse résolution. Chaque position à l'écran définit la position horizontale et verticale d'un pixel, et cette position est déterminée par les coordonnés x et y du pixel. Le document qui suit montre les coordonnées x et y sous la forme (x,y). Bien que ces coordonnés commencent à (0,0), dans le coin supérieur gauche de l'écran, normalement la première position horizontale utilisée sera $81 et la première position verticale sera $2C. Ces positions d'origine sont les mêmes en NTSC et en PAL.

Le 'hardware' permettrait d'indiquer la position du point d'origine avant ($81,$2C), mais alors une partie de l'affichage ne serait pas visible. La différence entre la position du point d'origine absolue (0,0) et la position normale ($81,$2C) est une conséquence de la façon selon laquelle sont conçus plusieurs moniteurs vidéo pour l'affichage. Pour éviter la distorsion qui se manifesterait sur les bords extrêmes de l'écran, le faisceau de balayage balaie dans une zone plus grande que la partie antérieure pouvant être affichée par l'écran. Une position d'origine à ($81,$2C) permet de centrer un affichage de dimensions normales, en laissant un filet de huit pixels en basse résolution autour de la fenêtre d'affichage. La Planche 3-9 montre la relation existant entre une fenêtre d'affichage normale, la zone visible de l'écran et la zone réellement parcourue par le faisceau de balayage.

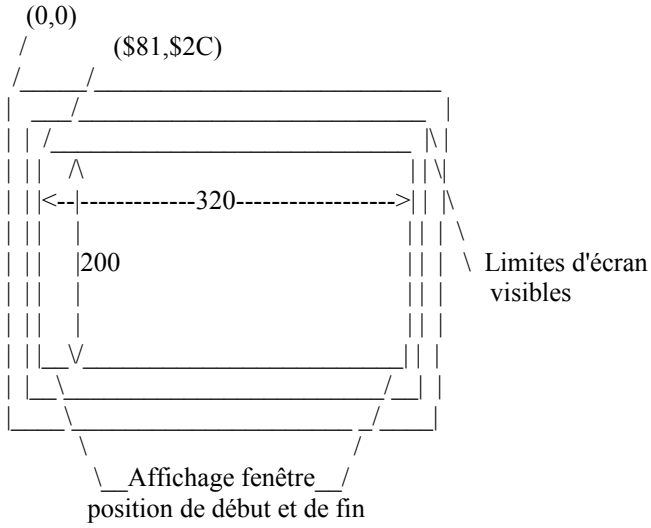


PLANCHE 3-9 : Placement de l'Affichage à l'Ecran

REGLAGE DE LA POSITION DU POINT D'ORIGINE DE LA FENETRE D'AFFICHAGE

Une position du point d'origine horizontale d'environ $81 et une position du point d'arrivée verticale d'environ $2C permettent de centrer l'affichage sur la plupart des écrans de télévision standard. Si on sélectionne le mode haute résolution (640 pixels horizontalement) ou le mode entrelacé (400 lignes en NTSC, 512 en PAL), la position du point d'origine n'est pas modifiée. Cette position est toujours interprétée en basse résolution, en mode non entrelacé. En d'autres termes, on sélectionne une position d'origine qui représente les bonnes coordonnés en basse résolution, en mode non entrelacé.

Le registre DIWSTRT (c'est-à-dire "Origine de la Fenêtre d'Affichage") est celui qui contrôle la position du point d'origine de la fenêtre d'affichage. Ce registre contient les deux éléments, l'horizontal et le vertical, des positions d'origine de la fenêtre d'affichage, connus respectivement comme HSTART et VSTART. L'exemple suivant initialise DIWSTRT pour une aire de jeu simple. Vous écrirez $2C pour VSTART et $81 pour HSTART.

```
LEA    CUSTOM,a0                ; Sauvegarde l'adresse de base des puces
                                ; spécialisées...
MOVE.W #$2C81,DIWSTRT(a0)       ; Registre du point d'origine de la
                                ; fenêtre d'affichage...
```

REGLAGE DE LA POSITION DU POINT D'ARRIVEE DE LA FENETRE D'AFFICHAGE

En outre, on devra mettre en place la position du point d'arrivée de la fenêtre d'affichage, c'est-à-dire le coin inférieur droit de cette même fenêtre. Si vous sélectionnez la haute résolution ou le mode entrelacé, la position du point d'arrivée ne sera pas modifiée. Celle-ci, comme la position d'origine est interprétée en basse résolution, en mode non entrelacé.

Le registre DIWSTOP (c'est-à-dire "Arrivée de la Fenêtre d'Affichage") contrôle la position d'arrivée de la fenêtre d'affichage. Ce registre contient les deux éléments, l'horizontal et le vertical, des positions d'arrivée de la fenêtre d'affichage, connus respectivement comme HSTOP et VSTOP. La suite d'instructions ci-dessous vous montre comment initialiser HSTOP et VSTOP pour une aire de jeu simple, en considérant que la position d'origine se trouve à ($81,$2C). Remarquez que la valeur HSTOP écrite est la valeur réelle négative 256 ($100). La valeur HSTOP normale est ($1C1) mais on l'écrit sous la forme ($C1). La valeur de HSTOP est la même en NTSC et en PAL.

La position VSTOP se limite à la moitié inférieure de l'écran. Ceci est rendu possible par le 'hardware' qui force le MSB [le bit de poids fort] de la position d'arrivée à complémenter le prochain MSB. Ceci permet -pour une position VSTOP supérieure à 256 ($100)- d'utiliser seulement 8 bits. Habituellement, le VSTOP est initialisé à ($f4) en NTSC et à ($2C) en PAL.

En NTSC, la valeur normale de DIWSTRT est ($2C81).
En NTSC, la valeur normale de DIWSTOP est ($F4C1).

En PAL, la valeur normale de DIWSTRT est ($2C81).
En PAL, la valeur normale de DIWSTOP est ($2CC1).

L'exemple suivant initialise DIWSTOP pour une aire de jeu simple à $F4 pour la position verticale et à $C1 pour celle horizontale.

```
LEA    CUSTOM,a0                ; Sauvegarde l'adresse de base des puces
                                ; spécialisées...
MOVE.W #$F4C1,DIWSTOP(a0)       ; Registre du point d'arrivée de la
                                ; fenêtre d'affichage
```

Tableau 3-9 : Récapitulation sur DIWSTRT et DIWSTOP.

	-Valeurs Nominales-		-Valeurs Possibles-	
	NTSC	PAL	MIN	MAX
DIWSTRT :				
VSTART	$2C	$2C	$00	$FF
HSTART	$81	$81	$00	$FF
DIWSTOP :				
VSTOP	$F4	$2C (=$12C)	$80	$7F (=$17F)
HSTOP	$C1	$C1	$00 (=$100)	$FF (=$1FF)

INFORMER LE SYSTEME SUR LA MANIERE DE PRELEVER ET D'AFFICHER LES DONNEES

Une fois définies les dimensions et l'emplacement de la fenêtre d'affichage, il faudra fournir au système l'emplacement à l'écran des données prélevées en mémoire. Pour ce faire, on décrit la position horizontale et verticale de de début et de fin de chaque ligne et l'on écrit la valeur de ces positions dans les registres d' "acquisition de données". Ces registres ont une résolution de quatre pixels (contrairement aux registres de la fenêtre d'affichage qui ont une résolution d'un pixel). Chaque position indiquée sera éloignée de quatre pixels par rapport à la précédente. Le pixel numéro 0 sera à la position 0; le pixel numéro 4 à la position 1 et ainsi de suite.

Le début d'acquisition des données et les positions du point d'origine de la fenêtre d'affichage interagissent l'un avec l'autre. Il est recommandé de limiter les valeurs de début d'acquisition de données à une résolution de programmation de 16 pixels (8 périodes d'horloge, en mode basse résolution, 4 périodes en mode haute résolution). Après la première acquisition de données, le 'hardware' a parfois besoin d'un temps d'arrêt pour pouvoir les afficher de manière concrète. La conséquence de ce retard sera une différence de 4,5 périodes d'horloge, concernant la couleur, entre la valeur du point d'origine de la fenêtre et le début d'acquisition des données.

La valeur normale de DDFSTRT en basse résolution est égale à ($0038).
La valeur normale de DDFSTRT en haute résolution est égale à ($003C).

Rappelez-vous que la résolution 'hardware' du point d'origine et celle du point d'aboutissement de la fenêtre d'affichage est égale au double de la résolution 'hardware' d'acquisition des données :

$$\frac{\$81}{2} - 8.5 = \$38$$

$$\frac{\$81}{2} - 4.5 = \$3c$$

La relation existant entre le début et la fin d'acquisition des données est la suivante

DDFSTRT = DDFSTOP - (8*(nombre de mots - 1)) en basse résolution

DDFSTRT = DDFSTOP - (4*(nombre de mots - 2)) en haute résolution

La valeur normale de DDFSTOP en basse résolution est égale à ($00D0). La valeur normale de DDFSTOP en haute résolution est égale à ($00D4).

L'exemple suivant initialise le début d'acquisition de données à l'adresse $0038 et la fin à $00D0, pour une aire de jeu simple.

```
LEA    CUSTOM,a0                ; Pointer sur l'adresse de base du 'hardware'
MOVE.W #$0038,DDFSTRT(a0)       ; Ecrire dans DDFSTRT
MOVE.W #$00D0,DDFSTOP(a0)       ; Ecrire dans DDFSTOP
```

Il faudra aussi informer le système sur les octets qui sont en mémoire et qui appartiennent à chaque ligne d'affichage horizontale. Pour ce faire, on indique la valeur modulo. Cette valeur se réfère au nombre d'octets en mémoire entre le dernier mot d'une ligne horizontale et le commencement du premier mot de la ligne suivante. Ainsi, la valeur modulo donne au système le moyen de convertir les données d'un plan de bits, stockées sous forme linéaire (chaque octet de données se trouvant à une adresse mémoire croissant de façon séquentielle), en une forme rectangulaire (une "ligne" de données disposées séquentiellement, suivie d'une autre ligne). Pour une aire de jeu simple, c'est-à-dire une aire de jeu en mémoire ayant les mêmes dimensions que la fenêtre d'affichage, la valeur modulo sera égale à 0, étant donné que la zone mémoire contient exactement le même nombre d'octets qu'on souhaite afficher à l'écran. Les Planches 3-10 et 3-11 montrent le dessin simple d'un plan de bits en mémoire et la manière de s'assurer qu'on prélève les bonnes données.

Les pointeurs d'adresse du plan de bits (BPLxPTH et BPLxPTL) sont utilisés par le système pour prélever les données à envoyer à l'écran. Ces pointeurs sont dynamiques; une fois qu'a commencé le prélèvement des données, ils sont sans arrêt augmentés d'une unité pour pouvoir pointer sur le mot suivant qu'il faudra prélever (les données sont prélevées par paquets de deux octets à la fois). Lorsque la condition de fin de ligne est atteinte (définie par le registre d'acquisition DDFSTOP), cette valeur modulo est additionnée au pointeur du premier mot qui devra être prélevé dans la prochaine ligne horizontale.

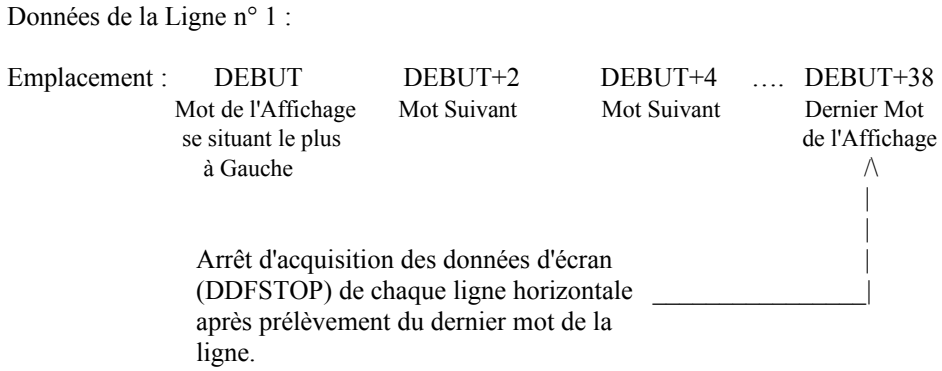


Planche 3-10 : Acquisition des Données de la Première Ligne pour Modulo = 0

Après prélèvement de la première ligne, les pointeurs BPLxPTH et BPLxPTL sur les plans de bits contiendront une valeur DEBUT+40. La valeur modulo (0 dans ce cas) sera additionnée à la valeur du pointeur courante, ainsi, quand le pointeur commencera l'acquisition de données de la ligne suivante, il prélèvera les données voulues sur cette nouvelle ligne. Les données de la ligne suivante commenceront donc à l'emplacement mémoire DEBUT+40.

Données de la Ligne n° 2 :

Emplacement :	DEBUT+40	DEBUT+42	DEBUT+44	….	DEBUT+78
	Mot de l'Affichage se situant le plus à Gauche	Mot Suivant	Mot Suivant		Dernier Mot de l'Affichage

Planche 3-11 : Acquisition des Données de la Deuxième Ligne pour Modulo = 0

Remarquez que les pointeurs contiennent toujours un nombre pair car les données sont prélevées de l'affichage à raison d'un seul mot à la fois.

Il y a deux registres modulo, BPL1MOD pour les plans de bits de numéro impair et BPL2MOD pour ceux de numéro pair. Ceci permet de distinguer les valeurs modulo de chaque aire de jeu en mode "double aire de jeu". Dans des applications normales, BPL1MOD et BPL2MOD sont identiques.

L'exemple suivant initialise le modulo à 0, pour une aire de jeu en basse résolution, avec un seul plan de bits. Celui-ci sera de numéro impair.

```
MOVE.W #0,BPL1MOD+CUSTOM     ; Initialiser modulo à 0
```

ACQUISITION DE DONNEES EN HAUTE RESOLUTION

Quand on utilise le mode haute résolution pour afficher une aire de jeu simple, il faut prélever 80 octets par ligne, au lieu de 40.

VALEUR MODULO EN MODE ENTRELACE

En mode entrelacé, il faudra définir à nouveau la valeur modulo car ce mode utilise deux balayages séparés de l'écran vidéo pour un seul affichage de l'aire de jeu. Lors du premier balayage, on prélève les lignes de numéro impair pour les envoyer à l'écran; et lors du deuxième balayage, on prélève les ligne de numéro pair.

Les plans de bits d'un écran de dimensions maximales, avec affichage en mode entrelacé, auront 400 lignes en NTSC (512 en PAL), au lieu de 200 lignes en NTSC (256 en PAL). En présumant que l'aire de jeu en mémoire ait une largeur normale de 320 pixels, les données de l'image entrelacée commenceront aux emplacements suivants (ce sont toutes des adresses d'octets) :

Ligne 1 DEBUT
Ligne 2 DEBUT+40
Ligne 3 DEBUT+80
Ligne 4 DEBUT+120

et ainsi de suite. Par conséquent, on emploiera un modulo de 40 pour sauter les lignes de l'autre champ. Pour les champs pairs, les pointeurs de plans de bits commencent à DEBUT. Pour les champs impairs, les pointeurs de plans commencent à DEBUT+40.

Dans les affichages en mode entrelacé, on peut employer le Copper pour gérer la réinitialisation des pointeurs de plans de bits.

AFFICHAGE ET RE-AFFICHAGE D'UNE AIRE DE JEU

On fait démarrer l'affichage d'une aire de jeu en s'assurant que les pointeurs de plans de bits ont été initialisés et que le DMA est activé. Vous validez le DMA d'un plan de bits en écrivant 1 dans le bit BPLEN du registre DMACON (c'est-à-dire, "Contrôle du DMA"). Pour des renseignements sur l'initialisation de ce registre, voyez le Chapitre 7, "'Hardware' de Contrôle du Système".

A chaque fois que l'aire de jeu sera redessinée, vous devrez réinitialiser les pointeurs de plans de bits. Leur réinitialisation est nécessaire car on les a augmentés d'une unité pour pouvoir pointer sur les mots suivants en mémoire et on doit les faire pointer à nouveau sur le prochain mot pour l'affichage suivant. On écrit des instructions Copper pour gérer le nouveau dessin ou pour faire exécuter cette opération comme une partie de la tâche de suppression verticale.

VALIDATION DE L'AFFICHAGE DES COULEURS

Le modèle A1000 de série a une sortie vidéo composite. Dans ce modèle, pour générer un signal d'affichage couleur composite, il faut mettre à 1 le bit 9 de BPLCON0. Les A500 et les A2000 n'ont pas la possibilité de générer un signal de sortie couleur composite sans l'ajout d'un 'hardware' adapté.

NOTE

La validation du transfert de couleur n'affecte pas le signal vidéo RVB. Celui-ci est généré correctement sans avoir à s'occuper de la sortie du signal vidéo composite.

RECAPITULATION SUR L'AIRE DE JEU SIMPLE

Nous récapitulons ci-dessous les étapes à suivre pour définir une aire de jeu simple :

1. Définir les Caractéristiques de l'Aire de Jeu

a. Spécifier la hauteur en lignes :

- En NTSC :
  - * 200 en mode non entrelacé.
  - * 400 en mode entrelacé.
- En PAL :
  - * 256 en mode non entrelacé.
  - * 512 en mode entrelacé.

b. Spécifier la largeur en pixels :

- 320 en mode basse résolution.
- 640 en mode haute résolution.

c. Spécifier la couleur de chaque pixel :

- Charger les couleurs voulues dans le registre de la table de couleurs.
- Définir la couleur de chaque pixel sous forme de valeur binaire qui pointe sur le registre de couleur qu'on souhaite.
- Construire les plans de bits.
- Initialiser les registres de plans de bits.
  - * Bits 12-14 de BPLCON0 - nombre de plans de bits (BPU2 - BPU0).
  - * BPLxPTH - pointeur sur la position de début en mémoire du plan de bits (écrit sous forme de mot long).

d. Spécifier la résolution :

o Basse résolution :

* 320 pixels pour chaque ligne horizontale.

* Mettre à zéro le bit 15 du registre BPLCON0 (HIRES).

o Haute résolution :

* 640 pixels pour chaque ligne horizontale.

* Mettre à 1 le bit n° 15 du registre BPLCON0 (HIRES).

e. Spécifier le mode, entrelacé ou non entrelacé :

o Mode entrelacé :

* 400 lignes verticales en NTSC, 512 en PAL.

* Mettre à 1 le bit n° 2 du registre BPLCON0 (LACE).

o Mode non entrelacé :

* 200 lignes verticales en NTSC, 256 en PAL.

* Mettre à zéro le bit n° 2 de BPLCON0 (LACE).

2. Réserver la Mémoire. Pour calculer les octets de données nécessaires pour un plan de bits, servez-vous de la formule suivante :

Octets par ligne * lignes de l'aire de jeu * nombre de plans de bits

3. Définir les dimensions de la Fenêtre d'Affichage.

o Ecrire la position du point d'origine de la fenêtre d'affichage dans DIWSTRT :

* Position horizontale dans les bits de 0 à 7 (bits de poids faible).

* Position verticale dans les bits de 8 à 15 (bits de poids fort).

o Ecrire la position d'arrivée de la fenêtre d'affichage dans DIWSTOP :

* Position horizontale dans les bits de 0 à 7.

* Position verticale dans les bits de 8 à 15.

4. Définir l'Acquisition des Données. Initialiser les registres DDFSTRT et DDFSTOP:

   o Pour DDFSTRT, utiliser la position horizontale, comme nous l'avons montré dans "Réglage de la Position du Point d'Arrivée de la Fenêtre d'Affichage".

   o Pour DDFSTOP, utiliser la position horizontale, comme nous l'avons montré dans "Réglage de la Position du Point d'Arrivée de la Fenêtre d'Affichage".

5. Définir la valeur Modulo. Initialiser les registres BPL1MOD et BPL2MOD. Mettre modulo à 0 pour le mode non entrelacé, à 40 pour l'entrelacé.

6. Ecrire les instructions Copper pour gérer le nouvel affichage.

7. Valider l'Affichage de la Couleur. Pour l'A1000, mettre à 1 le bit 9 de BPLCON0 pour valider l'affichage de la couleur sur un moniteur vidéo RVB. Le signal vidéo RVB n'est pas affecté. Seul l'A1000 possède une sortie vidéo composite, les autres modèles Amiga ne peuvent se servir de cette caractéristique, en utilisant le 'hardware' standard.

EXEMPLES DE CONSTRUCTION D'AIRES DE JEU SIMPLES

Les exemples suivants montrent comment initialiser les registres et comment écrire des listes pour le co-processeur pour construire deux aires de jeu différentes.

Le premier exemple met en place une aire de jeu de 320 x 200 avec un seul plan de bits à l'adresse $21000. En outre, nous mettons en place une liste Copper à l'adresse $20000.

Cet exemple repose sur le fichier 'include' "hw_examples.i", qu'on trouve à l'Appendice J.

```
   LEA    CUSTOM,a0                ; a0 pointe sur les puces spécialisées
   MOVE.W #$1200,BPLCON0(a0)       ; Un plan de bits, valider la couleur composite
   MOVE.W #0,BPLCON1(a0)           ; Mettre à 0 la valeur de défilement horizontal
   MOVE.W #0,BPL1MOD(a0)           ; Mettre modulo à 0 pour tous les plans de bits impair
   MOVE.W #$0038,DDFSTRT(a0)       ; Mettre le début d'acquisition des données à $38
   MOVE.W #$00D0,DDFSTOP(a0)       ; Mettre la fin d'acquisition des données à $D0
   MOVE.W #$2C81,DIWSTRT(a0)       ; Mettre DIWSTRT à $2C81
   MOVE.W #$F4C1,DIWSTOP(a0)       ; METTRE DIWSTOP à $F4Cl
   MOVE.W #$0F00,COLOR00(a0)       ; Mettre la couleur d'arrière-plan rouge
   MOVE.W #$0FF0,COLOR01(a0)       ; Mettre le registre de couleur n° 1 à jaune
;
; Remplir les plans de bits avec $FF00FF00 pour générer des bandes
;
   MOVE.L #$21000,a1           ; Pointer au début du plan de bits
   MOVE.L #$FF00FF00,d0        ; Nous écrirons $FF00FF00 en mots longs
   MOVE.W #2000,d1             ; 2000 mots longs = 8000 octets
;
LOOP:
   MOVE.L d0,(a1)+             ; Ecrire un mot long
   DBRA   d1,LOOP              ; Décrémenter le compteur et boucler jusqu'à la fin
;
; Mettre en place la liste Copper à l'adresse $20000
;
   MOVE.L #$20000,a1           ; Pointer sur la destination de la liste Copper
   LEA    COPPERL(pc).a2       ; Pointer a2 sur les données de la liste Copper
```

```
CLOOP:
   MOVE.L (a2),(a1)+          ; Transférer un mot
   CMPI.L #$FFFFFFFE,(a2)+    ; Tester le dernier mot long de la liste Copper
   BNE    CLOOP               ; Boucler jusqu'à la fin du transfert de toute la liste Copper
;
; Faire pointer le Copper sur la liste Copper
;
   MOVE.L #$20000,COP1LCH(a0)      ; Ecrire dans le registre d'adresse du Copper
   MOVE.W COPJMP1(a0),d0           ; Forcer le Copper à l'adresse $20000
;
; Démarrer le DMA
;
   MOVE.W #(DMAF_SETCLR!DMAF_COPPER!DMAF_RASTER!DMAF_MASTER),DMACON(a0)
                        ; Valide le plan de bits et le DMA du Copper

  BRA ....              ; Aller exécuter la prochaine tâche
;
; Ce sont les données de la liste Copper.
;
COPPERL:
   DC.W BPL1PTH,$0002          ; Transférer $0002 à l'adresse $0E0 (BPL1PTH)
   DC.W BPL1PTL,$1000          ; Transférer $1000 à l'adresse $0E2 (BPL1PTL)
   DC.W $FFFF,$FFFE            ; Fin de la liste Copper
;
```

Le deuxième exemple met en place un affichage en haute résolution, en mode entrelacé avec un plan de bits. Cet exemple repose sur le fichier 'include' "hw_examples.i", qu'on trouve à l'Appendice J.

```
   LEA    CUSTOM,a0                ; Adresse des puces spécialisées
   MOVE.W #$9204,BPLCON0(a0)       ; Haute résolution, un plan de bits, mode entrelacé
   MOVE.W #0,BPLCON1(a0)           ; Valeur 0 pour défilement horizontal
   MOVE.W #80,BPL1MOD(a0)          ; Modulo = 80 pour les plans de bits impairs
   MOVE.W #80,BPL2MOD(a0)          ; Idem pour les plans de bits pairs
   MOVE.W #$003C,DDFSTRT(a0)       ; Mettre le début d'acquisition des données en haute résolution
   MOVE.W #$00D4,DDFSTOP(a0)       ; Mettre la fin d'acquisition des données
   MOVE.W #$2C81,DIWSTRT(a0)       ; Mettre l'origine de la fenêtre d'affichage
   MOVE.W #$F4C1,DIWSTOP(a0)       ; Mettre l'arrivée de la fênetre d'affichage
;
; Mettre en place les registres de couleur
;
   MOVE.W #$000F,COLOR00(a0)       ; Couleur d'arrière-plan = bleu
   MOVE.W #$0FFF,COLOR01(a0)       ; Couleur de premier plan = blanc

; Mettre en place un plan de bits à l'adresse S20000

   LEA    $20000,a1                ; Faire pointer a1 sur le plan de bits
   LEA    CHARLIST(pc),a2          ; a2 pointe sur les données de caractère
   MOVE.W #400,d1                  ; Ecrire 400 lignes de données
   MOVE.W #20,d0                   ; Ecrire 20 mots longs par ligne
L1:
   MOVE.L (a2),(a1)+               ; Ecrire un mot long
   DBRA   d0,L1                    ; Décrémenter le compteur et boucler jusqu'à la fin

   MOVE.W #20,d0                   ; Réinitialiser le compteur de mots longs
   ADDQ.L #4,a2                    ; Pointer sur le prochain mot de la liste de caractères
   CMPI.L #$FFFFFFFF,(a2)          ; Fin de la liste de caractères ?
   BNE    L2
   LEA    CHARLIST(pc),a2          ; Oui, réinitialiser a2 au début de la liste
L2:
   DBRA   d1,L1                    ; Décrémenter le compteur de ligne et boucler jusqu'à la fin
;
; Démarrer le DMA
;
   MOVE.W #(DMAF_SETCLR!DMAF_RASTER!DMAF_MASTER),DMACON(a0)
                                   ; Ne valide que le DMA du plan de bits, ne valide pas celui du Copper
```

```
; Comme cet exemple n'a pas de liste Copper, il s'installe dans une
; boucle en attente de l'intervalle de suppression verticale. Quand
; celle-ci se produit, on teste LOF ("bit de trame paire") de VPOSR.
; Si LOF = 0, il s'agira d'une trame impaire et les pointeurs de plans
; de bits seront initialisés pour pointer sur $20050. Si LOF = 1, il
; s'agit alors d'une trame paire et les pointeurs seront initialisés
; pour pointer sur $20000. Ce procédé permet de garder les trames
; paires et les trames impaires dans une relation correcte, l'une
; par rapport à l'autre.

VLOOP:
   MOVE.W INTREQR(a0),d0          ; Lire les demandes d'interruption
   AND.W  #$0020,d0               ; Masquer tout, sauf la suppression verticale
   BEQ    VLOOP                   ; Boucler jusqu'à l'arrivée de la suppression verticale
   MOVE.W #$0020,INTREQ(a0)       ; Remettre en fonction l'interruption verticale
   MOVE.W VPOSR(a0),d0            ; Lire le bit LOF dans d0, bit 15
   BPL    VL1                     ; si LOF = 0, sauter
   MOVE.L #$20000,BPL1PTH(a0)     ; LOF = 1, pointer sur $20000
   BRA    VLOOP                   ; Et retour
VL1:
   MOVE.L #$20050,BPL1PTH(a0)     ; LOF = 0, pointer sur $20050
   BRA    VLOOP                   ; Et retour
;
; Liste des caractères
;
CHARLIST:
   DC.L   $18FC3DF0,$3C6666D8,$3C66C0CC,$667CC0CC
   DC.L   $7E66C0CC,$C36666D8,$C3FC3DF0,$00000000
   DC.L   $FFFFFFFF
```

CREATION D'AFFICHAGE D'UNE DOUBLE AIRE DE JEU

Pour plus de souplesse, lors de la conception d'un affichage d'arrière-plan, on peut spécifier deux aires de jeu à la place d'une seule. En mode "double aire de jeu", une aire s'affiche directement devant les autres. Par exemple, l'affichage d'un jeu sur ordinateur pourrait comprendre une action qui serait en rapport avec une aire de jeu d'arrière-plan, tandis que l'autre aire de jeu est en train de montrer un tableau de contrôle au premier plan. Dans ce cas, on pourra modifier aussi bien le premier plan que l'arrière-plan sans avoir à redessiner tout l'affichage. On pourra aussi déplacer les deux aires de jeu de manière indépendante.

L'affichage d'une double aire de jeu est similaire à celui de l'aire de jeu simple, il n'en diffère que sur les points suivants :

- Chaque aire de jeu d'une double aire de jeu est composée d'un, deux ou trois plans de bits.

- Les couleurs de chaque aire de jeu (sept au maximum, plus le transparent) sont prises dans des jeux de registres de couleur différents.

- On doit valider un bit pour activer le mode double aire de jeu.

La Planche 3-12 montre l'affichage d'une double aire de jeu.

Dans la Planche 3-12, une des couleurs de chacune des deux aires de jeu est "transparente" (la couleur 0 de l'aire de jeu n° 1 et la couleur 8 de l'aire de jeu n° 2). On peut utiliser la transparence pour pouvoir montrer des détails précis de l'aire de jeu d'arrière-plan.

En mode double aire de jeu, chaque aire peut avoir au maximum trois plans de bits. Les registres de couleur de 0 à 7 sont assignés à l'aire de jeu n° 1, en relation avec le nombre de plans de bits utilisés. Les registres de couleur de 8 à 15 sont assignés à l'aire de jeu n° 2.

ASSIGNATION D'UN PLAN DE BITS EN MODE DOUBLE AIRE DE JEU

Les trois plans de bits de numéro impair (1, 3 et 5) sont regroupés ensemble par le 'hardware' et peuvent être employés dans l'aire de jeu n° 1. De manière inverse, les trois plans de bits de numéro pair (2, 4 et 6) sont regroupés et peuvent être employés dans l'aire de jeu n° 2. Les plans de bits sont assignés alternativement, comme le montre la Planche 3-13.

NOTE

En mode haute résolution, on peut avoir jusqu'à deux plans de bits par aire de jeu, les plans 1 et 3 dans l'aire n° 1 et les plans 2 et 4 dans l'aire n° 2.

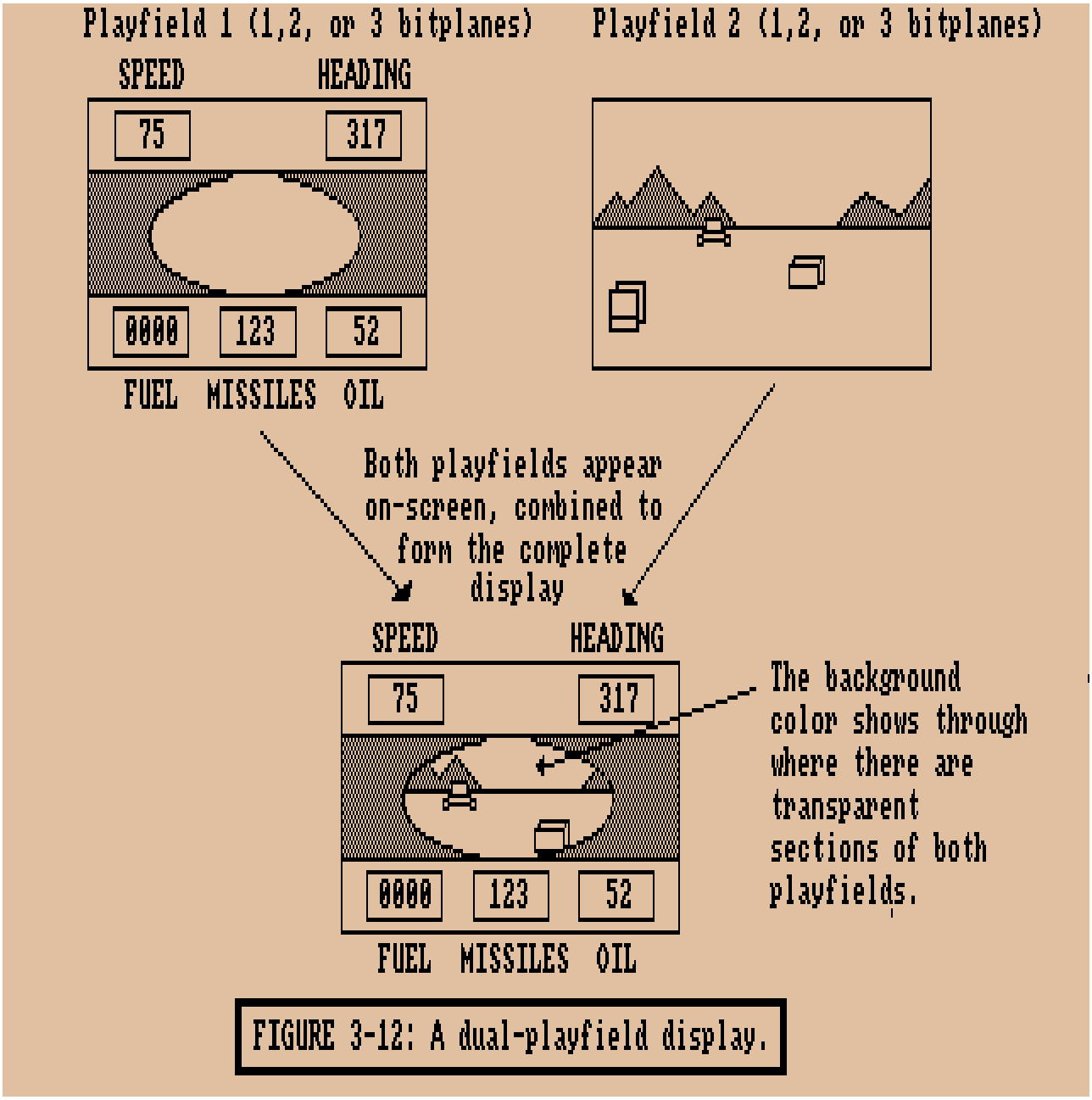
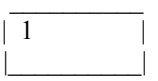
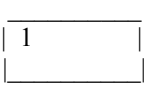
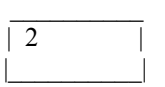
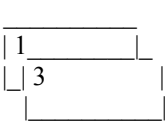
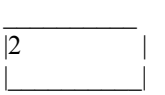
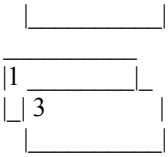
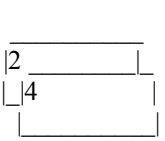
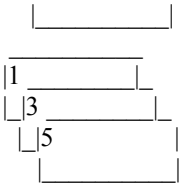
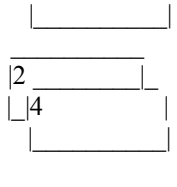
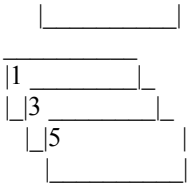
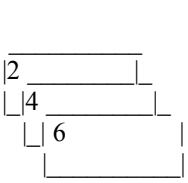


PLANCHE 3-12 : Affichage d'une Double Aire de Jeu.

Nombre de Plans de Bits "activés"	Aire de Jeu n° 1*	Aire de Jeu n° 2*
0	Aucun	Aucun
1		
2		
3		
4		
5		
6		

* NOTE : On peut placer une aire de jeu "devant" ou "derrière" les autres en utilisant le bit de "permutation"

PLANCHE 3-13: Nombre de Plan de Bits Assignés dans une Double Aire de Jeu

REGISTRES DE COULEUR EN MODE DOUBLE AIRE DE JEU

Quand vous vous servez d'une double aire de jeu, le 'hardware' interprète les numéros de couleur de l'aire de jeu n° 1 d'après les combinaisons de bits des aires n° 1, n° 3 et n° 5. Les bits du PLAN n° 5 sont les bits plus significatifs et constituent la partie plus significative du numéro du registre de couleur. Les bits du PLAN n° 0 sont les bits moins significatifs. Ces combinaisons de bits sélectionnent les premiers huit registres de couleur d'après la palette de couleur que nous montrons dans le Tableau 3-10.

Tableau 3-10 : Registres de Couleur de l'Aire de Jeu n° 1 en Mode Basse Résolution.

AIRE DE JEU N° 1

Combinaison de Bits	Couleur Sélectionnée
000	Mode Transparent
001	COLOR1
010	COLOR2
011	COLOR3
100	COLOR4
101	COLOR5
110	COLOR6
111	COLOR7

Le 'hardware' interprète les numéro de la couleur de l'aire de jeu n° 2 d'après les combinaisons de bits des aires de jeu n° 2, n° 4 et n° 6. Les bits du PLAN n° 6 sont les plus significatifs. Les bits du PLAN n° 2 sont les moins significatifs. Ces combinaisons de bits sélectionnent les registres de couleur d'après les deuxièmes huit couleurs de la table de couleurs comme nous le montrons dans le Tableau 3-11.

Tableau 3-11 : Registres de Couleur de l'Aire de Jeu n° 2 en Mode Basse Résolution.

AIRE DE JEU n° 2

Combinaison de Bits	Couleur Sélectionnée
000	Mode Transparent
001	COLOR09
010	COLOR10
011	COLOR11
100	COLOR12
101	COLOR13
110	COLOR14
111	COLOR15

La combinaison 000 sélectionne le mode transparent, pour afficher la couleur de n'importe quel autre objet (une autre aire de jeu, un sprite ou la couleur d'arrière-plan) pouvant se trouver "sous" l'aire de jeu.

Le Tableau 3-12 montre les registres de couleur en haute résolution, en mode double aire de jeu.

Tableau 3-12 : Registres de Couleur des Aires de Jeu n° 1 et n° 2 en Mode Haute Résolution.

AIRE DE JEU n° 1

Combinaison de Bits	Couleur Sélectionnée
00	Mode Transparent
01	COLOR1
10	COLOR2
11	COLOR3

AIRE DE JEU n° 2

Combinaison de Bits	Couleur Sélectionnée
00	Mode Transparent
01	COLOR09
10	COLOR10
11	COLOR11

CONTROLE ET PRIORITE D'UNE DOUBLE AIRE DE JEU

L'aire de jeu n° 1 ou l'aire de jeu n. 2 peuvent être l'une et l'autre prioritaires; ce qui signifie que n'importe laquelle des deux pourra s'afficher au-dessus de l'autre. Normalement, l'aire de jeu n° 1 est celle qui est prioritaire. Pour contrôler la priorité, on utilise le bit PF2PRI (bit n° 6) du registre BPLCON2. Lorsque PF2PRI = 1, l'aire de jeu n° 2 est prioritaire sur l'aire n° 1. Lorsque PF2PRI = 0, c'est l'aire de jeu n° 1 qui est prioritaire.

On peut aussi contrôler la priorité relative des aires de jeu et des sprites. Le Chapitre 7, "'Hardware' de Contrôle du Système" montre comment contrôler la priorité de ces objets.

On peut contrôler les deux aires de jeu séparément selon les critères suivants :

- Ces deux aires peuvent avoir une représentation en mémoire d'une taille différente et il est possible de sélectionner des parties de l'une ou de l'autre pour les afficher.

- On peut faire défiler séparément les deux aires de jeu.

NOTE

On devra faire très attention en faisant défiler une aire de jeu pendant que l'autre reste immobile. Quand on fait défiler des aires de jeu en basse résolution, on doit prendre un mot supplémentaire par rapport à la largeur de l'aire de jeu qu'on essaie de faire défiler (deux mots supplémentaires en haute résolution) pour fournir au système des données à afficher lors de la mise en place du véritable déroulement. Il n'y a que deux seuls registres disponibles : celui de début et celui de fin d'acquisition des données, et les deux aires de jeu se les partagent. Si on souhaite faire défiler une aire de jeu et garder l'autre immobile, on doit régler le début et la fin d'acquisition des données pour pouvoir gérer le déroulement des aires de jeu. Ensuite, on doit régler le modulo et les pointeurs de plans de bits de l'aire de jeu qu'on ne souhaite pas faire défiler pour que celle-ci conserve sa position pendant l'affichage. En basse résolution, on règle les pointeurs à -2 et le modulo à -2. En haute résolution, on règle les pointeurs à -4 et le modulo à -4.

ACTIVATION DU MODE DOUBLE AIRE DE JEU

Le fait d'écrire un 1 dans le bit 10 (DBLPF) du registre de contrôle BPLCON0 de l'aire de jeu du plan de bits sélectionne le mode double aire de jeu. La sélection de ce mode modifie, d'une part, la façon dont le 'hardware' regroupe les plans de bits pour l'interprétation de la couleur -tous les plans de bits de numéro impair sont regroupés ensemble, ainsi que tous les plans de bits de numéro pair- et modifie, d'autre part, la façon dont le 'hardware' peut déplacer les plans de bits à l'écran.

RECAPITULATION SUR LES AIRES DE JEU DOUBLES

La procédure à suivre pour définir des doubles aires de jeu est presque la même que celle nécessaire à la définition d'une aire de jeu simple. Lors du processus de création d'une aire de jeu double, par rapport à celui de création d'une aire de jeu simple, seules les étapes ci-dessous diffèrent :

- Le chargement des couleurs dans les registres. N'oubliez pas que les registres de couleur 0-7 sont utilisés par l'aire de jeu n° 1 et que les registres de 8 à 15 sont utilisés par l'aire de jeu n° 2 (si chaque aire de jeu possède trois plans de bits).

- La composition des plans de bits. Rappelez-vous que l'aire de jeu n° 1 est constituée des PLANS n° 1, n° 3 et n° 5, tandis que l'aire de jeu n° 2 est composée à partir des PLANS n° 2, n° 4 et n° 6.

- Le réglage des registres modulo. N'oubliez pas d'écrire la valeur modulo à la fois dans BPLlMOD et dans BPL2MOD, si vous voulez vous servir en même temps des plans de bits de numéro pair et de ceux de numéro impair.

Aux étapes ci-dessus viennent s'ajouter les suivantes :

- La définition de la priorité. Si on souhaite que l'aire de jeu n° 2 soit prioritaire, mettez à 1 le bit 6 (PF2PRI) de BPLCON2.

- L'activation du mode double aire de jeu. Mettre à 1 le bit 10 (DBLPF) de BPLCON0.

PLANS DE BITS ET FENETRES D'AFFICHAGE DE TOUTES LES DIMENSIONS

Nous avons vu comment construire des aires de jeu simples et doubles quand l'aire de jeu en mémoire a les mêmes dimensions que la fenêtre d'affichage. Cette section montrera comment définir et utiliser une aire de jeu dont la grande image en mémoire est plus grande que la fenêtre d'affichage, comment définir des fenêtres d'affichage de dimensions plus grandes ou plus petites que celles d'une aire de jeu normale et comment déplacer la fenêtre d'affichage dans cette grande image.

QUAND LA GRANDE IMAGE A UNE TAILLE PLUS GRANDE QUE LA FENETRE D'AFFICHAGE

Si on crée une image en mémoire plus grande que la fenêtre d'affichage, on devra choisir quelle partie de celle-ci on veut afficher. Le fait d'afficher une section plus importante de l'aire de jeu diffère de l'affichage des aires de jeu simples décrites jusqu'à maintenant sur les points suivants :

- Si la grande image en mémoire est plus grande que la fenêtre d'affichage, on devra spécifier à nouveau les valeurs modulo. On devra indiquer des valeurs quelconques non nulles.

- Il faudra réserver plus de mémoire pour une image mémoire plus importante.

SPECIFICATION DU MODULO

Pour une image mémoire plus grande que la fenêtre d'affichage, on devra spécifier à nouveau la valeur modulo pour pouvoir saisir les bons mots de données pour chaque ligne de l'affichage. A titre d'exemple, supposons que la fenêtre d'affichage ait une largeur standard de 320 pixels, ce qui donne 40 octets par ligne à afficher. Cependant, par rapport à la fenêtre d'affichage, la grande image en mémoire fait le double en largeur, c'est-à-dire qu'elle a une largeur de 80 octets. Présumons, en outre, qu'on veuille afficher la moitié gauche de la grand image. La Planche 3-14 montre la relation existant entre la grande image et l'image qui sera affichée.

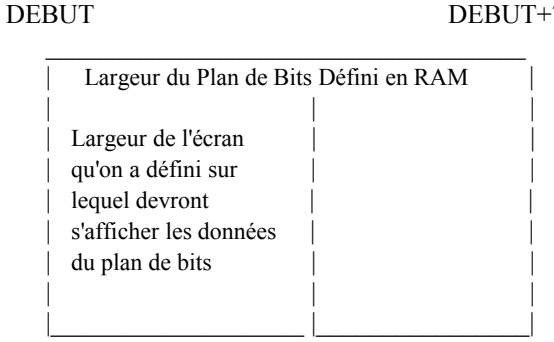


PLANCHE 3-14 : Image Mémoire plus Grande que l'Affichage

Etant donné qu'on devra prélever 40 octets par ligne, l'acquisition des données pour la ligne n° 1 se fera comme nous le montrons dans la Planche 3-15.

Données de la Ligne n° 1 :

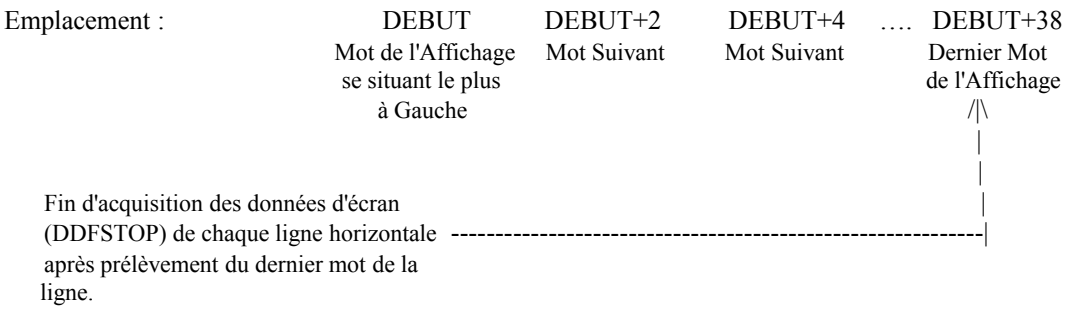


Planche 3-15 : Acquisition des Données de la Première Ligne pour Modulo = 40

A ce moment, BPLxPTH et BPLxPTL contiennent la valeur DEBUT+40. Cette valeur modulo de 40, sera additionnée à la valeur courante du pointeur afin de pouvoir prélever les données souhaitées pour cette ligne, quand commencera l'acquisition de données de la ligne suivante.

Données de la Ligne n° 2 :

Emplacement :	DEBUT+80	DEBUT+82	DEBUT+84	….	DEBUT+118
	Mot de l'Affichage se situant le plus à Gauche	Mot Suivant	Mot Suivant		Dernier Mot de l'Affichage

Planche 3-16 : Acquisition des Données de la Deuxième Ligne pour Modulo = 40

Pour pouvoir afficher la moitié droite de la grande image, on met en place une routine de suppression verticale pour déclencher les pointeurs de plans de bits à l'adresse DEBUT+40 plutôt qu'à l'adresse DEBUT avec un modulo qui reste à 40. Nous montrons l'organisation des données dans les Planches 3-17 et 3-18.

Données de la Ligne n° 1 :

Emplacement :	DEBUT+40	DEBUT+42	DEBUT+44	….	DEBUT+78
	Mot de l'Affichage se situant le plus à Gauche	Mot Suivant	Mot Suivant		Dernier Mot de l'Affichage

Planche 3-17 : Organisation des Données de la Moitié Droite de la Première ligne de la Grande Image.

Maintenant, les pointeurs de plans de bits contiennent la valeur DEBUT+80. On additionne le modulo (40) aux pointeurs pour pouvoir prélever les données courantes, quand les pointeur si commenceront leur prélèvement de données dans la deuxième ligne.

Données de la Ligne n° 2 :

Emplacement :	DEBUT+120	DEBUT+122	DEBUT+124	….	DEBUT+38
	Mot de l'Affichage se situant le plus à Gauche	Mot Suivant	Mot Suivant		Dernier Mot de l'Affichage

Planche 3-18 : Organisation des Données de la Moitié Droite de la Deuxième ligne de la Grande Image.

Souvenez-vous qu'en mode haute résolution, il faudra prélever deux fois plus d'octets qu'en mode basse résolution. Pour un affichage de dimensions normales, on prélève 80 octets par ligne horizontale, au lieu de 40.

SPECIFICATIONS POUR LE PRELEVEMENT DES DONNEES

Pour le placement des données dans chaque ligne horizontale de l'affichage, les registres d'acquisition des données indiquent la position de début et celle de fin. On indique l'acquisition des données comme nous l'avons montré dans la section "Construction d'une Aire de Jeu Simple".

RESERVATION DE MEMOIRE

Pour des images mémoire plus grandes, il faudra réserver plus de mémoire. Voici une formule générale pour calculer les besoins en mémoire :

octets par ligne * lignes de l'aire de jeu * # de plans de bits

Ainsi, si l'aire de jeu décrite dans cette section est composée de deux plans de bits, il lui faudra :

80 * 200 * 2 = 32000 octets de mémoire

Rappelez-vous qu'il s'agit de besoins en mémoire pour une seule aire de jeu. Il faudra de la mémoire supplémentaire pour les sprites, l'animation, l'audio ou les programmes d'application qu'on utilise.

SELECTION DU POINT D'ORIGINE DE LA FENETRE D'AFFICHAGE

Le point d'origine de la fenêtre d'affichage est celui indiqué par les coordonnés horizontale et verticale du coin supérieur gauche de la fenêtre elle-même. Un seul registre, DIWSTRT, contient les deux coordonnés verticale et horizontale, c'est-à-dire HSTART et VSTART. Les huit bits alloués à HSTART sont assignés aux 256 premières positions à partir de la position la plus à gauche possible. Par conséquent, il est possible de faire débuter la fenêtre d'affichage à une position de pixel quelconque dans cet intervalle.

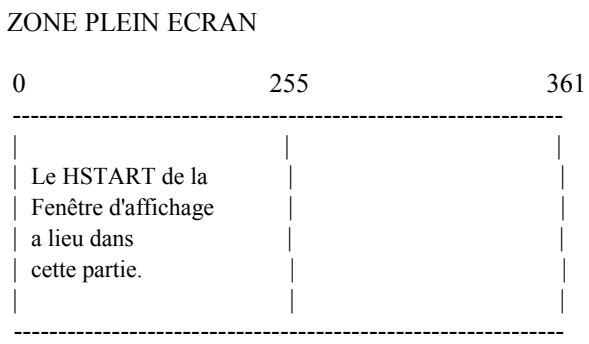


PLANCHE 3-19 : Position du Point d'Origine Horizontal de la Fenêtre d'Affichage

Les huit bits alloués à VSTART sont assignés aux 256 premières positions à partir de la section supérieure de l'affichage.

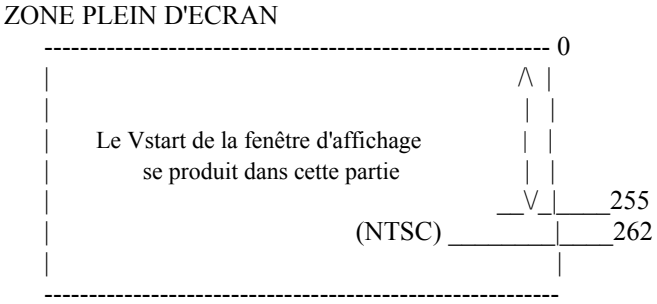


PLANCHE 3-20 : Position du Point d'Origine Vertical de la Fenêtre d'Affichage

Rappelez-vous qu'on sélectionne les valeurs du point d'origine comme si l'affichage était en basse résolution, en mode non entrelacé. Gardez à l'esprit, cependant, qu'en mode entrelacé, la fenêtre d'affichage devrait avoir, en hauteur, un nombre de lignes paires pour pouvoir posséder des champs pairs et des champs impairs compensés.

Pour fixer le point d'origine de la fenêtre d'affichage, écrire la valeur de HSTART dans les bits de 0 à 7 et celle de VSTART dans les bits de 7 à 15 de DIWSTRT.

SELECTION DU POINT D'ARRIVEE

La point d'arrivée de la fenêtre d'affichage est celui indiqué par les coordonnés horizontale et verticale du coin inférieur droit de la fenêtre elle-même. Un seul registre, DIWSTOP, contient les deux coordonnés verticale et horizontale, c'est-à-dire HSTOP et VSTOP.

Cf. les notes de la section "Construction d'une Aire de Jeu Simple" pour des instructions sur la mise en place de ces registres.

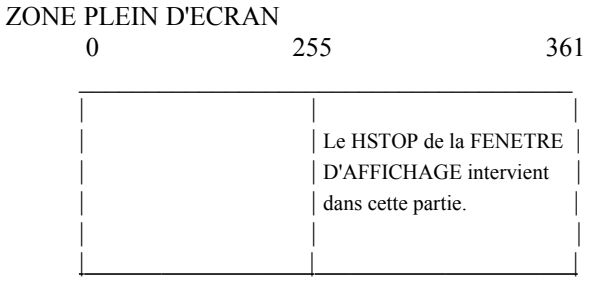


PLANCHE 3-21: Position du Point d'Arrivée Horizontal de la Fenêtre d'Affichage

Sélectionnez une valeur qui représente la bonne position en basse résolution et en mode non entrelacé.

ZONE PLEIN D'ECRAN

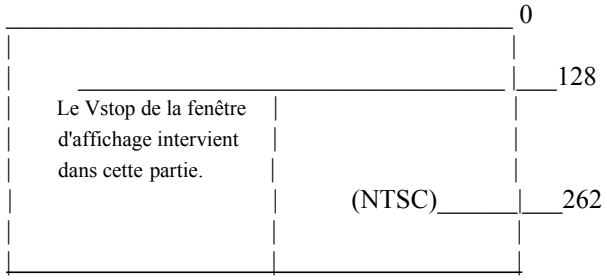


PLANCHE 3-20 : Position du Point d'Arrivée Vertical de la Fenêtre d'Affichage

Pour fixer le point d'arrivée de la fenêtre d'affichage, écrire HSTOP dans les bits de 0 à 7 et VSTOP dans les bits de 8 à 15 de DIWSTOP.

DIMENSIONS MAXIMALES DE LA FENETRE D'AFFICHAGE

Les dimensions maximales d'affichage d'une aire de jeu sont déterminées par le nombre maximal de lignes et de colonnes. Sur le plan vertical, les limitations sont évidentes. On ne peut afficher aucune donnée dans la zone de suppression verticale. Le tableau qui suit indique la zone d'affichage verticale disponible.

Tableau 3-13 : Zone Verticale Maximale Disponible sur l'Ecran Vidéo

Vertical Blank	NTSC	PAL
Start	0	0
Stop	$15 (21)	$1D (29)

	NTSC Normales	NTSC Entrelacées	PAL Normales	PAL Entrelacées
Lignes affichables sur l'écran vidéo	241	483 =525-(21*2)	283	567 =625-(29*2)

Sur le plan horizontal, la situation est similaire. A strictement parler, le 'hardware' impose à droite, à DDFSTOP,une limite extrême de ($D8) et, à gauche, une limite extrême de ($18). Ce qui donne une acquisition maximale de 25 mots en mode basse résolution. En mode haute résolution, le nombre maximal sera dans ce cas de 49 mots car, à droite, la limite extrême reste à ($D8) et qu'on prélève un seul mot dans cette limite.

Cependant, en réalité, la suppression horizontale limite l'écran vidéo affichable à 368 pixels en basse résolution (23 mots).
Ces chiffres sont les mêmes en NTSC et en PAL. En outre, on devrait noter qu'en mettant le début d'acquisition des données avant ($38), on pourrait invalider des sprites.

Tableau 3-14 : Zone Horizontale Maximale Disponible sur l'Ecran Vidéo

	Basse Résolution	Haute Résolution
DDFSTRT (standard)	$0038	$003C
DDFSTOP (standard)	$00D0	$00d4
DDFSTRT (limites 'hardware')	$0018	$0018
DDFSTOP (limites 'hardware')	$00D8	$00D8
acquisition de mots max.	25	49
pixels d'affichage max.	368 (low res.)	

DEPLACEMENT (DEROULEMENT) DES AIRES DE JEU

Si vous souhaitez avoir un affichage qui se déplace à l'arrière-plan, vous pourrez envisager une aire de jeu plus grande que la fenêtre d'affichage et la faire défiler. Si vous êtes en train d'utiliser des aires de jeu doubles, vous pourrez les faire défiler séparément.

A l'écran, dans un déroulement vertical, l'aire de jeu paraît se déplacer lentement vers le haut ou vers le bas. Tout ce qu'on aura à faire pour avoir un déroulement vertical sera d'augmenter ou de diminuer progressivement les adresses de départ des pointeurs de plans de bits de la taille de la ligne horizontale de l'aire de jeu. Ce qui aura pour effet d'afficher la partie supérieure ou inférieure du champ de l'image à chaque fois.

A l'écran, dans un déroulement horizontal, l'aire de jeu paraît se déplacer de droite à gauche ou de gauche à droite. Le déroulement horizontal fonctionne d'une manière différente du déroulement vertical car on doit se débrouiller pour prélever un mot de données supplémentaire à chaque ligne d'affichage et pour retarder l'affichage de ces données.

Pour d'autres types de déroulement, la réinitialisation des pointeurs ou les registres d'acquisition de données pourront être gérés par le Copper pendant l'intervalle de suppression verticale.

DEROULEMENT VERTICAL

Dans une fenêtre, on a la possibilité de faire défiler une aire de jeu vers le haut ou vers le bas. A chaque fois qu'on affiche une aire de jeu, les pointeurs de plans de bits commencent à un endroit progressivement plus grand ou plus petit de la grande image présente en mémoire. A mesure que la valeur du pointeur augmente, le système affiche une portion plus grande de la partie inférieure de l'image et l'image semble défiler vers le haut. A mesure que la valeur du pointeur diminue, le système affiche une section plus grande de la partie supérieure de l'image et celle-ci défile vers le bas.

Sur un système NTSC, avec un affichage de 200 lignes, chaque pas de déroulement pourra être aussi petit qu'1/200ème de l'écran. En mode entrelacé, chaque pas de déroulement pourra être égale à 1/400ème de l'écran en manipulant intelligemment les pointeurs, mais il est recommandé de faire un déroulement de deux lignes à la fois pour conserver la relation entre champs impairs/pairs. En utilisant un système PAL avec un affichage de 200 lignes, on pourra avoir un pas de déroulement d'1/256ème ou, en mode entrelacé, d'1/512ème de l'écran.

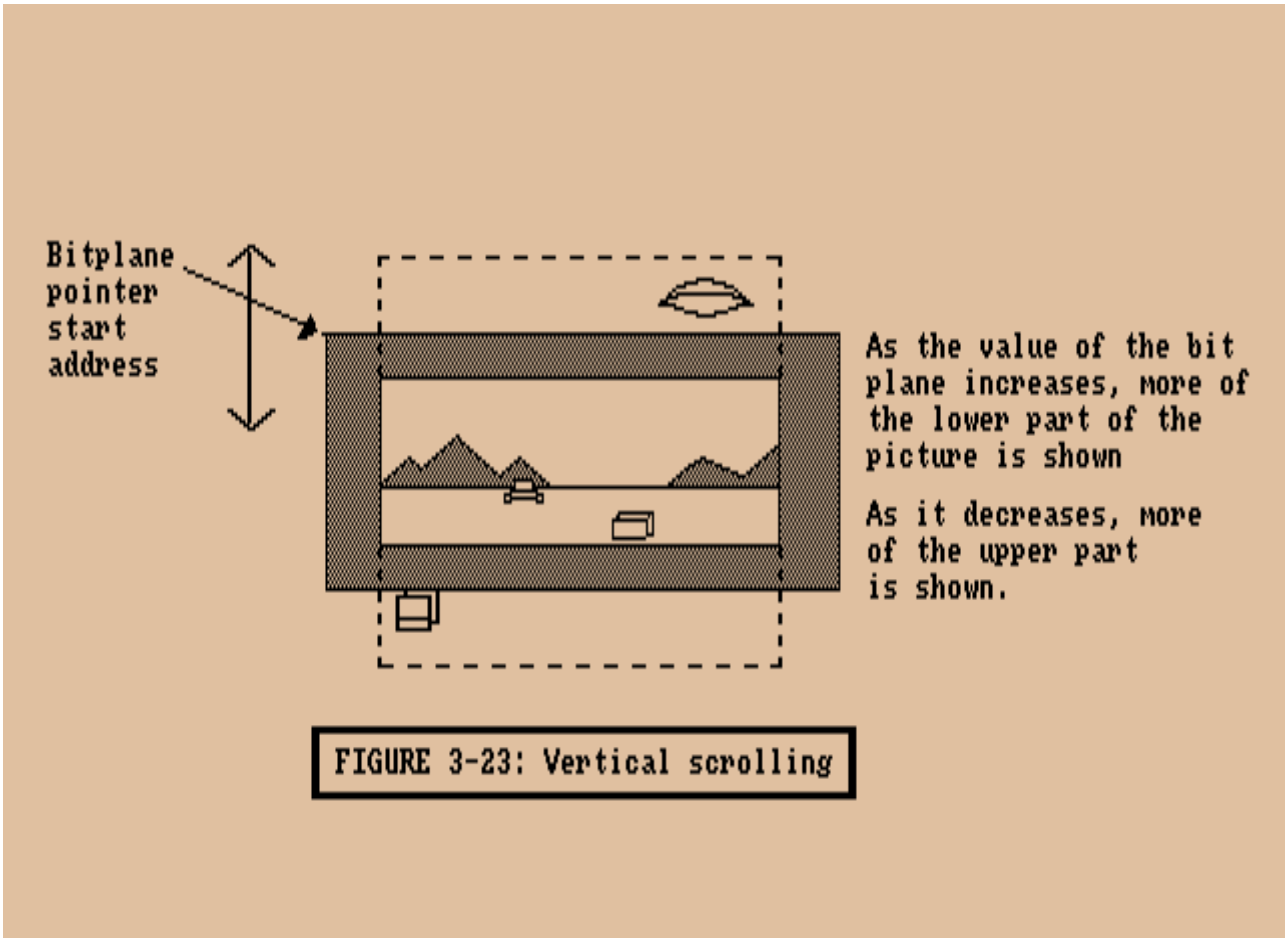


PLANCHE 3-23 : Déroulement Vertical

Pour mettre en place une aire de jeu pour un déroulement vertical, on devra créer des plans de bits suffisamment petits qu'ils puissent permettre une amplitude de déroulement telle qu'on la souhaite, on devra écrire un programme qui calcule les pointeurs de plans de bits pour le déroulement voulu et donner l'autorisation au Copper d'utiliser ces pointeurs.

Présumons qu'on veuille faire défiler une aire de jeu d'une seule ligne à la fois. Pour réaliser ceci, avant que chaque champ ne soit affiché, les pointeurs de plans de bits devront augmenter suffisamment pour être sûrs que chaque fois les pointeurs commencent une ligne plus bas. Pour un affichage en basse résolution, de dimensions normales, avec modulo 0, les pointeurs subiraient chaque fois une augmentation de 40 octets.

DEROULEMENT HORIZONTAL

A l'écran, on peut faire défiler des aires de jeu de gauche à droite et de droite à gauche. On contrôle la vitesse de défilement en indiquant la durée du délai en pixels. Délai signifie qu'on prélève un mot de données supplémentaire et qu'on ne l'affiche pas immédiatement. Ce mot en plus est placé à la gauche du bord à l'extrémité gauche de la fenêtre, avant le prélèvement normal des données. Dès que l'affichage se décale sur la droite, les bits de ce mot supplémentaire apparaissent à l'écran du côté gauche de la fenêtre à mesure que les bits du côté droit disparaissent en dehors de l'écran. Pour chaque pixel de délai, les données à l'écran décalent un pixel sur la droite à chaque champ d'affichage. En mode haute résolution, le déroulement se fera par une augmentation de deux pixels. La Planche 3-34 montre comment le délai et le prélèvement de données supplémentaires se combinent pour produire un effet de déroulement.

Pour mettre en place une aire de jeu et obtenir un déroulement horizontal, on devra

- o Définir des plans de bits suffisamment larges pour obtenir le déroulement voulu.
- o Initialiser les registres d'acquisition des données pour que chaque ligne horizontale, y compris le mot supplémentaire, soit placée correctement à l'écran.
- o Mettre les bits de délai.
- o Mettre en place le modulo pour que -pour chaque ligne- les pointeurs de plans de bits se trouvent sur le mot juste.
- o Ecrire les instructions du Copper pour gérer les modifications lors de l'intervalle de suppression verticale.

SPECIFICATION DE L'ACQUISITION DES DONNEES LORS D'UN DEROULEMENT HORIZONTAL

Le début normal d'acquisition de données pour des affichages qui ne défilent pas se situe à ($38). Si on veut avoir un déroulement horizontal, alors l'acquisition des données devra commencer un mot plus tôt (DDFSTRT = $0030). Incidemment, ceci invalide le sprite n° 7. DDFSTOP reste inchangé. N'oubliez pas que la mise en place des registres d'acquisition des données affecte les deux aires de jeu.

SPECIFICATION DU MODULO LORS D'UN DEROULEMENT HORIZONTAL

Comme toujours, le modulo aura deux unités de moins que la différence entre l'adresse du prochain mot qu'on veut prélever et l'adresse du dernier mot qu'on a prélevé. A titre d'exemple, pour un déroulement horizontal, prenons un affichage de 40 octets dans une "grande image" de 80 octets. Etant donné que le déroulement horizontal nécessite une acquisition de données de deux mots supplémentaires, les données de chaque ligne auront une longueur de 42 octets.

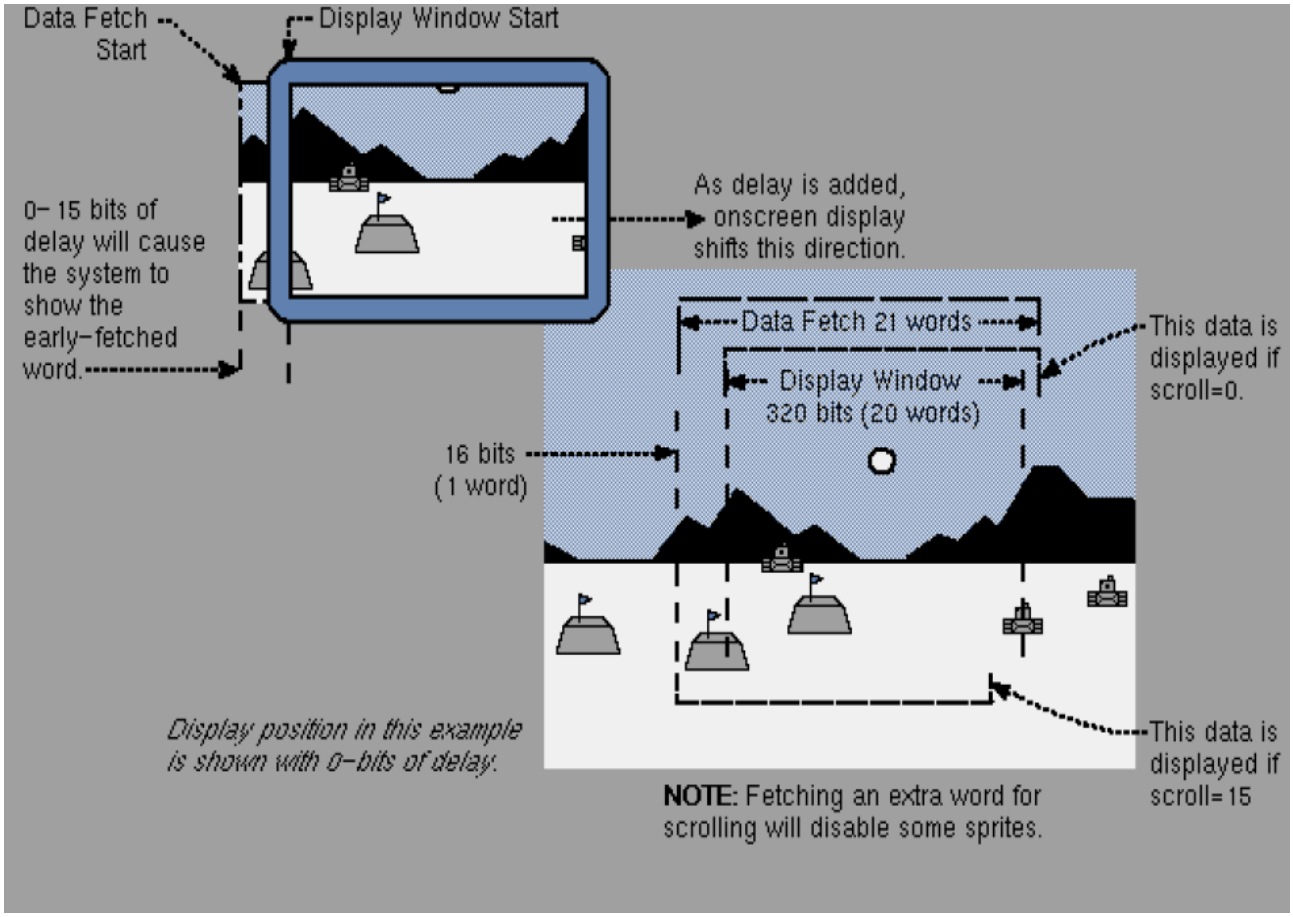


PLANCHE 3-24 : Déroulement Horizontal

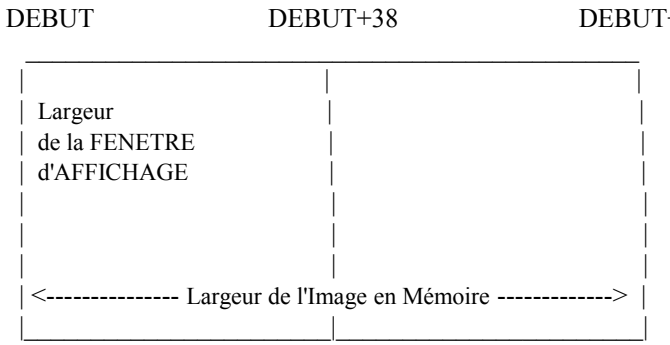


Planche 3-25 : Image en Mémoire plus Grande que la Fenêtre d'Affichage

Données de la Ligne n° 1 :

Emplacement :	DEBUT	DEBUT+2	DEBUT+4	….	DEBUT+40
	Mot de l'Affichage se situant le plus à Gauche	Mot Suivant	Mot Suivant		Dernier Mot de l'Affichage

Planche 3-26 : Données de la Ligne n° 1 - Déroulement Horizontal

A ce moment, les pointeurs de plans de bits contiennent la valeur DEBUT+42. En lui additionnant un modulo de 30, on obtient un point initial correct pour la prochaine ligne.

Données de la Ligne n° 2 :

Emplacement :	DEBUT+80	DEBUT+82	DEBUT+84	....	DEBUT+120
	Mot de l'Affichage se situant le plus à Gauche	Mot Suivant	Mot Suivant		Dernier Mot de l'Affichage

Planche 3-27 : Données de la Ligne n° 2 - Déroulement Horizontal

Dans les registres BPLxMOD, on mettra en place un modulo pour chaque plan de bits utilisé.

SPECIFICATION DE LA DUREE DU DELAI

La durée du délai lors d'un déroulement horizontal est contrôlée par les bits 7-0 de BPLCON1. Pour chaque plan de bits, on met en place un délai de manière séparée; les bits 3-0 pour l'aire de jeu n° 1 (plans de bits n° 1, n° 3 et n° 5) et les bits 7-4 pour l'aire de jeu n° 2 (plans de bits n°2, n° 4 et n° 6).

NOTE

Mettez toujours les six bits, même si vous n'avez qu'une seule aire de jeu. Mettez 3-0 et 7-4 à la même valeur si vous n'utilisez qu'une seule aire de jeu.

L'exemple suivant fixe le délai du déroulement horizontal à 7 pour les deux aires de jeu.

```
MOVE.W #$77,BPLCON1+CUSTOM
```

RECAPITULATION SUR LES AIRES DE JEU QU'ON VEUT FAIRE DEFILER

La procédure à suivre pour définir une aire de jeu qui devra défiler est la même que celle employée pour définir une aire de jeu simple; seules les étapes ci-dessous diffèrent :

- La définition de l'acquisition des données. Prélever un mot supplémentaire par ligne horizontale et faire commencer la ligne 16 pixels avant le début normal (sans déroulement) d'acquisition des données.

- La définition du modulo. La valeur de ce dernier a deux unités de moins par rapport à la valeur qu'il aurait quand il n'y a pas de déroulement.

Aux étapes ci-dessus viennent s'ajouter les suivantes :

- Pour le déroulement vertical, réinitialiser les pointeurs de plans de bits pour le pas d'augmentation du déroulement lui-même. Réinitialiser BPLxPTH et BPLxPTL pendant l'intervalle de suppression verticale.

- Pour le déroulement horizontal, indiquez le délai. Mettre les bits 7-0 de BPLCON1 pour les bits de délai de 0 à 15.

SUJETS AVANCES

Cette section décrit des particularités qu'on emploie moins souvent ou qui existent en option.

INTERACTIONS ENTRE LES AIRES DE JEU ET LES AUTRES OBJETS

Les aires de jeu se partagent l'affichage avec les sprites. Le Chapitre 7, "'Hardware' de Contrôle du Système" explique comment il est possible de donner des priorités d'affichage vidéo différentes aux aires de jeu qui sont en relation avec des sprites et comment les aires de jeu peuvent entrer en collision avec les sprites et les unes avec les autres.

MODE 'HOLD-AND-MODIFY'

C'est un mode spécial qui permet d'afficher à l'écran jusqu'à 4096 couleurs en même temps. Normalement, dès qu'on sélectionne une valeur constituée d'une combinaison de plans de bits, les données présentes dans le registre de couleur sélectionné sont chargées dans les circuits de couleur de sortie pour que le pixel soit écrit à l'écran. Par conséquent, chaque pixel aura la couleur qui se trouve dans le registre de couleur sélectionné.

Cependant, en mode 'hold-and-modify', on retient la valeur des circuits de couleur de sortie et on en modifie l'une des trois composantes (rouge, verte ou bleue) par le biais des bits en provenance de certains plans de bits pré-sélectionnés. Après modification, on écrit le pixel à l'écran.

Le mode 'hold-and-modify' permet de produire à l'écran des gradients de couleur très fins ou des hachures. Par exemple, on pourrait dessiner un jeu de 16 vases, chacun d'une couleur différente, en employant toutes les 16 couleurs de la palette de couleur. Ensuite, pour chaque vase, on utiliserait le mode 'hold-and-modify' pour ombrer plus finement chaque vase ou pour le rehausser ou pour lui ajouter une couleur totalement différente. Remarquez que chaque pixel 'hold-and-modify' ne peut modifier qu'une des trois valeurs de couleur à la fois. Par conséquent, cet effet n'autorise qu'un contrôle limité.

En mode 'hold and modify', on utilise tous les six plans de bits. Les plans 5 et 6 sont utilisés pour modifier la manière dont seront traités les bits des plans 1 - 4, de la façon suivante :

- Si la combinaison de bits 6-5 pour un pixel donné est égal à 00, on suivra la procédure normale de sélection de couleur. Ainsi, les combinaisons de bits des plans 4 - 1, dans cet ordre d'importance, seront utilisées pour choisir un des 16 registres de couleur (registres 0 - 15).

- Si l'on n'utilise que cinq plans de bits, le système attribuera aux données du sixième plan une valeur de 0 automatiquement.

- Si la combinaison de bits est égale à 01, la couleur du pixel se trouvant tout de suite à gauche après ce pixel sera dupliquée et ensuite modifiée. Les combinaisons de bits des plans 4 - 1 sont utilisées pour remplacer les quatre bits "bleus" du registre de couleur correspondant.

o Si la combinaison de bits est égale à 10, la couleur du pixel se trouvant tout de suite à gauche après ce pixel sera dupliquée et ensuite modifiée. Les combinaisons de bits des plans 4 - 1 sont utilisées pour remplacer les quatre bits "rouges".

o Si la combinaison de bits est égale à 11, la couleur du pixel se trouvant tout de suite à gauche après ce pixel sera dupliquée et ensuite modifiée. Les combinaisons de bits des plans 4 - 1 sont utilisées pour remplacer les quatre bits "verts".

En utilisant le mode 'hold-and-modify', il est possible de traiter avec un registre qui détermine une seule couleur, c'est-à-dire COLOR0, la couleur d'arrière-plan. On traite tout l'écran en tant que modification de cette couleur d'origine, en accord avec le schéma ci-dessus.

Le bit 11 du registre BPLCON0 sélectionne le mode 'hold-and-modify'. Voici les bits où il faut écrire pour valider le mode 'hold-and-modify' :

o Bit HOMOD, bit 11, est à 1.

o Bit DBLPF, bit 10, est à 0 (indication du mode aire de jeu simple).

o Bit HIRES, bit 15, est à 0 (indication du mode basse résolution).

o Les Bits BPU2, BPUI et BPU0 - bits 14, 13 et 12 sont à 101 ou à 110 (validation de cinq ou six plans de bits).

Le programme d'exemple ci-dessous génère un affichage avec six plans de bits en mettant en service le mode 'hold-and-modify'. Tous les 32 registres de couleur sont chargés avec le noir pour prouver que les couleurs seront bien générées par 'hold-and-modify'. Les 'equates' sont ceux que nous utilisons habituellement et, dans ce cas, nous ne les mettrons pas.

```
; D'abord, mettre en place les registres de contrôle.
;
   LEA    CUSTOM,a0                ; Faire pointer a0 sur les puces spécialisées
   MOVE.W #$6A00,BPLCON0(a0)       ; Six plans de bits, en mode 'hold-and-modify'
   MOVE.W #0,BPLCON1(a0)           ; Déroulement horizontal = 0
   MOVE.W #0,BPL1MOD(a0)           ; Modulo pour les plans de bits impairs = 0
   MOVE.W #0,BPL2MOD(a0)           ; Idem pour les plans de bits pairs
   MOVE.W #$0038,DDFSTRT(a0)       ; Fixer le début d'acquisition des données
   MOVE.W #$00D0,DDFSTOP(a0)       ; Fixer la fin d'acquisition des données
   MOVE.W #$2C81,DIWSTRT(a0)       ; Fixer le point d'origine de la fenêtre d'affichage
   MOVE.W #$F4C1,DIWSTOP(a0)       ; Fixer le point d'arrivée de la fenêtre d'affichage
;
; Mettre tous les registres de couleur = noir pour prouver que le mode
; 'hold-and-modify' est opérationnel
;
   MOVE.W #32,d0                   ; Initialiser le compteur
   LEA    CUSTOM+COLOR00,a1        ; Faire pointer al sur le premier registre de couleur
CREGLOOP:
   MOVE.W #$0000,(a1)+             ; Ecrire noir dans un registre de couleur
   DBRA   d0,CREGLOOP              ; Diminuer le compteur d'un pas et boucler jusqu'à ce que ce soit fini
;
; Remplir six plans de bits avec un motif facilement reconnaissable.
;
; NOTE : Nous ne faisons ceci qu'à titre d'exemple. Normalement, pour ces
;        plans de bits, on devrait réserver la place en mémoire à partir
;        de la mémoire MEMF_CHIP libre du système.
;
```

```
   MOVE.W  #2000,d0              ; 2000 mots longs par plan de bits
   MOVE.L  #$21000,a1            ; Faire pointer a1 sur le plan n° 1
   MOVE.L  #$23000,a2            ; Faire pointer a2 sur le plan n° 2
   MOVE.L  #$25000,a3            ; Faire pointer a3 sur le plan n° 3
   MOVE.L  #$27000,a4            ; Faire pointer a4 sur le plan n° 4
   MOVE.L  #$29000,a5            ; Faire pointer a5 sur le plan n° 5
   MOVE.L  #$2B000,a6            ; Faire pointer a6 sur le plan n° 6
FPLLOOP:
   MOVE.L  #$55555555,(a1)+      ; Remplir le plan n° 1 avec $55555555
   MOVE.L  #$33333333,(a2)+      ; Remplir le plan n° 2 avec $33333333
   MOVE.L  #$0F0F0F0F,(a3)+      ; Remplir le plan n° 3 avec $0F0F0F0F
   MOVE.L  #$00FF00FF,(a4)+      ; Remplir le plan n° 4 avec $00FF00FF
   MOVE.L  #$CF3CF3CF,(a5)+      ; Remplir le plan n° 5 avec $CF3CF3CF
   MOVE.L  #$3CF3CF3C,(a6)+      ; Remplir le plan n° 6 avec $3CF3CF3C
   DBRA    d0,FPLLOOP            ; Diminuer le compteur d'un pas et boucler jusqu'à ce que ce soit fini
;
; Mettre en place la liste Copper à l'adresse $20000.
;
; NOTE : De même que pour les plans de bits, la liste Copper devrait être
;            réservée à partir de la mémoire MEMF_CHIP libre du système.
;
   MOVE.L #$20000,a1             ; Faire pointer al sur destination de la liste Copper
   LEA    COPPERL(pc),a2         ; Faire pointer a2 sur l'image de la liste Copper
CLOOP:
   MOVE.L (a2),(a1)+             ; Transférer un mot long
   CMPI.L #$FFFFFFFE,(a2)+       ; Tester la fin de la liste Copper
   BNE    CLOOP                  ; Boucler jusqu'au transfert total de la liste Copper
;
; Faire pointer le Copper sur la liste Copper
;
   MOVE.L #$20000,COP1LCH(a0   ; Charger les registres de saut du Copper
   MOVE.W COPJMP1(a0),d0         ; Forcer le chargement dans le P.C. Copper (compteur ordinal)
;
; Lancer les DMA
;
   MOVE.W #$8380,DMACON(a0)       ; Valider le plan de bits et le DMA du Copper

   BRA ....partie qui reste à faire....
;
; Liste Copper pour six plans de bits.
; Le plan n° 1 se trouve à l'adresse $21000
; le 2 se trouve à l'adresse $23000
; le 3 se trouve à l'adresse $25000
; le 4 se trouve à l'adresse $27000
; le 5 se trouve à l'adresse $29000
; le plan 6 se trouve à l'adresse $2B000.
;
; NOTE : Les adresses de ces plans de bits sont valables seulement à titre
;            d'exemple. Cf. la note ci-dessus.
;
COPPERL:
   DC.W   BPL1PTH,$0002          ; Pointeur sur le plan n° 1 = $21000
   DC.W   BPL1PTL,$1000
   DC.W   BPL2PTH,$0002          ; Pointeur sur le plan n° 2 = $23000
   DC.W   BPL2PTL,$3000
   DC.W   BPL3PTH,$0002          ; Pointeur sur le plan n° 3 = $25000
   DC.W   BPL3PTL,$5000
   DC.W   BPL4PTH,$0002          ; Pointeur sur le plan n° 4 = $27000
   DC.W   BPL4PTL,$7000
   DC.W   BPL5PTH,$0002          ; Pointeur sur le plan n° 5 = $29000
   DC.W   BPL5PTL,$9000
   DC.W   BPL6PTH,$0002          ; Pointeur sur le plan n° 6 = $2B000
   DC.W   BPL6PTL,$B000
   DC.W   $FFFF,$FFFE            ; Attendre l'impossible, c'est-à-dire quitter
```

CREATION D'UN AFFICHAGE AVEC PLUSIEURS AIRES DE JEU DIFFERENTES

La bibliothèque 'graphics.library' offre la possibilité de partager l'écran en différents "ViewPorts", chacun avec ses propres couleurs et sa propre résolution. Pour plus de renseignements, consultez l' "Amiga ROM Kernel Manual".

UTILISATION D'UNE SOURCE VIDEO EXTERNE

Une carte en option disposant d'un 'genlock' est disponible pour l'Amiga. Le 'genlock' vous permet d'introduire dans votre affichage graphique des images en provenance d'une source vidéo externe (comme celle d'un magnétoscope, d'une caméra vidéo ou d'un lecteur de vidéodisques). Quand on utilise un 'genlock', la couleur d'arrière-plan est remplacée par l'affichage en provenance de la source vidéo externe. Pour plus de renseignements, consultez les instructions fournies avec cette carte.

RECAPITULATION DES REGISTRES DES AIRES DE JEU

Cette section récapitule tous les registres utilisés dans ce chapitre et la signification de la mise en place de leurs bits. Les registres de couleur seront résumés dans la section suivante. Pour la récapitulation de tous les registres, voir l'Appendice A.

BPLCON0 - Contrôle du Plan de Bits

NOTE

On ne peut utiliser les bits de ce registre de manière indépendante.

Bit 0 - inutilisé

Bit 1 - ERSY (valide la synchronisation externe)
1 = Synchronisation externe validée (permet la production de la synchronisation du 'genlock')
0 = Synchronisation externe invalidée

Bit 2 - LACE (valide le mode entrelacé)
1 = mode entrelacé validé
0 = mode non entrelacé validé

Bit 3 - LPEN (valide le crayon optique)

Bits 4-7 inutilisés (faire 0)

Bit 8 - GAUD (valide l'audio du 'genlock')
1 = Audio du 'genlock' validée
0 = Audio du 'genlock' invalidée (lors des périodes de suppression, ce bit ne travaille pas pendant la transmission du pixel)

Bit 9 - COLOR ON (valide la couleur)
1 = impulsion de synchro couleur pour vidéo composite validée
0 = impulsion de synchro couleur pour vidéo composite invalidée

Bit 10 - DBLPF (valide une double aire de jeu)
1 = aires de jeu doubles validées
0 = aire de jeu simple validée

Bit 11 - HOMOD (valide le mode 'hold-and-modify')
1 = 'hold-and-modify' validé
0 = 'hold-and-modify' invalidé

Bits 14, 13,12 - BPU2, BPU1, BPU0
Nombre de plans de bits utilisé

000 = seulement une couleur d'arrière-plan
001 = 1 plan de bits, PLAN 1
010 = 2 plans de bits, PLANS 1 et 2
011 = 3 plans de bits, PLANS 1- 3
100 = 4 plans de bits, PLANS 1- 4
101 = 5 plans de bits, PLANS 1- 5
110 = 6 plans de bits, PLANS 1- 6
111 inutilisé

Bit 15 - HIRES (valide la haute résolution)
1 = mode haute résolution
0 = mode basse résolution

BPLCON1 - Contrôle du Plan de Bits

Bits 3-0 - PF1H(3-0)
Délai pour l'aire de jeu n° 1

Bits 7-4 - PF2H(3-0)
Délai pour l'aire de jeu n° 2

Bits 15-8 inutilisés

BPLCON2 - Contrôle du Plan de Bits

Bit 6 - PF2PRI

1 = L'aire de jeu n° 2 est prioritaire
0 = L'aire de jeu n° 1 est prioritaire

Bits 0-5 Priorité des sprites de l'aire de jeu

Bits 7-15 inutilisés

DDFSTRT - Début d'Acquisition des Données
(Position initiale d'acquisition des données)

Bits 15-8 - inutilisés

Bits 7-2 - bits H8-H3 de position du pixel
le bit H3 n'est respecté qu'en mode haute résolution

Bits 1-0 - inutilisés

DDFSTOP - Fin d'Acquisition des Données
(Position finale d'acquisition des données)

Bits 15-8 - inutilisés

Bits 7-2 - bits H8-H3 de position du pixel
le bit H3 n'est respecté qu'en mode haute résolution

Bits 1-0 - inutilisés

BPLxPTH - Pointeur de Plans de Bits
(Mot de poids fort du pointeur de plans de bits, où x est le numéro du plan de bits)

BPLxPTL - Pointeur de Plans de Bits
(Mot de poids faible du pointeur de plans de bits, où x est le numéro du plan de bits)

DIWSTRT - Début de la Fenêtre d'Affichage
(Coordonnés verticale et horizontale du point d'origine)

Bits 15-8 - VSTART (V7-V0)
Bits 7-0 - HSTART (H7-H0)

DIWSTOP - Fin de la Fenêtre d'Affichage
(Coordonnés verticale et horizontale du point d'arrivée)

Bits 15-8 - VSTOP (V7-V0)

Bits 7-0 - HSTOP (H7-H0)

BPL1MOD - Modulo du Plan de Bits
(Plans de numéro impair, aire de jeu n° 1)

BPL2MOD - Modulo du Plan de Bits
(Plans de numéro pair, aire de jeu n° 2)

RECAPITULATION SUR LA SELECTION DES COULEURS

Cette section contient la récapitulation de la sélection des couleurs pour une aire de jeu, y compris le contenu des registres de couleur, des couleurs d'exemple et la différence qui existe entre la sélection de couleurs en mode basse résolution et en mode haute résolution.

CONTENU DES REGISTRES DE COULEUR

Le Tableau 3-15 indique le contenu de chaque registre de couleur. Ce sont des registres destinés seulement à l'écriture.

Tableau 3-15 : Contenu des Registres de Couleur

Bits	Contenu
15-12	(Inutilisés - mettre à 0)
11- 8	Rouge
7- 4	Vert
3- 0	Bleu

UNE SELECTION DU CONTENU DES REGISTRES DE COULEUR

Le Tableau 3-16 montre un choix de couleurs et les valeurs hexadécimales à charger dans les registres de couleur pour pouvoir les obtenir.

Tableau 3-16 : Quelques Valeurs de Registres et Couleurs Résultantes

Valeur	Couleur	Valeur	Couleur
$FFF	Blanc	$1FB	Eau claire
$D00	Rouge brique	$6FE	Bleu ciel
$F00	Rouge	$6CE	Bleu clair
$F80	Rouge orangé	$00F	Bleu
$F90	Orange	$61F	Bleu brillant
$FB0	Orange doré	$06D	Bleu foncé
$FD0	Jaune cadmium	$91F	Pourpre
$FF0	Jaune citron	$ClF	Violet
$BF0	Vert jaune	$FlF	Magenta
$8E0	Vert clair	$FAC	Rose
$0F0	Vert	$DB9	Ocre
$2C0	Vert foncé	$C80	Marron
$0B1	Vert forêt	$A87	Marron foncé
$0BB	Bleu Vert	$CCC	Gris clair
$0DB	Eau	$999	Gris moyen
		$000	Noir

SELECTION DE COULEUR EN BASSE RESOLUTION

Le Tableau 3-17 montre une sélection de couleurs d'aires de jeu en mode basse résolution. Si les combinaisons de bits en provenance des aires de jeux sont celles indiquées, le système prendra la couleur dans le numéro du registre de couleur indiqué.

Tableau 3-17 : Sélection de Couleur en Basse Résolution

Aire de Jeu Simple		**Aire de Jeu Double**	
Mode Normal (Plans de Bits 5,4,3,2,1)	Mode 'Hold-and-modify' (Plans de Bits 4,3,2,1)		Numéro du Registre de Couleur
		Aire de Jeu n° 1 Plans de Bits 5,3,1	
00000	0000	000	0 *
00001	0001	001	1
00010	0010	010	2
00011	0011	011	3
00100	0100	100	4
00101	0101	101	5
00110	0100	110	6
00111	0111	111	7
		Aire de Jeu n° 2 Plans de Bits 6,4,2	
01000	1000	000 **	8
01001	1001	001	9
01010	1010	010	10
01011	1011	011	11
01100	1100	100	12
01101	1101	101	13
01110	1110	110	14
01111	1111	111	15
10000	|	|	16
10001	|	|	17
10010	|	|	18
10011	|	|	19
10100	NON	NON	20
10101	UTILISE	UTILISE	21
10110	DANS	DANS	22
10111	CE	CE	23
11000	MODE	MODE	24
11001	|	|	25
11010	|	|	26
11011	|	|	27
11100	|	|	28
11101	|	|	29
11110	|	|	30
11111	|	|	31

* Le registre de couleur n° 0 définit toujours la couleur d'arrière-plan.
** Sélectionne le mode "transparent" au lieu de sélectionner le registre de couleur n° 8.

SELECTION DE COULEUR EN MODE 'HOLD-AND-MODIFY'

En mode 'hold-and-modify', le contenu des registres de couleur est modifié comme indiqué dans le Tableau 3-18. Ce mode n'est effectif que si le bit 10 de BPLCON0 = 1.

Tableau 3-18 : Sélection de Couleur en Mode 'Hold-and-modify'

Plan 6	**Plan 5**		**Résultat**
0	0	Opération normale	(utilise le registre de couleur)
0	1	Garde le Vert et le Rouge	B = Contenu des Plans 4-1
1	0	Garde le Vert et le Bleu	R = Contenu des Plans 4-1
1	1	Garde le Bleu et le Rouge	G = Contenu des Plans 4-1

SELECTION DE COULEUR EN MODE HAUTE RESOLUTION

Le Tableau 3-19 explique la sélection des couleurs en mode haute résolution. Si la combinaison de bits en provenance des aires de jeu est celle indiquée, le système prendra la couleur dans le numéro du registre de couleur indiqué.

Tableau 3-19 : Sélection de Couleur en Haute Résolution

Aire de Jeu Simple Plans de Bits 4,3,2,1	**Aire de Jeu Double**	**Numéro du Registre de Couleur**
	Aire de jeu n° 1 Plans de Bits 3,1	
0000	00 *	0 **
0001	01	1
0010	10	2
0011	11	3
0100	|	4
0101	INUTILISE	5
0110	DANS CE MODE	6
0111	|	7
	Aire de jeu n° 2 Plans de Bits 4,2	
1000	00 *	8
1001	01	9
1010	10	10
1011	11	11
1100	|	12
1101	INUTILISE	13
1110	DANS CE MODE	14
1111	|	15

* Sélectionne le mode "transparent".
* Le registre de couleur n° 0 définit toujours la couleur d'arrière-plan.

CHAPITRE 4

LE 'HARDWARE' DES SPRITES

INTRODUCTION

Les sprites sont des objets 'hardware' qu'on crée et déplace indépendamment de l'affichage de l'aire de jeu et indépendamment l'un de l'autre. Avec les aires de jeu, les sprites constituent l'affichage graphique de l'Amiga. On peut créer des effets d'animation plus complexes en utilisant le 'blitter', décrit dans le chapitre "Le 'Hardware' du Blitter". A l'écran, les sprites sont produits par huit canaux DMA spécialisés et propres aux sprites. Les sprites simples ont une largeur de 16 pixels et un nombre quelconque de lignes. Pour les pixels des sprites, on peut choisir parmi trois couleurs et l'un des pixels peut aussi être transparent, laissant voir ainsi tout objet pouvant se trouver au-dessous du sprite. Pour obtenir des objets plus grands et plus complexes, on a la possibilité de combiner les sprites.

Les canaux DMA des sprites peuvent être employées à plusieurs reprises à l'intérieur d'un même champ d'affichage. Ainsi, on ne sera pas limité par le fait d'avoir à l'écran seulement huit sprites à la fois.

AU SUJET DE CE CHAPITRE

Dans ce chapitre, nous traiterons des sujets suivants :

- Définition des dimensions, de la forme, de la couleur et de la position des sprites à l'écran.
- Affichage et déplacement des sprites.
- Fusion de sprites pour obtenir des images plus complexes, une plus grande largeur ou plus de couleurs.
- Réemploi d'un canal DMA des sprites à plusieurs reprises à l'intérieur d'un champ d'affichage pour pouvoir créer à l'écran plus de huit sprites à la fois.

CREATION D'UN SPRITE

Pour créer un sprite, on devra d'abord le définir, créer ensuite en mémoire une structure formelle de données. On définit un sprite en indiquant ses caractéristiques :

- Largeur maximale à l'écran de 16 pixels.
- Hauteur illimitée.
- Forme quelconque.
- Un combianison de trois couleurs, plus le transparent.
- Position quelconque à l'écran.

POSITION A L'ECRAN

On définit la position d'un sprite à l'écran par un ensemble de coordonnés X,Y. La position (0,0), où X = 0 et Y = 0, est le coin supérieur gauche de de l'affichage. On définit l'emplacement d'un sprite en indiquant les coordonnés de son pixel supérieur gauche. On définit toujours la position du sprite comme si son affichage était en basse résolution et en mode non entrelacé. Le système de coordonnés X,Y et la définition de la position d'un sprite sont représentés de manière graphique dans la Planche 4-1. Remarquez qu'à cause de l' 'overscan' de l'affichage, la position (0,0) (c'est-à-dire X = 0, Y =0), d'ordinaire, ne se trouve pas dans une région visible de l'écran.

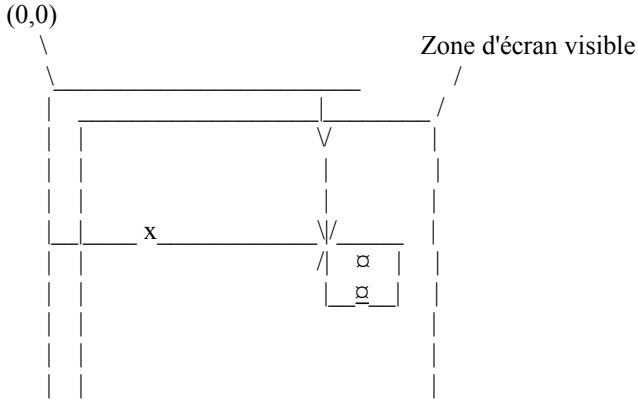


PLANCHE 4-1 : Définition de la Position du Sprite à l'Ecran.

La quantité visible de cette zone est aussi affectée par les dimensions de la fenêtre d'affichage de l'aire de jeu (définie par les valeurs présentes dans DDFSTRT, DDFSTOP, DIWSTRT, DIWSTOP, etc.). Voyez le Chapitre "Le 'Hardware' de l'Aire de Jeu" pour plus de renseignements sur l' 'overscan' et les fenêtres d'affichage.

POSITION HORIZONTALE

La position horizontale d'un sprite (la valeur X) peut se trouver à un quelconque pixel de l'écran, entre 0 et 447. Cependant, un objet, pour être visible, devra se trouver dans les limites de la fenêtre d'affichage. Dans les exemples de ce chapitre, nous utiliserons une fenêtre avec des positions horizontales qui vont du pixel 64 au pixel 383 (autrement dit, chaque ligne horizontale aura une longueur de 320 pixels). On peut définir des fenêtres plus grandes ou plus petites, selon les besoins, mais nous vous recommandons de lire le Chapitre "Le 'Hardware' de l'Aire de Jeu" avant d'entreprendre quoi que ce soit. Le faisceau vidéo balaie une partie d'écran plus importante, mais, d'habitude, cette zone n'est pas visible à l'écran.

Si, pour un sprite, on indique une valeur X qui le fait commencer en dehors de la fenêtre d'affichage, alors, une partie du sprite ou la totalité de celui-ci pourra ne pas s'afficher à l'écran. Ceci est parfois souhaitable; on dit d'un tel sprite qu'il a été "avalé".

Pour faire apparaître un sprite à l'écran à la bonne position horizontale de la fenêtre d'affichage, on additionne simplement l' 'offset' de la valeur gauche de la fenêtre à la valeur X voulue. Dans l'exemple ci-dessus, cela reviendrait à ajouter 64 à la valeur X. Par exemple, pour afficher le coin supérieur le plus à gauche d'un sprite à une position se trouvant à 94 pixels du bord gauche de l'écran, on devra faire ce calcul :

Position X voulue + 'offset' horizontal de la fenêtre d'affichage = 94 + 64 = 158.

Ainsi, 158 devient la valeur X, qu'on écrira dans la structure de données.

NOTE

La position X représente l'emplacement du tout premier (le plus à gauche) pixel d'un sprite ayant la largeur maximale de 16 bits. C'est toujours le cas, même si on indique comme transparents les pixels les plus à gauche et que ceux-ci n'apparaissent pas à l'écran.

Si le sprite que nous montrons dans la Planche 4-2 se trouvait à une position X = 158, l'image réelle à l'écran commencerait quatre pixels après, à 162. Les quatre premiers pixels du sprite sont transparents et permettent donc d'afficher l'arrière-plan.

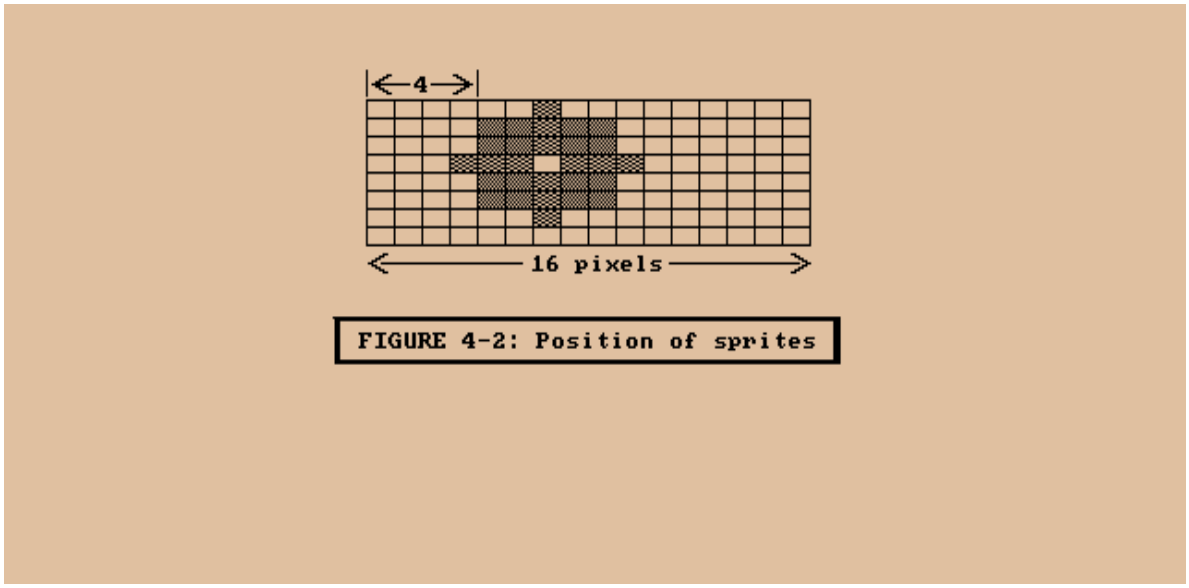


FIGURE 4-2: Position of sprites

PLANCHE 4-2 : Position de Sprites

POSITION VERTICALE

Pour le bord supérieur du sprite, on a la possibilité de sélectionner une quelconque position de la ligne 0 à la ligne 262. Dans les exemples de ce chapitre, nous utiliserons une fenêtre NTSC avec des positions verticales se situant entre la ligne 44 et la ligne 243. Si l'on indique une position verticale (la valeur Y) inférieure à 44 (par ex., au-dessus de la partie supérieure de la fenêtre d'affichage) le bord supérieur du sprite n'apparaît pas à l'écran.

Pour faire apparaître un sprite à l'écran à sa bonne position verticale, on additionne la valeur d'Y à la position voulue. En employant les chiffres ci-dessus, on ajoutera 44 à la position Y voulue. Par exemple, pour afficher le pixel supérieur le plus à gauche 25 lignes au-dessous du bord supérieur de l'écran, on devra faire ce calcul :

Position y voulue + 'offset' vertical de la fenêtre d'affichage = 25 + 44 = 69

Ainsi, 69 devient la valeur Y, qu'on écrira dans la structure de données.

SPRITES "CLIPPES"

Comme nous l'avons dit précédemment, les sprites seront partiellement ou totalement clippés s'ils passent au-dessus ou au-delà des limites de la fenêtre d'affichage. Des valeurs de 64 (horizontale) et 44 (verticale) sont "normales" pour un affichage centré sur un moniteur vidéo au standard NTSC. Voyez le Chapitre 3, "Le 'Hardware' de l'Aire de Jeu", pour plus de renseignements sur les 'offsets' concernant l'affichage. On trouvera aussi dans ce chapitre des informations sur les affichages en PAL. Si l'on choisit d'autres valeurs pour établir une fenêtre d'affichage, les sprites seront clippés en conséquence.

DIMENSIONS DES SPRITES

Les sprites ont une largeur de 16 pixels et peuvent avoir n'importe quelle hauteur... être aussi petits qu'une ligne ou plus grands que l'écran. Probablement, sur le plan vertical, vous souhaiterez avoir un très grand sprite pour pouvoir en afficher une seule partie à la fois.

La dimension d'un sprite s'appuie sur celle du pixel qui est égale à 1/320ème de la largeur de l'écran, 1/200ème de la hauteur d'un écran en mode NTSC ou 1/256ème de la hauteur en mode PAL. Cette dimension du pixel correspond à celle en basse résolution et en mode non entrelacé d'une aire de jeu normale d'une taille maximale. Les sprites, cependant, sont indépendants des modes d'affichage d'une aire de jeu, ainsi, le fait de changer la résolution ou l'entrelacement de l'aire de jeu ne produira aucun effet sur les dimensions ou sur la résolution d'un sprite.

FORME DES SPRITES

Un sprite peut avoir n'importe quelle forme, pourvu que celle-ci s'accommode d'une largeur de 16 pixels. On définit une forme de sprite en indiquant les pixels qui devront réellement s'afficher à chaque emplacement du sprite. Par exemple, les Planches 4-3 et 4-4 montrent un engin spatial dont on a souligné la forme avec des X. La première Planche montre seulement l'engin tel qu'on pourrait l'esquisser sur du papier quadrillé. La deuxième Planche montre l'engin spatial à l'intérieur d'une largeur de 16 pixels. Les 0 qui entourent l'engin soulignent la portion de sprite qui n'est pas recouverte par l'engin lui-même et qui sera transparente à l'affichage.

```
      X X
    X X X X X
X X X X X X X X X X
X X X X X X X X X X
    X X X X X X
      X X
```

PLANCHE 4-3 : Forme d'Engin Spatial

o o o o **x x** o o o o o o o o o o
o o **x x x x x x** o o o o o o o o
x x x x x x x x x x o o o o o o
x x x x x x x x x x o o o o o o
o o **x x x x x x** o o o o o o o o
o o o o **x x** o o o o o o o o o o

PLANCHE 4-4 : Sprite avec une Forme d'Engin Spatial Définie

Dans cet exemple, la portion plus large de la forme est de 10 pixels et la forme est décalée sur la gauche du sprite. Chaque fois que la forme est plus restreinte que le sprite, on peut contrôler quelle partie du sprite est utilisée pour définir cette forme. Cette forme-ci pourrait aussi commencer au pixels 2-7 au lieu de commencer au pixel 1.

COULEUR DES SPRITES

Lorsqu'on utilise les sprites individuellement (c'est-à-dire, quand ceux-ci ne sont pas "rattachés", dans le sens décrit plus loin dans la partie "Sprites Rattachés"), chaque pixel peut appartenir à l'une des trois couleurs ou être transparent. On sélectionne les couleurs exactement de la même manière qu'on sélectionne les couleurs des aires de jeu. La Planche 4-5 montre comment on détermine la couleur de chaque pixel d'un sprite.

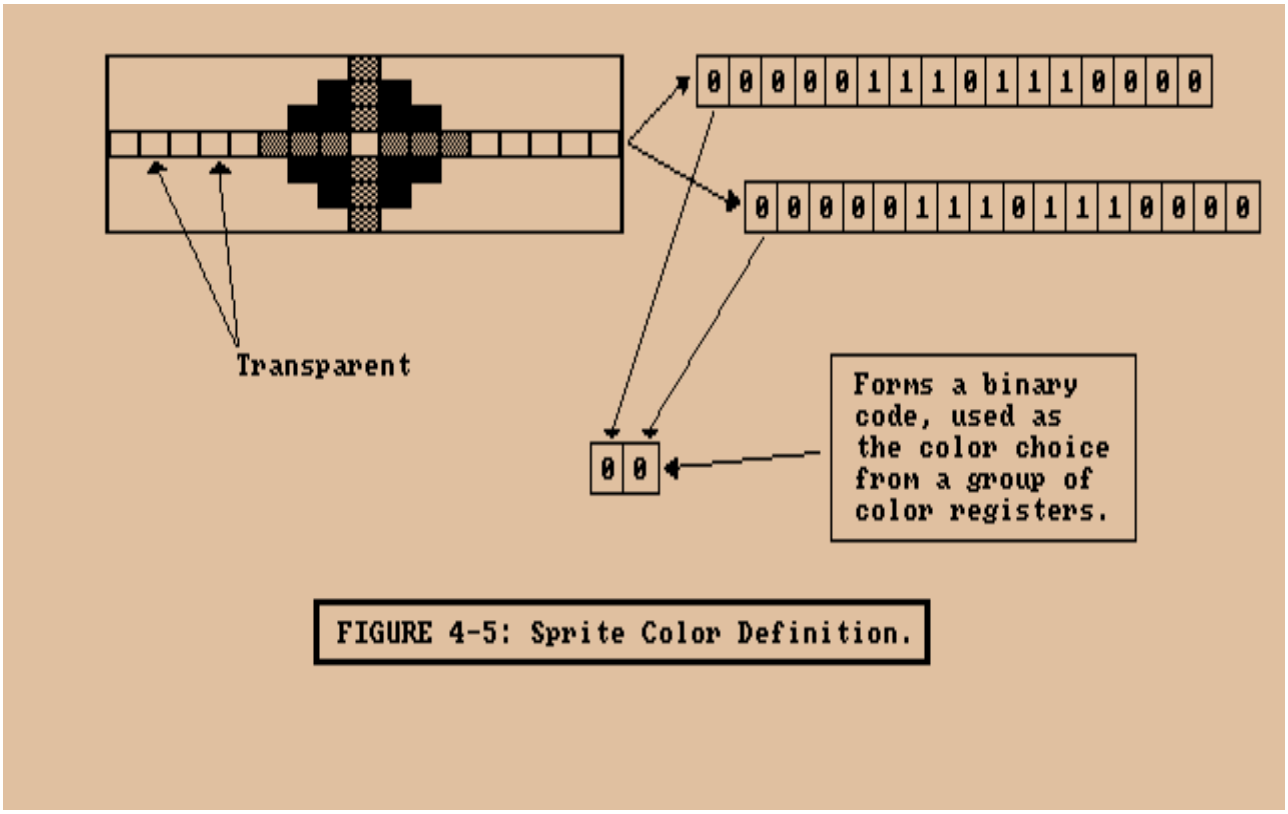


PLANCHE 4-5 : Définition de Couleur d'un Sprite

Les 0 et les 1 des deux mots de données, définissant chaque ligne d'un sprite dans la structure de données, constituent un chiffre binaire. Ce chiffre pointe sur l'un des quatre registres de couleur assignés au canal DMA de sprites. Les huit sprites emploient les registres de couleur 16 -31 du système. Dans le but de pouvoir sélectionner la couleur, les huit sprites sont organisés par paires et chaque paire emploie quatre registres de couleur comme nous le montrons dans la Planche 4-6.

NOTE

La valeur de la couleur du premier registre de chaque groupe de quatre registres est ignorée par les sprites. Quand les bits du sprite sélectionnent ce registre, ils utilisent la valeur "transparent".

Les codes 01,10 ou 11 sélectionnent l'un des trois registres possibles du groupe normal des registres de couleur, d'où sera tirée la donnée pour la couleur véritable.

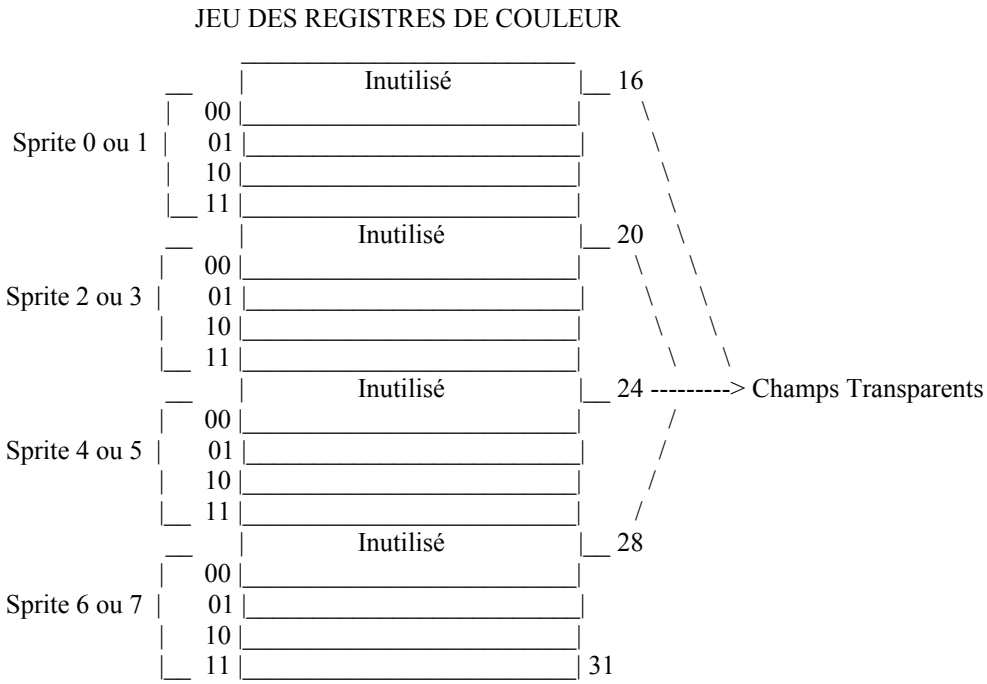


PLANCHE 4-6 : Attributions des Registres de Couleur

Si, dans un sprite, on a besoin de certaines couleurs, on devra charger les registres de couleur du sprite avec les couleurs voulues. Le Chapitre "Le 'Hardware' de l'Aire de Jeu" contient des renseignements sur le chargement des registres de couleur.

Dans ce système de couleurs, le chiffre binaire 00 est particulier. Un pixel avec une valeur 00 devient transparent et laisse voir la couleur de n'importe quel autre sprite ou aire de jeu de priorité vidéo inférieure. Un objet de priorité inférieure s'affiche "au-dessous" d'un objet de priorité supérieure. Parmi les sprites et les aires de jeu, chaque sprite possède une priorité vidéo déterminée. On peut faire varier les priorités entre sprites et aires de jeu. (Cf. Chapitre 7, "Le 'Hardware de Contrôle du Système" pour plus d'informations sur la priorité des sprites.)

CONCEPTION D'UN SPRITE

Pour ce qui est de la conception, il est commode de dessiner d'abord le sprite sur papier. On pourra indiquer les couleurs voulues sous forme de nombres de 0 à 3. Par exemple, l'engin spatial dont nous avons parlé plus haut pourrait ressembler à quelque chose comme :

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

La prochaine étape sera la conversion des chiffres 0-3 en chiffres binaires, chiffres qui seront employés pour construire les mots du descripteur de couleurs de la structure de données du sprite. La section ci-dessous montre comment il faudra procéder.

CONSTRUCTIUON DE LA STRUCTURE DE DONNEES

Après avoir défini le sprite, on devra construire sa structure de données, qui est une suite de mots de 16 bits dans une zone de mémoire contigüe. Certains de ces mots contiennent des renseignements sur la position et le contrôle et certains autres contiennent la description des couleurs. Pour créer la structure de données d'un sprite, il vous faudra :

- Ecrire la position horizontale et verticale du sprite dans le premier mot de contrôle.

- Ecrire la position verticale de fin dans le deuxième mot de contrôle.

- Traduire les chiffres décimaux 0 - 3 indiquant la couleur, chiffres présents dans l'image quadrillée du sprite, en chiffres binaires. Utilisez les valeurs binaires pour bâtir les mots du descripteur (données) de couleurs et écrivez ces mots dans la structure de données.

- Ecrire les mots de contrôle qui indiquent la fin de la structure de données du sprite.

NOTE

Les données de sprite, comme toutes les autres données auxquelles ont accès les puces spécialisées, devront être chargées en mémoire 'Chip'. Assurez-vous que les structures de données du sprite en mémoire 'Chip' ont une adresse divisible par deux.

Le Tableau 4-1 montre une structure de données de sprite avec l'implantation en mémoire et la fonction de chaque mot :

TABLEAU 4-1 : Structure de Données de Sprite

Implantation en Mémoire	**Mots de 16 bits**	**Fonction**
N	Mot n° 1 contrôle sprite	Position vertical et horizontale de début
N+1	Mot n° 2 contrôle sprite	Position verticale de fin
N+2	Mot de poids faible du	Bits de couleur pour la ligne 1 descripteur de couleur
N+3	Mot de poids fort du	Bits de couleur pour la ligne 1 descripteur de couleur
N+4	Mot de poids faible du	Bits de couleur pour la ligne 2 descripteur de couleur
N+5	Mot de poids fort du	Bits de couleur pour la ligne 2 descripteur de couleur
	.	
	.	
	.	
	Mots de fin de données	Deux mots indicant l'utilisation à venir du sprite

Toutes les adresses mémoire des sprites sont des adresses en mots. Il faudra avoir une quantité de mémoire contigüe suffisante pour trouver la place pour les deux mots d'information de contrôle, pour les deux mots de chaque ligne horizontale du sprite et pour les deux mots de fin des données.

Comme les puces spécialisées doivent pouvoir avoir accès à cette structure de données, il faudra s'assurer que les données se trouvent bien en mémoire 'Chip'.

La Planche 4-7 montre comment la structure de données est rattachée au sprite.

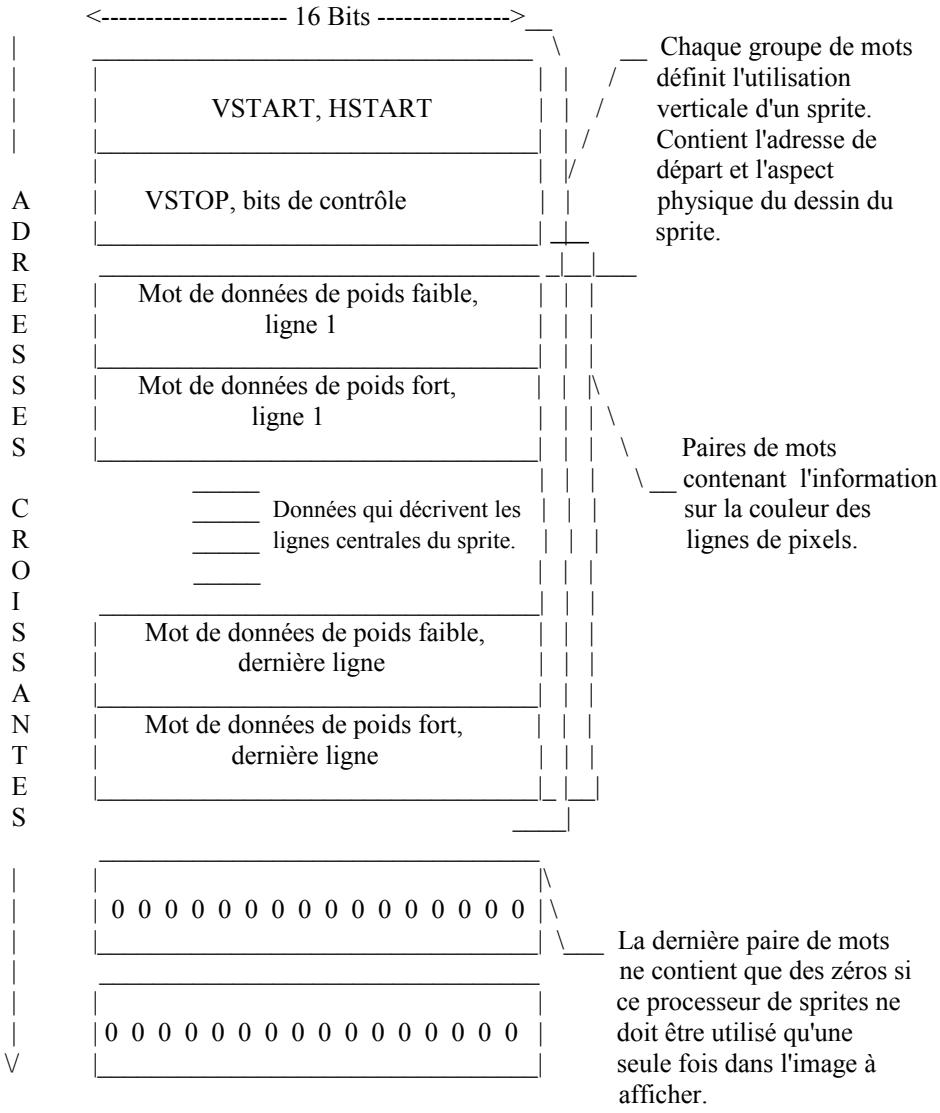


PLANCHE 4-7. PREMIERE PARTIE : Organisation de la Structure de Données

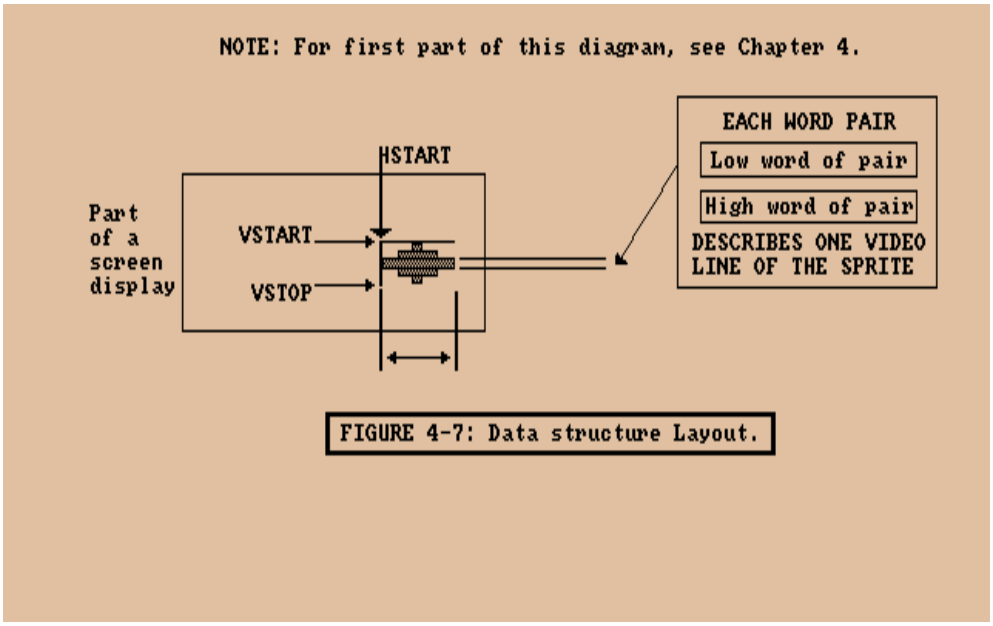


PLANCHE 4-7. DEUXIEME PARTIE : Organisation de la Structure de Données

MOT n° 1 CONTROLE SPRITE : SPRxPOS

Ce mot contient la position de début verticale (VSTART) et horizontale (HSTART) du sprite. Il indique l'emplacement de la ligne supérieure la plus éloignée du sprite.

Bits 15-8 contiennent les 8 bits de poids faible de VSTART
Bits 7-0 contiennent les 8 bits de poids fort de HSTART

MOT n° 2 CONTROLE SPRITE : SPRxCTL

Ce mot contient la position de fin verticale du sprite à l'écran (par ex., la ligne APRES la dernière rangée du sprite affichée). Il contient aussi des données en rapport avec le rattachement du sprite, dont nous parlerons par la suite.

SPRxCTL

Bits 15-8	Les huit bits de poids faible de VSTOP
Bit 7	(Utilisé pour le rattachement)
Bits 6-3	Inutilisé (mettre à zéro)
Bit 2	Le bit de poids fort de VSTART
Bit 1	Le bit de poids fort de VSTOP
Bit 0	Le bit de poids faible de HSTART

Les valeurs (VSTOP - VSTART) définissent la hauteur, en lignes de balayage, du sprite quand celui-ci s'affiche.

MOTS DU DESCRIPTEUR DE COULEUR DU SPRITE

Il faut deux mots du descripteur de couleur pour décrire chaque ligne horizontale d'un sprite; le mot le plus significatif et le mot le moins significatif. Pour calculer de combien de mots vous aurez besoin pour le descripteur de couleur, multipliez la hauteur du sprite en lignes par 2. Les bits les plus significatifs du mot du descripteur de couleur contribuent à constituer le chiffre de poids fort le plus à gauche du nombre binaire du sélecteur de couleur de chaque pixel; le mot moins significatif contribue à constituer le chiffre le plus à droite.

Pour composer les mots du descripteur de couleur, il faudra d'abord composer une image du sprite, qui montre la couleur de chaque pixel représentée par un nombre de 0 à 3. Chaque nombre représente l'une des trois couleurs des registres de couleur du sprite. Par exemple, voici encore une fois le sprite d'engin spatial :

```
0000122332210000
0001223333221000
0012223333222100
0001223333221000
0000122332210000
```

Ensuite, on traduit chaque nombre de cette image en nombre binaire. Nous montrons, ci-dessous, la première ligne en binaire. Les nombres binaires seront représentés sous forme verticale avec le chiffre de poids faible dans la ligne supérieure et celui de poids fort dans la ligne au-dessous. Voici comment les deux mots du descripteur de couleur seront écrits en mémoire.

```
0000100110010000  <---  Mot De Poids Faible Du Sprite
0000011111100000  <---  Mot De Poids Fort Du Sprite
```

La première ligne ci-dessus deviendra le mot de poids faible du descripteur de couleur de la ligne n° 1 du sprite. la seconde ligne deviendra le mot de poids fort du descripteur de couleur. De cette façon, on traduit chaque ligne du sprite en chiffres binaires 0 et 1. Cf. Planche 4-7.

Chaque nombre binaire composé d'une combinaison de deux mots de données pour chaque ligne se réfère à un registre de couleur spécifique au segment de la table de couleurs du canal de ce sprite-là. Le sprite du canal 0, par exemple, prend ses couleurs dans les registres 17 - 19. Dans le Tableau 4-2, nous indiquons les nombres binaires correspondant aux registres de couleurs pour le canal DMA du sprite.

Tableau 4-2 : Registres de Couleur du Sprite

NOMBRE BINAIRE	NUMERO DU REGISTRE DE COULEUR
00	Transparent
01	17
10	18
11	19

Souvenez-vous que 00 binaire signifie toujours transparent et ne se réfère jamais à une couleur, sauf pour le fond.

MOTS DE FIN DE DONNEES

Quand la position verticale du compteur de faisceau vidéo est égale à la valeur VSTOP dans les mots de contrôle du sprite, les deux prochains mots prélevés dans la structure de données du sprite sont écrits dans les registres de contrôle du sprite au lieu d'être envoyés aux registres de couleur. Ces deux mots sont interprétés par le

'hardware' de la même façon que les mots originaux chargés en premiers dans les registres de contrôle. Si la valeur VSTART, contenue dans ces mots, est inférieure à la position courante du faisceau vidéo, le sprite ne sera pas utilisé à nouveau dans le champ d'affichage. Par cohérence, on devrait employer la valeur 0 pour les deux mots, quand on aura fini d'utiliser un sprite. Nous parlerons plus tard du réemploi des sprites.

La structure de données qui suit est celle du sprite de l'engin spatial. Elle sera située à V = 65 et H = 128 dans la partie normalement visible de l'écran.

```
SPRITE:
        DC.W    $6D60, $7200    ; VSTART, HSTART, VSTOP
        DC.W    $0990, $07E0    ; Première paire de mots du descripteur
        DC.W    $13C8, $0FF0
        DC.W    $23C4, $1FF8
        DC.W    $13C8, $0FF0
        DC.W    $0990, $07E0
        DC.W    $0000, $0000    ; Fin des données du sprite.
```

AFFICHAGE D'UN SPRITE

Après avoir construit la structure de données, il faut dire au système de l'afficher. Cette section traite de l'affichage des sprites en mode "automatique". Dans ce mode, une fois que le canal DMA de sprite commence à saisir et à afficher les données, l'affichage continue jusqu'à ce que la position verticale VSTOP ne soit atteinte. Nous décrirons le mode manuel plus loin dans ce chapitre.

Pour l'affichage d'un sprite, on suit les étapes suivantes :

1. Décider lequel des huit canaux DMA de sprites il faut utiliser (en s'assurant que le canal choisi est libre).

2. Mettre en place les pointeurs du sprite pour dire au système où trouver les données du sprite.

3. Activer l'accès direct à la mémoire si celui-ci n'est pas encore activé.

4. A chaque champ d'affichage successif, pendant l'intervalle de suppression verticale, réécrire les pointeurs du sprite.

ATTENTION

Si le canal DMA est activé pendant que le système est en train d'afficher un sprite (ce qui se produit après VSTART, mais avant VSTOP), le système continuera à afficher la ligne de données du sprite qui vient juste d'être prélevée. Ce qui engendre l'affichage d'une barre verticale à l'écran. Il est recommandé d'invalider le canal DMA du sprite seulement pendant la suppression verticale ou sur une partie de l'affichage où l'on est sûr qu'il n'y aura pas de sprites à afficher.

SELECTION D'UN CANAL DMA ET MISE EN PLACE DES POINTEURS

Quand on décide du canal DMA à employer, on devrait prendre en considération les couleurs affectées au sprite et la priorité vidéo de celui-ci.

Le canal DMA des sprites utilise deux pointeurs pour lire les données du sprite et les mots de contrôle. Pendant l'intervalle de suppression verticale, avant le premier affichage du sprite, il faut écrire dans ces pointeurs les adresses mémoire du sprite. Les pointeurs de chaque sprite sont SPRxPTH et SPRxPTL, où "x" est le numéro du canal DMA du sprite. SPRxPTH contient les trois bits de poids fort de l'adresse mémoire du premier mot du sprite, tandis que SPRxPTL contient les seize bits de poids faible. Le bit le moins significatif de SPRxPTL est ignoré, étant donné que les données de sprite doivent être divisibles par deux. Ainsi, on n'utilise que quinze bits de SPRxPTL. Comme d'habitude, on peut écrire un mot long dans SPRxPTH.

Dans l'exemple ci-dessous, le processeur initialise les pointeurs de données du sprite n° 0. Normalement, ce travail est effectué par le Copper. Le sprite se trouve à l'adresse $20000.

```
MOVE.L #$20000,SPR0PTH+CUSTOM    ; Ecrire S20000 dans le pointeur de sprite du sprite n° 0...
```

Ces pointeurs sont dynamiques; ils sont augmentés d'une unité par le canal DMA du sprite pour pointer d'abord sur les mots de contrôle, ensuite sur les mots de données et, enfin, sur les mots de fin de données. Après avoir lu l'information concernant le contrôle du sprite et après l'avoir stockée dans d'autres registres, ils continuent à lire les mots du descripteur de couleur. Ces mots sont stockés dans les registres de données du sprite, qui sont utilisés par le canal DMA des sprites pour afficher les données à l'écran. Pour plus de renseignements sur la manière dont les canaux DMA des sprites gèrent l'affichage, voyez ci-dessous la section "Particularités du 'Hardware' des Sprites".

REINITIALISATION DES POINTEURS D'ADRESSE

Pour un seul champ d'affichage, le système lira la structure de données automatiquement et produira à l'écran le sprite avec les couleurs indiquées dans les registres de couleur du sprite. Si on souhaite que le sprite soit affiché dans les plans d'affichage successifs, on devra réécrire le contenu des pointeurs du sprite pendant l'intervalle de suppression verticale. Ceci est nécessaire parce que dans le champ d'affichage, les pointeurs sont augmentés d'une unité pour pointer sur les données qui devront être prélevées à mesure que l'affichage d'écran progressera.

Cette réécriture devient une partie de la routine de suppression verticale, qu'on peut gérer avec les instructions de la liste du Copper.

EXEMPLE D'AFFICHAGE D'UN SPRITE

Cet exemple affiche le sprite engin spatial à l'emplacement V = 65, H = 128.
N'oubliez pas d'inclure le fichier "hw_examples.i", qu'on trouve dans l'Appendice J.

```
; En premier, nous mettons en place une seule aire de jeu.
;
  LEA CUSTOM,a0                 ; Fait pointer a0 sur les puces spécialisées
  MOVE.W #$1200,BPLCON0(a0)     ; Validation d'une couleur pour le plan de bits
  MOVE.W #$0000,BPL1MOD(a0)     ; Modulo = 0
  MOVE.W #$0000,BPLCON1(a0)     ; Valeur de déroulement horizontal = 0
  MOVE.W #$0024,BPLCON2(a0)     ; Les sprites sont prioritaires sur les aires de jeu
  MOVE.W #$0038,DDFSTRT(a0)     ; Fixer le début d'acquisition des données
  MOVE.W #$00D0,DDFSTOP(a0)     ; Fixer la fin d'acquisition des données

; Définitions pour la fenêtre d'affichage.

  MOVE.W #$2C81,DIWSTRT(a0)     ; Mettre en place la position du point d'origine de la fenêtre d'affichage.
                                ; Point d'origine vertical dans l'octet de poids fort.
                                ; Point d'origine horizontal * 2 dans l'octet de poids faible.
  MOVE.W #$F4C1,DIWSTOP(a0)     ; Mettre en place le point d'aboutissement vertical de la fenêtre
                                ; d'affichage dans l'octet de poids fort.
                                ; Point d'aboutissement horizontal * 2 dans l'octet de poids faible.

; Mettre en place les registres de couleur.
;
  MOVE.W #$0008,COLOR00(a0)     ; Couleur d'arrière-plan = bleu foncé
  MOVE.W #$0000,COLOR01(a0)     ; Couleur de premier plan = noir
  MOVE.W #$0FF0,COLOR17(a0)     ; Couleur 17 = jaune
  MOVE.W #$00FF,COLOR18(a0)     ; Couleur 18 = cyan
  MOVE.W #$0FOF,COLORl9(a0)     ; Couleur 19 = magenta
;
; Mettre la liste Copper à l'adresse $20000.
;
  MOVE.L #$20000,a1             ; Faire pointer A1 sur la destination de liste Copper
  LEA    COPPERL(pc),a2         ; Faire pointer A2 sur l'origine de la liste Copper

CLOOP:
  MOVE.L (a2),(a1)+             ; Mettre un mot long
  CMP.L  #$FFFFFFFE,(a2)+       ; Tester la fin de la liste
  BNE    CLOOP                  ; Boucler jusqu'à ce que toute la liste
                                ; soit transférée
;
; Mettre le sprite à l'adresse $25000.
;
  MOVE.L #$25000,a1             ; Faire pointer A1 sur la destination du sprite
  LEA    SPRITE(pc),a2          ; Faire pointer A2 sur l'origine du sprite

SPRLOOP:
  MOVE.L (a2),(a1)+             ; Mettre un mot long
  CMP.L  #$00000000,(a2)+       ; Tester la fin du sprite
  BNE    SPRLOOP                ; Boucler jusqu'à ce que tout le sprite soit transféré
;
; Maintenant, nous écrivons un sprite bidon à l'adresse $30000, puisque tous les huit canaux sont activés en
; même temps et que nous n'en utiliserons qu'un. Les sprites restants pointeront sur les données de ce sprite bidon.
;
  MOVE.L #$00000000,$30000      ; L'écrire
;
```

```
; Faire pointer le Copper sur la liste Copper.
;
  MOVE.L #$20000,COP1LC(a0)
;
; Remplir le plan de bits avec $FFFFFFFF.
;
  MOVE.L #$21000,a1              ; Faire pointer A1 sur le plan de bits
  MOVE.W #l999,d0                ; 2000-1 (pour 'dbf') mots longs = 8000 octets
FLOOP
  MOVE.L #$FFFFFFFF,(al)+        ; Mettre un mot long $FFFFFFFF
  DBF    d0,FLOOP                ; Décrémenter, répéter jusqu'à ce que la condition soit fausse.
;
; Mettre en route le DMA.
;
  MOVE.W d0,COPJMP1(a0)          ; Forcer le chargement dans le Copper du compteur de programme
  MOVE.W #$83A0,DMACON(a0)       ; Plan de bits, Copper et DMA de sprite
  RTS                            ; ..retour au reste du programme..
;
; Voici une liste Copper pour un plan de bits et 8 sprites.
; Le plan de bits se trouve à l'adresse $21000.
; Le sprite n° 0 se trouve à l'adresse $25000; tous les autres se trouvent
; à l'adresse $30000 (le sprite bidon).
;
COPPERL:
  DC.W   BPL1PTH,$0002           ; Pointeur du plan de bits n° 1 = $21000
  DC.W   BPL1PTL,$1000
  DC.W   SPR0PTH,$0002           ; Pointeur de sprite n° 0 = $25000
  DC.W   SPR0PTL,$5000
  DC.W   SPR1PTH,$0003           ; Pointeur de sprite n° 1 = $30000
  DC.W   SPR1PTL,$0000
  DC.W   SPR2PTH,$0003           ; Pointeur de sprite n° 2 = $30000
  DC.W   SPR2PTL,$0000
  DC.W   SPR3PTH,$0003           ; Pointeur de sprite n° 3 = $30000
  DC.W   SPR3PTL,$0000
  DC.W   SPR4PTH,$0003           ; Pointeur de sprite n° 4 = $30000
  DC.W   SPR4PTL,$0000
  DC.W   SPR5PTH,$0003           ; Pointeur de sprite n° 5 = $30000
  DC.W   SPR5PTL,$0000
  DC.W   SPR6PTH,$0003           ; Pointeur de sprite n° 6 = S30000
  DC.W   SPR6PTL,$0000
  DC.W   SPR7PTH,$0003           ; Pointeur de sprite n° 7 = $30000
  DC.W   SPR7PTL,$0000
  DC.W   $FFFF,$FFFE             ; Fin de la liste Copper
;
; Données de sprite pour le sprite engin spatial. Celui-ci s'affiche à l'écran
; à V-65 et à H-128.
;
SPRITE:
  DC.W   $6D60,$7200             ; VSTART, HSTART, VSTOP
  DC.W   $0990,$07E0             ; Première paire de mots du descripteur
  DC.W   $13C8,$0FF0
  DC.W   $23C4,$1FF8
  DC.W   $13C8,$0FF0
  DC.W   $0990,$07E0
  DC.W   $0000,$0000             ; Fin de données du sprite
```

DEPLACEMENT D'UN SPRITE

Un sprite généré en mode automatique peut être déplacé si on indique une position différente dans la structure de données. A chaque champ d'affichage, les données sont relues et le sprite est redessiné. Cependant, si on change les données de position avant que le sprite ne soit redessiné, celui-ci s'affichera dans une nouvelle position et aura l'air de bouger.

On doit faire attention à ne pas déplacer le sprite (ce qui revient à changer les données des mots de contrôle) pendant que le système utilise ses données pour trouver l'endroit où afficher l'objet. Si l'on faisait cela, le système pourrait trouver une position de départ dans un champ et une position d'arrivée dans le champ suivant à mesure qu'il prélève les données destinées à l'affichage. Ceci provoquerait un "incident" et sèmerait la pagaïe dans l'écran. Par conséquent, on ne devra changer le contenu des mots de contrôle qu'au moment où le système n'essaie pas de les lire. D'ordinaire, la période de suppression verticale est un moment digne de confiance, par conséquent le déplacement des sprites devient une partie des tâches de la suppression verticale et est géré par le Copper, comme nous le montrons dans l'exemple ci-dessous.

Comme les sprites se déplacent à l'écran, ils peuvent entrer en collision l'un avec l'autre ou avec l'une ou l'autre de deux aires de jeu. On peut se servir du 'hardware' pour détecter ces collisions et exploiter cette possibilité pour créer des effets particuliers. De plus, on peut utiliser la détection de collision pour garder un objet qui se déplace à l'intérieur de certaines limites bien déterminées de l'écran. Nous décrivons la détection de collision au Chapitre 7, "Le 'Hardware' de Contrôle du Système".

Dans cet exemple de déplacement d'un sprite, l'engin spatial rebondit à l'écran, en changeant de direction chaque fois qu'il atteint un bord.

Les données de position du sprite, contenant VSTART et HSTART, se trouvent en mémoire à l'adresse $25000. VSTOP se trouve à l'adresse $25002. Pour déplacer le sprite, on écrit dans ces emplacements. Une fois par trame, VSTART est augmenté d'une unité (ou décrémenté) et HSTART de 2 unités. Puis, le système calcule à nouveau le nouveau VSTOP qui sera le nouveau VSTART+6.

```
   MOVE.B #151,d0          ; Initialiser le compte horizontal
   MOVE.B #194,d1          ; Initialiser le compte vertical
   MOVE.B #64,d2           ; Initialiser la position horizontale
   MOVE.B #44,d3           ; Initialiser la position verticale
   MOVE.B #1,d4            ; Initialiser la valeur d'augmentation horizontale
   MOVE.B #1,d5            ; Initialiser la valeur de diminution verticale
;
; Ici, nous attendons le début de redéfinition de l'écran.
; Ceci garantit un affichage sans incidents.
;
   LEA CUSTOM,a0           ; Mettre en place le pointeur des puces spécialisées
VLOOP:
   MOVE.B VHPOSR(a0),d6    ; Lire la position verticale du faisceau

; Insérez la ligne suivante seulement si vous vous servez d'une machine PAL.
;  CMP.B  #$20,d6           ; Comparer avec la fin de l'écran PAL.
   BNE.S  VLOOP            ; Boucler si ce n'est pas la fin de l'écran.
```

```
; Alternativement, on peut utiliser le code suivant :
; VLOOP:
;  MOVE.W INTREQR(a0),d6          ; Lire le mot de demande d'interruption
;  AND.W  #$0020,d6               ; Masquer tout, sauf le bit de suppression verticale
;  BEQ    VLOOP                   ; Boucler jusqu'à ce que ce bit soit à 1
;  MOVE.W #$0020,INTREQ(a0)       ; Le bit vertical est activé, donc il faut le réinitialiser

; Notez, svp, que ceci marchera seulement si vous avez mis hors service la
; validation d'interruption de suppression (ce qui n'est pas recommandée
; pour des longues périodes).

   ADD.B  d4,d2                   ; Augmenter d'une unité la valeur horizontale
   SUBQ.B #1,d0                   ; Diminuer d'une unité le compteur horizontal
   BNE    L1
   MOVE.B #151,d0                 ; Compte épuisé, réinitialiser à 151
   EOR.B  #$FE,d4                 ; Inverser la valeur d'augmentation
L1:
   MOVE.B d2,$25001               ; Ecrire la nouvelle valeur HSTART dans le sprite
   ADD.B  d5,d3                   ; Augmenter d'une unité la valeur verticale
   SBQ.B  #1,d1                   ; Diminuer d'une unité le compteur vertical
   BNE    L2
   MOVE.B #194,d1                 ; Compte épuisé, réinitialiser à 194
   EOR.B  #$FE,d5                 ; Inverser la valeur d'augmentation

L2:
   MOVE.B d3,$25000               ; Ecrire la nouvelle valeur VSTART dans le sprite
   MOVE.B d3,d6                   ; Maintenant on doit calculer le nouveau VSTOP
   ADD.B  #6,d6                   ; VSTOP toujours égal VSTART+6 pour l'engin spatial
   MOVE.B d6,$25002               ; Ecrire le nouveau VSTOP dans le sprite
   BRA    VLOOP                   ; Boucler à l'infini
```

CREATION DE SPRITES SUPPLEMENTAIRES

Pour employer des sprites supplémentaires, on devra créer une structure de données pour chacun d'eux et aménager l'affichage comme nous l'avons montré dans la section précédente, en nommant les pointeurs SPR1PTH et SPR1PTL pour le canal DMA de sprites n° 1, SPR2PTH et SPR2PTL pour le canal DMA de sprites n° 2, et ainsi de suite.

NOTE

Quand on valide le canal DMA d'un sprite, on valide les canaux DMA de tous les sprites et on les met tous en mode automatique. Ainsi, il n'est pas nécessaire de répéter cette opération lorsqu'on utilise des canaux DMA supplémentaires pour des sprites.

Une fois que les canaux DMA ont été validés, tous les pointeurs des huit sprites devront avoir été initialisés soit sur un sprite réel soit sur un sprite nul de sauvegarde. Un sprite non initialisé pourrait provoquer l'apparition à l'écran de sprites vidéo parasites.

Souvenez-vous que certains sprites peuvent devenir inutilisables lors de l'affichage à l'écran quand on alloue des cycles DMA supplémentaires, par exemple, quand on valide un affichage extrêmement large ou un déroulement excessivement ample. (cf. Planche 6-9 : Allocation d'une Plage de Temps pour les DMA).

En outre, rappelez-vous que chaque paire de sprites prend ses couleurs dans des registres de couleur différents, comme nous le montrons dans le Tableau 4-3.

Tableau 4-3 : Paires de Registres de Couleur du Sprite

NUMERO DES SPRITES	REGISTRES DE COULEUR
0 et 1	17-19
2 et 3	21-23
4 et 5	25-27
6 et 7	29-31

NOTE

Certains sprites deviennent inutilisables lors de l'affichage à l'écran quand on alloue des cycles DMA supplémentaires, par ex., quand on valide un affichage extrêmement large ou un déroulement horizontal excessivement ample. (cf. Planche 6-9 : Allocation d'une Plage de Temps pour les DMA).

PRIORITE DES SPRITES

Quand on a plus d'un sprite à l'écran, on pourrait avoir besoin de prendre en considération la priorité vidéo relative des sprites, c'est-à-dire quel sprite s'affiche devant ou derrière un autre. Chaque sprite a une priorité vidéo déterminée qui respecte celle de tous les autres. Le sprite ayant le numéro le plus bas a la priorité plus élevée et s'affiche devant tous les autres; le sprite ayant le numéro le plus élevé a la priorité plus basse. Ceci est illustré par la Planche 4-8.

NOTE

Pour plus de renseignements sur la priorité des Sprites, voir le Chapitre 7, "Le 'Hardware' de Contrôle du Système".

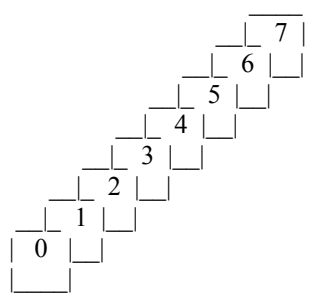


PLANCHE 4-8 : Priorité des Sprites

REEMPLOI DES CANAUX DMA DES SPRITES

Chacun des huit canaux DMA des sprites peuvent générer plus d'une image contrôlable de façon indépendante. Il pourrait y avoir des moments où l'on voudrait plus de huit objets ou alors on pourrait rester avec moins de huit objets car on aurait "rattaché" quelques sprites pour pouvoir produire plus de couleurs ou des objets plus grands ou on en aurait fait chevaucher certains pour créer des images plus complexes. Il est possible de réemployer chaque canal DMA de sprites plusieurs fois dans le même champ d'affichage, comme nous l'indiquons dans la Planche 4-9.

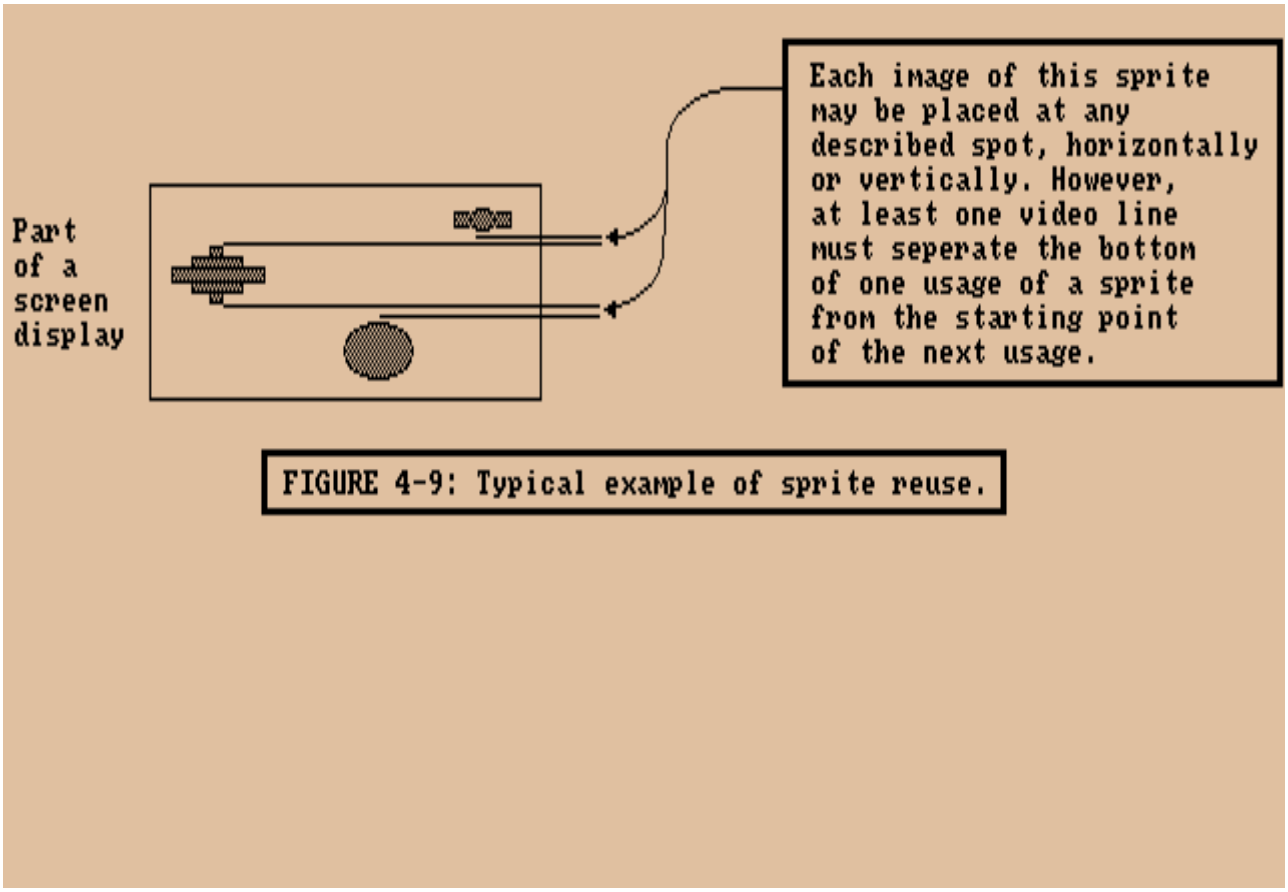


PLANCHE 4-9 : Exemple Typique de Réemploi d'un Sprite

Quand on emploie un seul sprite, deux mots, constitués de zéros, sont placés à la fin de la structure de données pour stopper le canal DMA pendant le prélèvement de données d'un sprite bien précis dans un champ d'affichage précis. Pour réemployer un canal DMA, on remplace cette paire de zéros par une autre structure de données de sprite complète, décrivant le réemploi du canal DMA à une position d'écran inférieure par rapport à celle utilisée lors du premier emploi. On met les deux mots constitués de zéros à la fin de la structure de données qui renferme toutes les informations destinées à l'utilisation du canal DMA. Par exemple, la Planche 4-10 montre la structure de données qui décrit l'image ci-dessus.

LISTE D'AFFICHAGE DES SPRITES

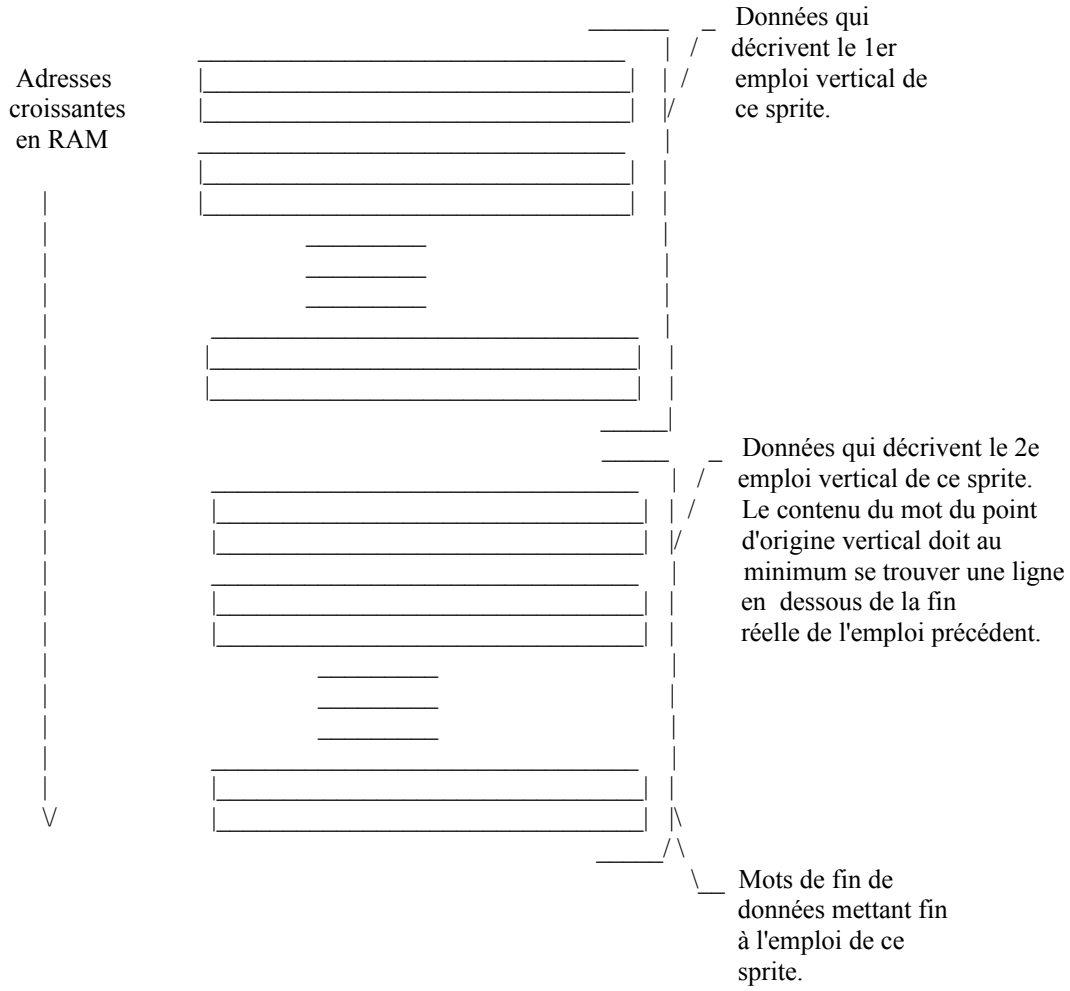


PLANCHE 4-10 : Structure de Données Typique pour le Réemploi d'un Sprite

Les seules limitations concernant le réemploi des sprites dans un seul champ d'affichage est que la dernière ligne du premier emploi du sprite doit être séparée de la première ligne, lors du prochain emploi, au minimum par une ligne horizontale de balayage. Cette limitation est indispensable car chacun des huit canaux DMA n'a droit qu'à deux cycles DMA par ligne de balayage horizontale. Le canal de sprites a besoin de temps, pendant la ligne non utilisée, pour prélever le mot de contrôle décrivant le prochain emploi du sprite.

L'exemple qui suit affiche le sprite d'engin spatial et puis l'affiche à nouveau en tant qu'objet différent.

Seule la liste de données du sprite est affectée, c'est la raison pour laquelle nous ne donnons ici que cette seule liste de données. Cependant, le sprite aura une plus belle allure, si on met en place les registres de couleur comme indiqué dans l'exemple.

```
   LEA CUSTOM,a0
   MOVE.W #$0F00,COLOR17(a0)      ; Couleur 17 rouge
   MOVE.W #$0FF0,COLOR18(a0)      ; Couleur 18 jaune
   MOVE.W #$0FFF,COLORl9(a0)      ; Couleur 19 blanc
SPRITE:
   DC.W   $6D60,$7200
   DC.W   $0990,$07E0
   DC.W   $13C8,$0FF0
   DC.W   $23C4,$1FF8
   DC.W   $13C8,$0FF0
   DC.W   $0990,$07E0
   DC.W   $8080,$8D00             ; VSTART, HSTART, VSTOP du nouveau sprite
   DC.W   $1818,$0000
   DC.W   $7E7E,$0000
   DC.W   $7FFE,$0000
   DC.W   $FFFF,$2000
   DC.W   $FFFF,$2000
   DC.W   $FFFF,$3000
   DC.W   $FFFF,$3000
   DC.W   $7FFE,$1800
   DC.W   $7FFE,$0C00
   DC.W   $3FFC,$0000
   DC.W   $0FF0,$0000
   DC.W   $03C0,$0000
   DC.W   $0180,$ÿ0000
   DC.W   $0000,$0000             ; Fin des données du sprite
```

SPRITES RECOUVERTS

Pour déplacer des objets plus larges ou plus complexes, on a la possibilité de faire chevaucher les sprites. Les faire chevaucher signifie simplement que les sprites auront les mêmes positions ou des positions d'écran relativement proches. On peut avoir un position d'écran relativement proche dans un objet qui a une largeur supérieure à 16 pixels.

La priorité vidéo intrinsèque au sprite garantit que celui-ci s'affichera au-dessous de l'autre quand les sprites se chevaucheront. Les circuits qui s'occupent des priorités donnent au sprite ayant le numéro moins élevé une priorité plus élevée et au sprite ayant le numéro le plus élevé une priorité moins élevée. Cependant, lorsqu'on envisage des affichages avec des sprites qui se chevauchent, il faut s'assurer que le sprite de "premier plan" possède un numéro d'ordre inférieur à celui du sprite d' "arrière-plan". Dans la Planche 4-11, par exemple, on devrait faire générer la cage par un canal DMA de sprites avec un numéro d'ordre moins élevé que celui du canal qui génère le singe.

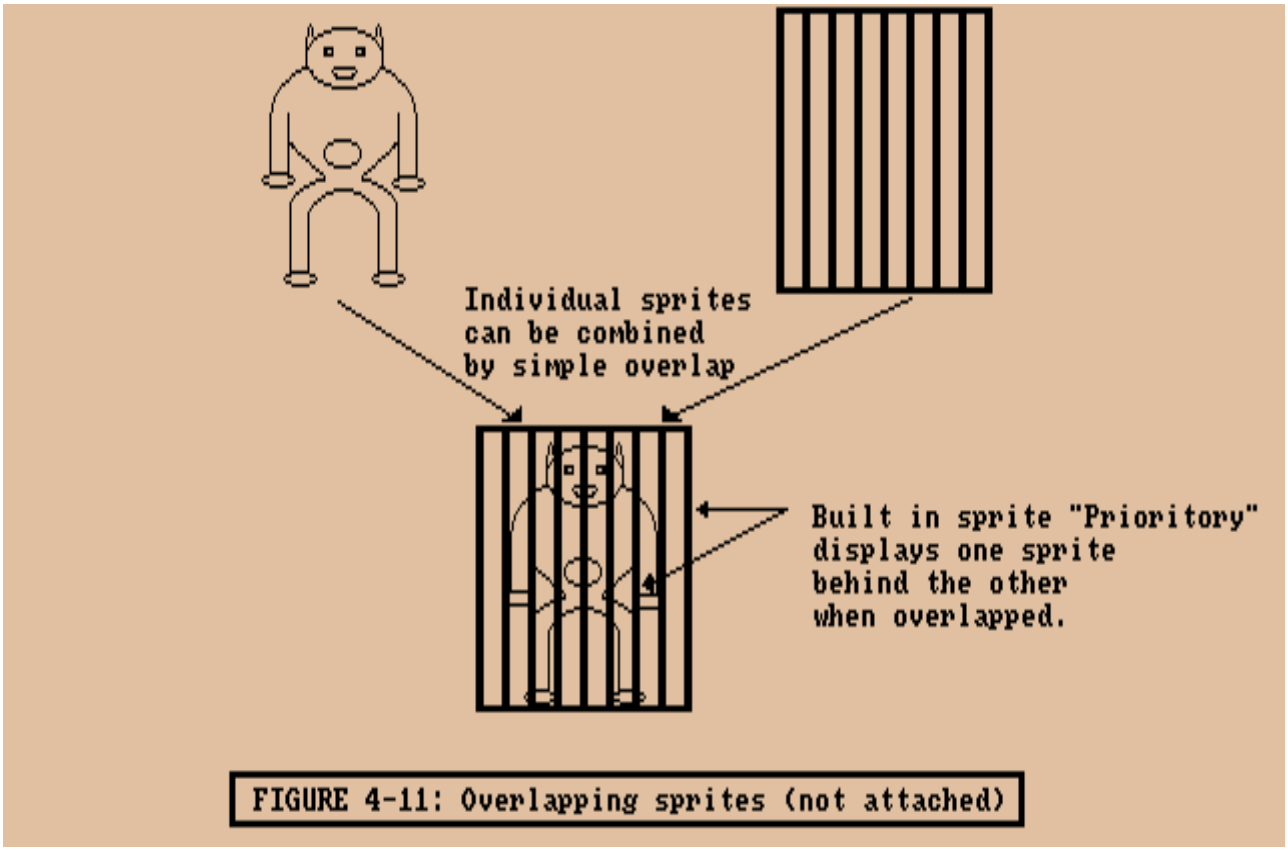


PLANCHE 4-11 : Chevauchement de Sprites (non Rattachés)

On peut créer un affichage de sprite plus grand en mettant côte-à-côte deux sprites. Par exemple, la Planche 4-12 montre le sprite d'engin spatial et la manière d'en doubler la largeur par l'utilisation de deux sprites mis côte-à-côte.

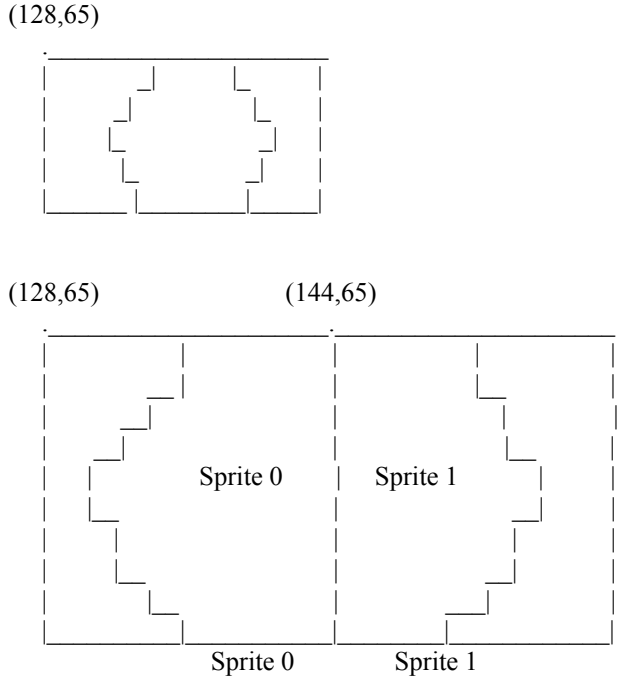


FIGURE 4-12: Mise de deux Sprites Côte-à-Côte

SPRITES RATTACHES

Il est possible de créer des sprites possédant un choix possible de quinze couleurs (plus le transparent) au lieu de trois (plus le transparent), en "rattachant" deux sprites. Pour créer des sprites rattachés, on devra :

- o Utiliser deux canaux par sprite, en créant deux sprites de même dimension, se trouvant à la même position.

- o Mettre en place un bit qui s'appelle ATTACH dans le mot de contrôle du deuxième sprite.

Les quinze couleurs sont sélectionnées dans l'étendue totale des registres de couleur disponibles pour les sprites - registres de 17 à 31. Le choix de couleurs supplémentaires est possible car chaque pixel contient quatre bits, au lieu de deux seulement dans une situation normale, pour un sprite qui ne serait pas rattaché. Chaque sprite faisant partie de cette paire rattachée apporte deux bits au nombre binaire du sélecteur de couleur. Par exemple, si on utilise les canaux DMA de sprites 0 et 1, les mots du descripteur de couleur de poids fort et de poids faible de la ligne n° 1 des deux structures de données seront réunis dans la ligne n° 1 de l'objet rattaché.

On peut rattacher les sprites selon les combinaisons suivantes :

Sprite 1 avec le sprite 0
Sprite 3 avec le sprite 2
Sprite 5 avec le sprite 4
Sprite 7 avec le sprite 6

Tous ces rattachements, ou aucun, peuvent être actifs dans le même champ d'affichage. A titre d'exemple, présumons de vouloir mettre plus de couleurs dans le sprite d'engin spatial et que nous sommes en train d'utiliser les canaux DMA de sprites 0 et 1. Dans ce sprite, nous avons cinq couleurs, plus le transparent.

0000154444510000
0001564444651000
0015676446765100
0001564444651000
0000154444510000

La première ligne de ce sprite a besoin des quatre mots de données, indiqués dans le Tableau 4-4, pour constituer les chiffres binaires corrects du sélecteur de couleur.

Tableau 4-4 : Mots de Données de la Première Ligne du Sprite Engin Spatial

	NUMERO DU PIXEL															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Ligne 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ligne 2	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
Ligne 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ligne 4	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0

Le sprite ayant un numéro d'ordre plus élevé (le numéro 1, dans cet exemple) apporte les bits de poids fort (les bits les plus à gauche) au nombre binaire. Le mot de données de poids fort de chaque sprite apporte le chiffre le plus à gauche. Par conséquent, les lignes ci-dessus seront écrites dans les structures de données du sprite de la manière suivante :

Ligne 1 Sprite 1 mot de poids fort pour la ligne n° 1 du sprite
Ligne 2 Sprite 1 mot de poids faible pour la ligne n° 1 du sprite
Ligne 3 Sprite 0 mot de poids fort pour la ligne n° 1 du sprite
Ligne 4 Sprite 0 mot de poids faible pour la ligne n° 1 du sprite

Voir la Planche 4-7 pour l'ordre dans lequel ces mots sont stockés en mémoire. Ne pas oublier que ces données sont contenues dans deux structures de sprite.

Les nombres binaires de 0 à 15 sélectionnent les registres de 17 à 31, comme nous le montrons dans le tableau 4-5.

Tableau 4-5 : Registres de Couleur de Sprites Rattachés

Nombre Décimal	Nombre Binaire	Registre de Couleur Numero
0	0000	16 *
1	0001	17
2	0010	18
3	0011	19
4	0100	20
5	0101	21
6	0110	22
7	0111	23
8	1000	24
9	1001	25
10	1010	26
11	1011	27
12	1100	28
13	1101	29
14	1110	30
15	1111	31

* Inutilisé; produit le pixel transparent.

Le rattachement devient effectif seulement si on met à 1 le bit ATTACH, le bit 7 du 2ème mot du registre de contrôle, dans la structure de données des sprites de numéro impair. Par conséquent, dans cet exemple, on mettra à 1 le bit 7 du 2ème mot de contrôle de sprite de la structure de données du sprite n° 1.

Quand on déplace les sprites, la liste Copper doit les garder tous les deux, l'un par rapport à l'autre, exactement dans la même position relative. Si on ne les garde pas ensemble à l'écran, leurs pixels changent de couleur. Chaque sprite retournera trois couleurs, plus le transparent, mais ces couleurs peuvent être différentes des couleurs ordinaires, de celles du sprite qui n'est pas rattaché. La sélection des couleurs des sprites ayant un numéro d'ordre moins élevé se fera dans les registres de couleur 17-19. La sélection des couleurs des sprites ayant un numéro d'ordre plus élevé se fera dans les registres 20, 24 et 28.

La structure de données qui suit est celle d'un engin spatial avec six couleurs réalisé avec deux sprites rattachés.

```
SPRITE0:
   DC.W   $6D60,$7200    ; VSTART = 65, HSTART = 128
   DC.W   $0C30,$0000    ; Premier mot du descripteur de couleur
   DC.W   $1818,$0420
   DC.W   $342C,$0E70
   DC.W   $1818,$0420
   DC.W   $0C30,$0000
   DC.W   $0000,$0000    ; Fin du sprite 0
SPRITE1:
   DC.W   $6D60,$7280    ; Identique au sprite 0 mais avec le bit ATTACH validé
   DC.W   $07E0,$0000    ; Premier mot du descripteur du sprite 1
   DC.W   $0FF0,$0000
   DC.W   $1FF8,$0000
   DC.W   $0FF0,$0000
   DC.W   $07E0,$0000
   DC.W   $0000,$0000    ; Fin du sprite 1
```

MODE MANUEL

Il est presque toujours mieux de charger le sprites en utilisant les canaux DMA automatiques. Parfois, cependant, il est utile de charger ces registres directement à partir d'un des micro-processeurs. On peut activer les sprites "manuellement", quand ils ne sont pas employés par un canal DMA. Le même sprite qui est en train d'afficher une icône, contrôlée par les DMA, proche de la partie supérieure de l'écran, peut aussi être chargé à nouveau manuellement pour afficher une barre verticale en couleur proche de la partie inférieure de l'écran. On peut activer des sprites manuellement même quand le sprite DMA est hors service.

On affiche des sprites manuellement en écrivant dans les registres de données SPRxDATB et SPRDxATA d'un sprite, dans cet ordre. On écrira dans SPRxDATA en dernier car l'adresse "arme" la sortie du sprite, lors de la prochaine comparaison horizontale. Les données écrites seront alors affichées à chaque ligne, à la position horizontale qu'on aura donnée dans la partie "H" des registres de position SPRxPOS et SPRxCTL. Si les données sont inchangées, on aura pour résultat une barre verticale. Si les données sont chargées à nouveau à chaque ligne, on pourra produire un sprite complexe.

On peut mettre fin au sprite (le "désarmer") en écrivant dans le registre SPRxCTL. Si l'on écrit dans le registre SPRxPOS, on peut déplacer le sprite horizontalement à chaque instant, même pendant son utilisation normale.

PARTICULARITES DU 'HARDWARE' DES SPRITES

Les sprites sont produits par l'ensemble des circuits que nous montrons dans la Planche 4-13. Cette planche montre sous forme de diagramme comment une paire de mots de données se transforme en une suite de pixels qui s'affichent à l'écran.

Les éléments qui composent l'ensemble des circuits d'affichage des sprites sont expliqués ci-dessous.

- o Registres de données du sprite. Les registres SPRxDATA et SPRxDATB renferment le modèle de bits qui décrit une ligne horizontale d'un sprite pour chacun des huit sprites. La largeur d'une ligne est de 16 pixels, et chaque ligne est définie par deux mots qui permettent le choix de trois couleurs et d'un transparent.

- o Convertisseurs parallèle-série. Chacun des 16 bits du modèle de bits de données du sprite est envoyé individuellement aux circuits de sélection des couleurs au moment où le pixel associé avec ce bit va être affiché à l'écran.

Dès que les données sont transférées à partir des registres de données du sprite, chaque convertisseur parallèle-série commence le décalage des bits en sortie du convertisseur, le bit le plus significatif (le plus à gauche) en premier. Le décalage intervient une fois pour chaque pixel en basse résolution et continue jusqu'au transfert total des 16 bits vers les circuits d'affichage. Le décalage et la sortie de données ne recommencent pas jusqu'à la fois suivante, où le convertisseur sera à nouveau chargé à partir des registres de données.

Comme l'image vidéo est produite par un faisceau électronique qui balaie l'écran de gauche à droite, l'image sous forme de bits des données concorde exactement avec l'image qui s'affiche réellement à l'écran (les données les plus significatives à gauche).

- o Données vidéo série du sprite. Les données du sprite vont dans le circuit de priorité pour établir la priorité entre sprites et aires de jeu.

- o Registres de position du sprite. Ces registres, qui s'appellent SPRxPOS, renferment la valeur de la position horizontale (valeur X) et celle de la position verticale (valeur Y) de chacun des huit sprites.

- o Registres de contrôle du sprite. Ces registres, qui s'appellent SPRxCTL, renferment la position de fin de chacun des huit sprites et indiquent si un sprite est ou n'est pas rattaché.

- o Compteur de faisceau. Le compteur de faisceau informe le système sur l'emplacement courant du faisceau vidéo en train de produire l'image.

- o Comparateur. Ce dispositif compare la valeur du compteur de faisceau à la valeur Y du registre de position SPRxPOS. Si le faisceau a atteint la position où doit s'afficher le pixel supérieur le plus à gauche du sprite, le comparateur envoie un signal de chargement au convertisseur série-parallèle et l'affichage du sprite peut commencer.

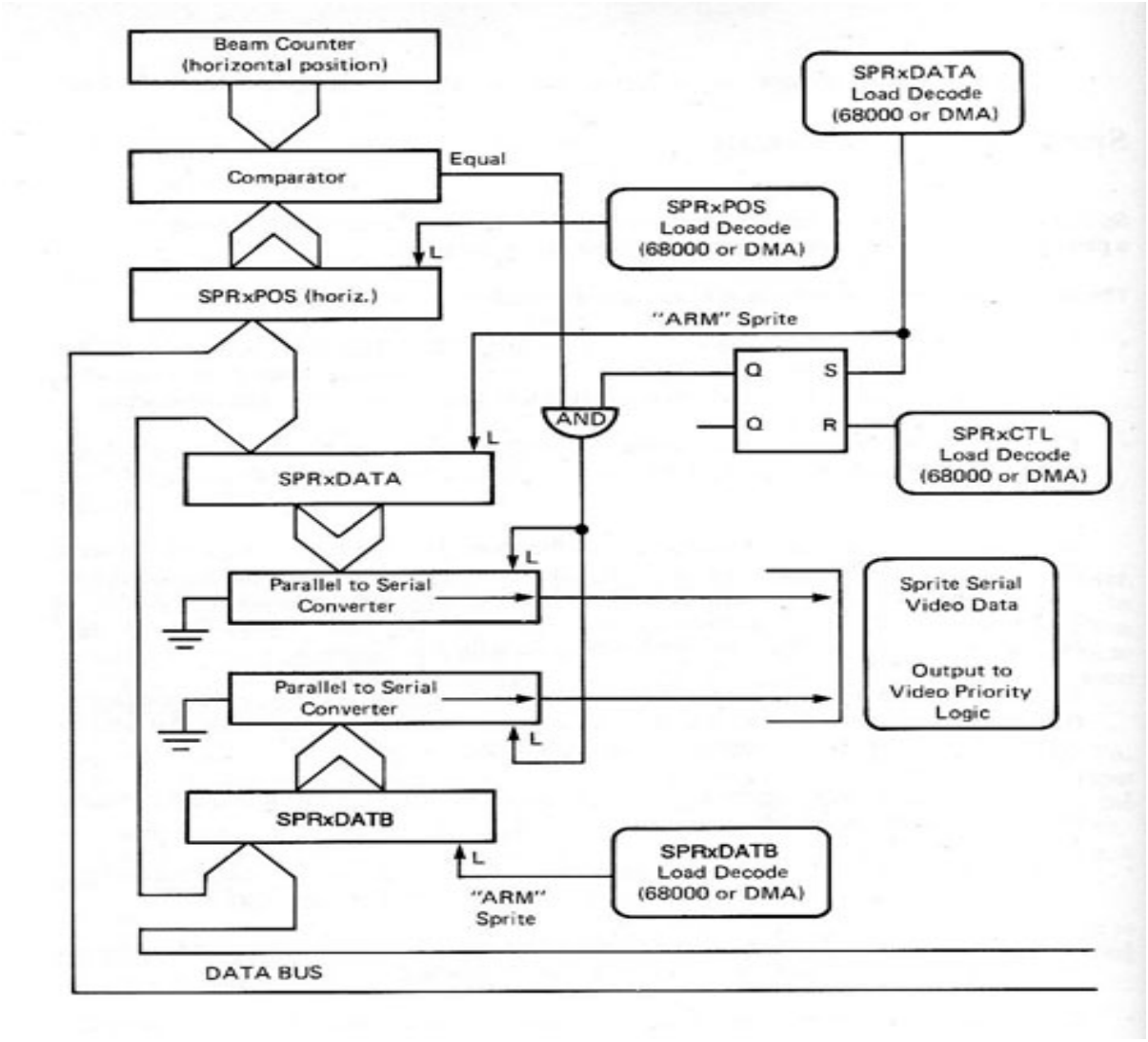


PLANCHE 4-13 : Circuits de Contrôle des Sprites

La Planche 4-13 montre les choses suivantes :

o L'écriture dans les registres de contrôle du sprite met hors service les circuits du comparateur horizontal. Ceci empêche le système d'envoyer toute sortie, à partir des registres de données, au convertisseur série ou à l'écran.

o L'écriture dans le registre de données A du sprite met en service le comparateur horizontal. Ce qui permet de valider une sortie à l'écran quand la position horizontale du faisceau vidéo est égale à la valeur horizontale du registre de position.

o Si le comparateur est en service, les données du sprite sont envoyées à l'affichage, avec le pixel de données du sprite le plus à gauche placé dans la position définie dans la section horizontale de SPRxPOS.

o Tant que le comparateur reste en service, le contenu courant du registre de données du sprite sera envoyé sur la sortie à une position horizontale sélectionée d'une ligne vidéo.

o Les données du registre de données du sprite ne se modifient pas. Celui-ci est parfois réécrit par l'utilisateur ou modifié sous contrôle DMA.

Voyons maintenant comment les éléments décrits ci-dessus produisent un affichage DMA automatique. Quand les sprites sont en mode DMA, le registre 18 bits du pointeur de sprite (composé de SPRxPTH et de SPRxPTL) est employé pour lire les deux premiers mots de la structure de données du sprite. Ces mots contiennent la position initiale et finale du sprite. Puis, les pointeurs écrivent ces mots dans SPRxPOS et dans SPRxCTL. Après écriture, la valeur des pointeurs pointe sur l'adresse du premier mot de données (mot de données de poids faible de la ligne 1 du sprite).

Le fait d'écrire dans le registre SPRxCTL invalide le sprite. A cet instant, le canal DMA de sprite attend jusqu'à ce que la valeur verticale du compteur de faisceau soit égale aux données présentes dans la section VSTART (valeur Y) de SPRxPOS. Lorsque ces deux valeurs concordent, le système valide l'accès aux données du sprite.

Le canal DMA de sprite examine le contenu de VSTOP (dans SPRxCTL, qui indique l'emplacement de la ligne se trouvant après la dernière ligne du sprite) et celui de VSTART (dans SPRxPOS) pour voir combien de lignes de données devront être prélevées. On prélève deux mots par ligne sur la hauteur du sprite et on écrit ces mots dans les registres de données du sprite. Le premier mot est stocké dans SPRxDATA et le deuxième dans SPRxDATB.

Pour chaque ligne de balayage horizontale, le prélèvement et le stockage interviennent pendant l'intervalle de suppression horizontale, dans la partie gauche de l'écran, loin de l'endroit où commence l'affichage. Ceci arme les comparateurs horizontaux du sprite et leur permet de commencer à envoyer les données de sprite en sortie vers l'écran quand la valeur du compteur de faisceau horizontal concorde avec la valeur stockée dans la section HSTART (valeur X) de STRxPOS.

Si la valeur VSTOP - VSTART est égale à zéro, il n'y aura pas de sortie de sprite. Puis, on prélevera la paire de mots de données suivante, mais on ne la stockera pas dans les registres de données du sprite. Cette paire de mots deviendra par contre la paire de mots de données suivante pour SPRxPOS et SPRxCTL.

Si on n'utilise qu'une seule fois un sprite dans un seul champ d'affichage, la paire de mots de données de fin, qui suit les mots du descripteur de couleur du sprite, est chargée automatiquement comme contenu suivant des registres SPRxPOS et SPRxCTL. Pour stopper le sprite après cette première suite de données, la paire de mots en question ne devrait contenir que des zéros.

Ainsi, si l'on a créé un modèle de sprite en mémoire, ce même modèle sera reproduit sous forme de pixels automatiquement sous contrôle des DMA, une seule ligne à la fois.

RECAPITULATION DES REGISTRES DES SPRITES

Il y a huit jeux complets de registres utilisés pour décrire les sprites. Chaque jeu comprend cinq registres. Ici, nous ne décrirons que les registres du sprite 0. Tous les autres sont identiques, si l'on excepte le nom du registre qui devra comprendre le numéro approprié.

POINTEURS

Les pointeurs sont des registres utilisés par le système pour pointer sur les données courantes qu'il faut employer. Lors d'un affichage à l'écran, les registres sont augmentés d'une unité pour pointer les données à utiliser à mesure que l'affichage de l'écran progresse. Par conséquent, les registres de pointeur devront être réécrits au début de la période de suppression verticale.

SPR0PTH et SPR0PTL

Cette paire de registres contient l'adresse 32 bits du mot de données DMA du sprite n° 0.

Voici les noms des registres de pointeur des autres sprites :

SPR1PTH	SPR1PTL
SPR2PTH	SPR2PTL
SPR3PTH	SPR3PTL
SPR4PTH	SPR4PTL
SPRSPTH	SPRSPTL
SPR6PTH	SPR6PTL
SPR7PTH	SPR7PTL

REGISTRES DE CONTROLE

SPR0POS

Celui-ci est le registre de position du sprite 0. Le mot écrit dans ce registre contrôle la position à l'écran à laquelle on devra placer le coin supérieur gauche du sprite. Le bit le plus significatif du premier mot de données sera placé à l'écran à cette position.

NOTE

Les sprites possèdent une résolution de placement à l'écran de 320 par 200, en NTSC (de 320 par 256, en PAL). La résolution des sprites est indépendante de celle de l'aire de jeu.

POSITION DES BITS :

- Les bits 15-8 indiquent la position d'origine verticale, bits V7 - V0.

- Les bits 7-0 indiquent la position d'origine horizontale, bits H8 - H1.

NOTE
Normalement, seul le canal DMA du sprite écrit dans ce registre.
Cf. le détail ci-dessus pour ce qui est de l'organisation des données du sprite.
Habituellement, ce registre est remis à jour directement par les DMA.

SPR0CTL

Normalement, ce registre n'est utilisé que par le canal DMA du sprite. Il renferme l'information de contrôle employée pour contrôler le processus d'acquisition des données du sprite. Positions des bits :

- Les bits 15-8 indiquent la position verticale finale d'une image de sprite, bits V7 - V0.

- Le bit 7 est le bit 'attach'. Ce bit n'est valable que pour les sprites de numéro impair. Il indique que les sprites 0,1 (ou 2,3 ou 4,5 ou 6,7) seront pris en considération ensemble pour l'interprétation des couleurs et, comme tels, ils sont appelés bits de profondeur quatre. Le sprite de numéro impair (avec un numéro d'ordre plus élevé) contient des bits d'une signification binaire plus élevée.

En mode 'attach', les sprites rattachés sont normalement déplacés ensemble horizontalement et verticalement sous le contrôle du processeur. Ceci permet une plus grande sélection de couleurs dans les limites du sprite lui-même. Les sprites, bien que rattachés, sont, cependant, susceptibles de mouvement, individuellement, et adopteront cet ensemble de couleurs plus important seulement si leurs bords se chevauchent.

- Les bits 6-3 sont réservés pour un usage à venir (les mettre à zéro).

- Le bit 2 est le bit V8 du point d'origine verticale.

- Le bit 1 est le bit V8 du point d'arrivée verticale.

- Le bit 0 est le bit H0 du point d'origine horizontal.

Voici les Registres de position et de contrôle des autres sprites :

SPR1POS	SPR1CTL
SPR2POS	SPR2CTL
SPR3POS	SPR3CTL
SPR4POS	SPR4CTL
SPR5POS	SPRSCTL
SPR6POS	SPR6CTL
SPR7POS	SPR7CTL

REGISTRES DE DONNEES

Les registres suivants, bien que définis dans l'espace adressable du processeur principal, d'habitude, ne sont utilisés que par le processeur pour l'affichage. Ce sont les registres qui renferment les données obtenues pendant les cycles DMA.

SPR0DATA, SPR0DATB	Registres de données du Sprite n° 0
SPR1DATA, SPR1DATB	Registres de données du Sprite n° 1
SPR2DATA, SPR2DATB	Registres de données du Sprite n° 2
SPR3DATA, SPR3DATB	Registres de données du Sprite n° 3
SPR4DATA, SPR4DATB	Registres de données du Sprite n° 4
SPR5DATA, SPR5DATB	Registres de données du Sprite n° 5
SPR6DATA, SPR6DATB	Registres de données du Sprite n° 6
SPR7DATA, SPR7DATB	Registres de données du Sprite n° 7

RECAPITULATION DES REGISTRES DE COULEUR DES SPRITES

Les mots de données du sprite sont utilisés pour sélectionner la couleur des pixels du sprite à partir de l'ensemble des registres de couleur comme indiqué dans les tableaux suivants.

Si les combinaisons de bits d'un sprite simple sont celles indiquées dans le tableau 4-6, alors les couleurs seront prises dans les registres indiqués.

Tableau 4-6 : Registres de Couleur d'un Sprite Simple

SPRITES SIMPLES Sprite	Valeur	Registre de Couleur
0 ou 1	00	Inutilisé *
	01	17
	10	18
	11	19
2 ou 3	00	Inutilisé *
	01	21
	10	22
	11	23
4 ou 5	00	Inutilisé *
	01	25
	10	26
	11	27
6 ou 7	00	Inutilisé *
	01	29
	10	30
	11	31

* Sélectionne le mode transparent.

Si les combinaisons de bits sont celles indiquées dans le tableau 4-7, alors les couleurs seront prises dans les registres indiqués.

Tableau 4-7 : Registres de Couleur des Sprites Rattachés

SPRITES RATTACHES

Valeur	Registre de Couleur
0000	Sélectionne le mode transparent
0001	17
0010	18
0011	19
0100	20
0101	21
0110	22
0111	23
1000	24
1001	25
1010	26
1011	27
1100	28
1101	29
1110	30
1111	31

INTERACTION ENTRE SPRITES ET AUTRES OBJETS

Les aires de jeu se partagent l'affichage avec les sprites.
Le Chapitre 7, "Le 'Hardware' de Contrôle du Système" montre comment on peut donner des priorités d'affichage vidéo différentes aux aires de jeu par rapport aux sprites et comment les aires de jeu peuvent entrer en collision (recouvrir) les sprites ou se superposer l'une avec l'autre.

CHAPITRE 5

LE 'HARDWARE' AUDIO

INTRODUCTION

Ce chapitre explique comment accéder directement au 'hardware' audio pour produire des sons. Voici les sujets principaux traités dans ce chapitre :

- Un court aperçu sur la façon dont un ordinateur produit des sons.
- Comment produire des sons simples fixes et changeants et des sons plus complexes.
- Comment utiliser les canaux audio pour des effets spéciaux, en les rattachant, si on le souhaite, pour avoir un son stéréo ou en employant un canal pour en moduler un autre.
- Comment produire des sons de qualité en tenant compte des limitations du système.

Une section, à la fin du chapitre, donne les valeurs à utiliser pour créer des notes de musique de la gamme musicale tempérée.

Ce chapitre n'est pas un tutoriel sur la synthèse du son par ordinateur; une description détaillée sur la création du son par ordinateur demanderait un document bien plus important. Le but, ici, est de montrer le chemin et d'indiquer comment utiliser les possibilités de l'Amiga. La création du son par ordinateur est amusante mais complexe et, d'habitude, cela demande à l'utilisateur beaucoup d'essais et est à l'origine de beaucoup d'échecs - on utilise des instructions pour créer des sons et les écouter, on corrige des paramètres et on les écoute à nouveau et ainsi de suite.

Pour plus de renseignements sur la création musicale par ordinateur, nous recommandons les ouvrages suivants :

- Wayne A. Bateman, Introduction to Computer Music (New York, John Wiley and Sons, 1980).

- Hal Chamberlain, Musical Applications of Microprocessors (Rochelle Park, New Jersey, Hayden, 1980).

INTRODUCTION A LA GENERATION DES SONS

Le son se propage dans l'air et arrive au pavillon de l'oreille sous la forme de cycles répétés de variations de pression ou ondes sonores. On peut représenter les sons par des graphiques qui illustrent comment varie la pression de l'air dans le temps. Les attributs d'un son, tel qu'on l'entend, sont liés à l'aspect du graphique. Si la forme d'onde est régulière et répétitive, elle aura une sonorité de hauteur uniforme (aigüe ou grave), comme celle d'une note de musique seule.
Chaque répétition d'une forme d'onde s'appelle période du son.
Si la forme d'onde est irrégulière, le son présentera peu de variations ou aucune, comme le son d'un choc intense ou celui de l'eau qui coule.
La répétition d'une forme d'onde (sa fréquence)produit des effets sur sa hauteur; les sons avec des fréquences élevées ont une hauteur plus élevée.
L'être humain peut entendre des sons d'une fréquence entre 20 et 20000 périodes par seconde.
L'amplitude de la forme d'onde (le point le plus haut du graphique) est en rapport avec l'intensité du son qu'on perçoit.
Pour finir, l'aspect général d'une forme d'onde détermine sa qualité ou timbre.
La Planche 5-1 montre un type particulier de forme d'onde qu'on appelle sinusoïdale et en représente une période qui a une sonorité simple.

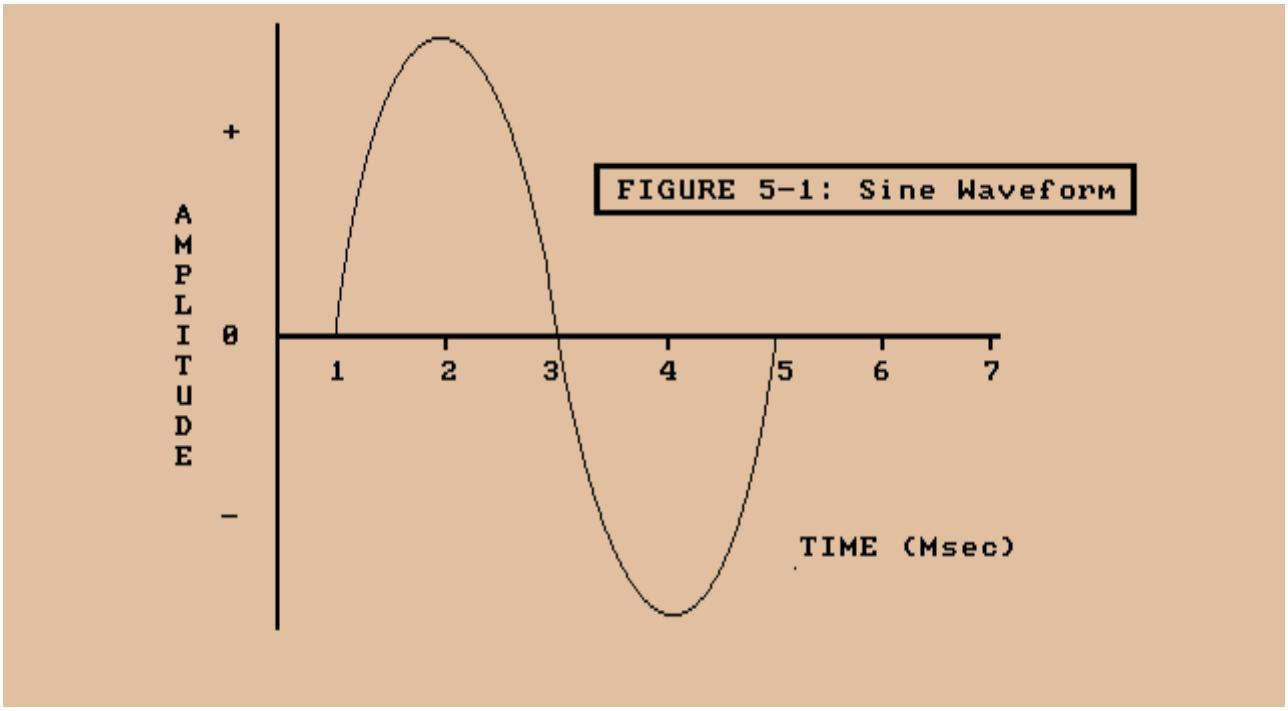


PLANCHE 5-1 : Forme d'Onde Sinusoïdale

Dans l'enregistrement électronique du son et dans les périphériques de sortie, on représente les attributs du son par deux paramètres : l'amplitude et la fréquence. La fréquence est le nombre de périodes par seconde et l'unité de fréquence plus courante est le Hertz (Hz) qui correspond à une période par seconde. Des valeurs plus importantes, ou des fréquences plus élevées, se mesurent en Kilohertz (KHz) ou en Mégahertz (MHz).

La fréquence entretient un rapport très étroit avec la hauteur du son qu'on perçoit. Quand la fréquence augmente, la hauteur s'élève. Cette relation est exponentielle. Une augmentation de 100 à 200 Hertz provoque une forte élévation de la hauteur, mais une augmentation de 1000 Hertz à 1100 Hertz est difficilement perceptible. La hauteur musicale se représente en octaves. Par rapport à un autre, un son qui a une octave en plus aura une fréquence double par rapport à celle du premier son et la hauteur qu'on perçoit sera deux fois plus élevée.

Le deuxième paramètre qui définit une forme d'onde est l'amplitude. Dans un circuit électronique, l'amplitude est en rapport avec la tension ou le courant du circuit. Si un signal est dirigé vers un haut-parleur, alors l'amplitude s'exprime en Watts. L'intensité des sons qu'on perçoit se mesure en décibels (db). L'étendue d'une oreille humaine est d'environ 120 db; 1 db est égal au plus faible son qu'on puisse entendre. En gros, tous les 10 db correspondent à un doublement du son et 1 db est le plus petit changement d'amplitude perceptible dans un son modérément fort. Le volume, qui est l'amplitude du signal sonore en sortie, correspond logarithmiquement au niveau du décibel.

Les deux paramètres fréquence et amplitude d'une onde sinusoïdale sont totalement indépendants. Cependant, lorsqu'on perçoit un son il y a interaction entre volume et hauteur. Les sons dans les basses fréquences diminuent d'intensité plus rapidement que les sons dans les hautes fréquences.

Le troisième attribut d'un son, le timbre, dépend de la présence ou de l'absence de sons supplémentaires ou harmoniques. Toute forme d'onde complexe est un véritable mélange d'ondes sinusoïdales d'amplitude, de fréquence et de phase (le point de départ de l'onde dans l'axe du temps) différentes. Ces éléments qui composent une onde sinusoïdale s'appellent des harmoniques. Une forme d'onde carrée, par exemple, possède un nombre infini infini d'harmoniques.

Pour résumer, tous les sons stables peuvent se décrire par leur fréquence, leur amplitude totale et l'amplitude relative de leurs harmoniques. Les équivalents audibles de ces paramètres sont, respectivement, la hauteur, l'intensité et le timbre. Un son changeant est un son stable dont les paramètres changent dans le temps.

Dans la production électronique du son, un appareil analogique, comme un magnétophone, enregistre les formes d'onde d'un son et ses fréquences périodiques comme une représentation continuellement variable de la pression atmosphérique. Le magnétophone reproduit ensuite le son en envoyant les formes d'onde à un amplificateur où celles-ci sont transformées en formes d'onde analogiques basse tension. L'amplificateur envoie ces formes d'onde sous forme de tension dans un haut-parleur qui les transforme en vibrations de pression atmosphérique que l'auditeur percevra comme un son.

Un ordinateur ne peut stocker l'information d'une forme d'onde analogique. Dans la production du son par ordinateur, une forme d'onde devra être représentée comme une chaîne de nombres finie. On opère cette transformation en divisant l'axe du temps du graphique de la forme d'onde en segments égaux, chacun desquels représente un instant suffisamment court pour que la forme d'onde ne se modifie pas trop. Chaque point résultant s'appelle un échantillon. Ces échantillons sont stockés en mémoire et l'on pourra les entendre par la suite à une fréquence qu'on a la possibilité de déterminer. L'ordinateur envoie ces échantillons à un convertisseur numérique/analogique (DAC) qui les transforme en une forme d'onde analogique en tension de sortie. Pour produire des sons, les formes d'onde analogiques sont d'abord dirigées vers un amplificateur, puis vers un haut-parleur.

La planche 5-2 montre des exemples d'onde sinusoïdale, d'onde carrée et d'onde triangulaire, avec un tableau des échantillons de chacune.

NOTE

Les illustrations ne sont pas à réduire et il y a moins de points dans les formes d'onde qu'il n'y d'échantillons dans le tableau. Les deux valeurs de l'axe d'amplitude 127 et -128 représentent la limite supérieure et celle inférieure de l'amplitude relative.

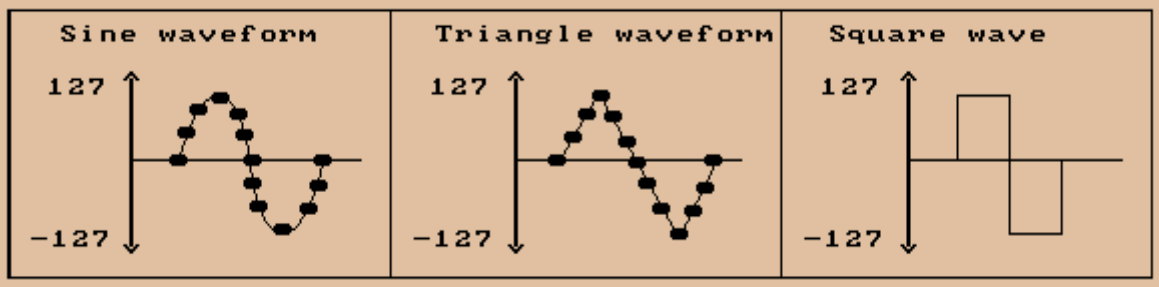


FIGURE 5-2: Digitised Amplitude Values.

PLANCHE 5-2 : Valeurs d'Amplitude Numérisées

VALEURS D'AMPLITUDE SELON LE TEMPS

TEMPS	SINUSOIDALE	CARREE	TRIANGULAIRE
0	0	100	0
1	39	100	20
2	75	100	40
3	103	100	60
4	121	100	80
5	127	100	100
6	121	100	80
7	103	100	60
8	75	100	40
9	39	100	20
10	0	-100	0
11	-39	-100	-20
12	-75	-100	-40
13	-103	-100	-60
14	-121	-100	-80
15	-127	-100	-100
16	-121	-100	-80
17	-103	-100	-60
18	-75	-100	-40
19	-39	-100	20

LE 'HARDWARE' SON DE L'AMIGA.

L'Amiga possède quatre canaux 'hardware' pour le son. On peut programmer chacun de ces canaux indépendamment pour produire des effets sonores complexes. On peut aussi rattacher les canaux pour qu'un canal module le son d'un autre canal ou combiner deux canaux pour des effets stéréo.

Chaque canal audio comprend un convertisseur numérique/analogique 8 bits, piloté par un canal d'accès direct en mémoire (DMA). Le DMA audio peut prélever deux échantillons de données pendant chaque ligne de balayage vidéo horizontale. Pour des sons simples et stables, le DMA peut jouer une forme d'onde automatiquement de manière répétitive; on peut aussi programmer toute sorte d'effets sonores complexes.

Il existe deux méthodes de production simple des sons sur l'Amiga - la génération automatique (DMA) et la génération directe des sons (sans DMA). Quand on utilise la génération de sons automatique, le système prélève les données automatiquement par un accès direct en mémoire.

CREATION ET ECOUTE D'UN SON

Cette section montre comment créer un son simple et stable et comment l'écouter. Plusieurs concepts de base qui s'appliquent à la génération du son sur l'Amiga seront introduits ici.

Pour produire une sonorité stable, on devra suivre les étapes principales suivantes :

1. Décider quel canal on va employer.

2. Définir la forme d'onde et créer la table d'échantillons en mémoire.

3. Mettre en place les registres qui informent le système sur l'endroit où se trouvent les données et sur la longueur de celles-ci.

4. Sélectionner le volume auquel le son devra être entendu.

5. Sélectionner la fréquence d'échantillonnage ou le taux de sortie des données.

6. Sélectionner un canal audio et faire démarrer le DMA.

CHOIX DU CANAL A UTILISER

L'Amiga possède quatre canaux audio. Les canaux 0 et 3 sont reliés à la prise de sortie stéréo gauche. Les canaux 1 et 2 sont reliés à la prise de sortie stéréo droite. Sélectionnez le canal du côté où devra se faire la sortie.

CREATION DES DONNEES DE LA FORME D'ONDE

La forme d'onde utilisée à titre d'exemple dans cette section est une onde sinusoïdale simple, produisant un son pur. Pour économiser de la mémoire, on ne définit normalement en mémoire qu'une seule période de la totalité de la forme d'onde. Pour un son stable, non changeant, les valeurs au début et à la fin de la forme d'onde, ainsi que la pente ou la direction des données au début et à la fin devront être en rapport étroit. Cela garantit qu'une répétition continue de la forme d'onde sonne comme un flot sonore continu.

Les données sons sont organisées comme un ensemble d'articles de données 8 bits; chaque article est un échantillon de la forme d'onde. Chaque mot de données prélevé pour le canal audio est composé de deux échantillons. Les valeurs de ces échantillons s'échelonnent entre -128 et +127.

A titre d'exemple, l'ensemble de données qu'on montre ci-dessous produit une approximation proche d'une onde sinusoïdale.

NOTE

Les données sont stockées dans des adresses d'octets successives avec la première valeur d'amplitude numérisée à l'adresse d'octet la plus basse, la deuxième à l'adresse d'octet suivante et ainsi de suite. Notez, en outre, que le premier octet de données doit commencer à une limite d'adresse divisible par deux. Ceci provient de ce que le DMA audio prélève un mot (16 bits) à la fois et qu'il utilise l'échantillon qu'il lit comme deux octets de données.

Pour utiliser le canal audio 0, écrire l'adresse de "audiodata" dans AUD0LC, où les données audio sont organisées comme indiqué ci-dessous. Pour plus de simplicité, dans le tableau ci-dessous, nous avons mis "AUDxLC" à la place de la combinaison des deux registres d'adresse véritables (AUDxLCH et AUDxLCL). Pour que les canaux DMA audio soient en mesure de prélever les données, l'adresse de données sur qui pointe AUD0LC doit se trouver quelque part en mémoire 'chip'.

TABLEAU 5-1 : Suite de Données d'Echantillons Audio du Canal 0

audiodata --->	AUD0LC *	100	98
	AUD0LC +2 **	92	83
	AUD0LC +4	71	56
	AUD0LC +6	38	20
	AUD0LC +8	0	-20
	AUD0LC +10	-38	-56
	AUD0LC +12	-71	-83
	AUD0LC +14	-92	-83
	AUD0LC +16	-100	-98
	AUD0LC +18	-92	-83
	AUD0LC +20	-71	-56
	AUD0LC +22	-38	-20
	AUD0LC +24	0	20
	AUD0LC +26	38	56
	AUD0LC +28	71	83
	AUD0LC +30	92	98

NOTES

* Les données audio se trouvent à une limite d'adresse divisible par deux.

** AUD0LC est mis à la place de AUD0LCL et AUD0LCH.

INFORMER LE SYSTEME SUR LES DONNEES

Pour prélever les données sons pour le canal audio, le système a besoin de savoir où elles se trouvent ainsi que d'en connaître la longueur (en mots).

Les registres d'adresse AUDxLCH et AUDxLCL contiennent respectivement les trois bits de poids fort et les quinze bits de poids faible de l'adresse initiale des données audio. Comme ces deux registres d'adresse sont contigus, le fait d'écrire un mot long dans AUDxLCH transfère en même temps l'adresse des données audio dans les deux emplacements mémoire. L' "x" présent dans le nom des registres de données indique le numéro du canal audio où se produira la sortie. Les canaux sont numérotés 0, 1, 2 et 3.

Ces registres sont des registres d'adresse distincts des registres de pointeur. Il ne faudra indiquer qu'une seule fois le contenu de ces registres; il ne sera pas nécessaire de les réinitialiser si on souhaite que le canal audio continue à entretenir la répétition d'une même forme d'onde. A chaque fois que le système prélève de la zone mémoire réservée aux données le dernier mot audio, il utilise le contenu de ces registres d'adresse pour retrouver le début des données. En présumant que le premier mot des données commence à l'adresse "audiodata" et qu'on utilise la canal 0, voici comment mettre en place les registres d'adresse :

```
WHERE0DATA:
    LEA CUSTOM,a0               ; Adresse de base des puces...
    LEA AUDIODATA,a1
    MOVE.L a1,AUDOLCH(a0)       ; Mettre l'adresse (32 bits) dans le registre d'adresse.
```

La longueur des données est le nombre d'échantillons d'une forme d'onde divisé par 2 ou alors le nombre de mots de l'ensemble des données. En utilisant l'ensemble de données d'échantillons ci-dessus, la longueur des données est de 16 mots. On écrira cette longueur dans le registre audio de longueur de données du canal concerné. Le registre de longueur s'appelle AUDxLEN, où "x" fait référence au numéro du canal. On mettra le registre de longueur AUD0LEN à 16 comme nous le montrons ci-dessous.

```
SETAUDOLENGTH:
    LEA CUSTOM,a0               ; Adresse de base des puces...
    MOVE.W #16,AUD0LEN(a0)      ; Stocker la longueur...
```

SELECTION DU VOLUME

Ici le volume est le volume général de tous les sons en provenance des canaux audio. La puissance relative des sons qui vous concernera quand vous combinerez des notes, est déterminée par l'amplitude de la forme d'onde. Pour chaque canal audio, il existe un registre de volume 6 bits. Pour contrôler le volume du son en sortie du canal audio sélectionné, on écrit la valeur voulue dans le registre AUDxVOL, où on remplace "x" par le numéro du canal. On peut indiquer des valeur de 64 à 0. Ces valeurs de volume correspondent à la valeur en décibels de chacun des 65 niveaux de volume.

Pour une sortie typique avec un volume de 64, avec des valeurs de données maximales de -128 à 127, la tension de sortie se situe entre +0,4 volts et -0,4 volts. Certains niveaux de volume et leurs valeurs correspondantes en décibels sont indiqués dans le Tableau 5-2.

TABLEAU 5-2 : Valeurs de Volume

VOLUME	VALEUR de DECIBELS	
64	0	(volume maximal)
48	-2.5	
32	-6.0	
16	-12.0	(12 db au-dessous du volume avec niveau maximal)

Pour instaurer un volume de 64 à 0, on en écrit la valeur dans les bits 5-0 de AUD0VOL. Par exemple:

```
SETAUDOVOLUME:
      LEA CUSTOM,a0
      MOVE.W #48,AUD0VOL(a0)
```

On indique les décibels en tant que valeurs négatives à partir d'une valeur maximale 0 car la manière dont un appareil d'enregistrement, comme un magnétophone, indique le niveau d'enregistrement est négative. D'ordinaire, un appareil enregistreur possède un cadran qui indique le 0 comme niveau d'enregistrement optimal. Tout ce qui est inférieur à cette valeur optimale est indiqué comme quantité négative.

SELECTION DU TAUX DE SORTIE DES DONNEES

La hauteur du son produit par une forme d'onde dépend de sa fréquence. Pour informer le système sur la fréquence à utiliser, il faudra spécifier la fréquence d'échantillonnage. Cette fréquence indique le nombre de signaux par seconde de l'horloge du système, ou intervalles de synchronisation, qui devront s'écouler entre chaque envoi d'échantillons (octets de données audio) au convertisseur numérique/analogique du canal audio. Pour chaque canal, il existe un registre de fréquence. La valeur de ce registre est utilisée pour les besoins du compte à rebours; chaque fois que le registre compte à rebours vers 0, on prélève un autre échantillon de l'ensemble des données de la forme d'onde en sortie. En unités, cette valeur de fréquence représente le nombre de signaux d'horloge par échantillon. La valeur de fréquence minimale qu'on doit utiliser est de 124 signaux par échantillon en NTSC (123 en PAL) et la valeur maximale est de 65535. Ces limitations s'appliquent aussi bien aux machines en PAL qu'à celles en NTSC. Pour un son de bonne qualité, il existe d'autres contraintes au sujet de la fréquence d'échantillonnage (cf. la section "Production d'un Son de Bonne Qualité").

NOTE

Une valeur de fréquence basse correspond à un son de fréquence plus élevée et une valeur de fréquence haute correspond à un son de fréquence plus basse.

LIMITATIONS LORS DU CHOIX DE LA PERIODE D'ECHANTILLONNAGE

La période d'échantillonnage est limitée par le nombre de cycles DMA qu'on réserve à un canal audio. On réserve à chacun de ces canaux une plage de temps par ligne de balayage horizontale dans l'affichage d'écran. Un canal audio peut prélever deux échantillons de données à chaque ligne de balayage horizontale. Le calcul ci-dessous donne le taux d'échantillonnage maximal en échantillons par seconde.

2 échantillons/ligne * 262.5 trames/trame * 59.94 trames/seconde = 31,469 échantillons/seconde

Un chiffre de 31469 est une valeur maximale théorique. Pour économiser les tampons, le 'hardware' est conçu pour gérer 28867 échantillons/seconde. L'intervalle de synchronisation est de 279,365 nano-secondes ou 0,279365 micro-secondes. Le taux d'échantillonnage maximal de 28867 échantillons par seconde est de 34,642 micro-secondes par échantillon (1/28867 = 0,000034642). La formule pour calculer la période d'échantillonnage est la suivante :

$$\text{Valeur de période} = \frac{\text{intervalle d'échantillon}}{\text{intervalle de synchro}} = \frac{\text{constante d'horloge}}{\text{échantillons par seconde}}$$

Ainsi, on obtient la valeur de période minimale en divisant 34,642 micro-secondes -par échantillon- par le nombre de micro-secondes présentes dans un intervalle :

$$\text{Période Max} = \frac{34{,}642 \text{ micro-sec./échant.}}{0{,}279365 \text{ micro-sec./interv.}} = 124 \text{ int.s de synchro/échantillon}$$

ou alors :

$$\text{Période Min} = \frac{3579545 \text{ signaux/seconde}}{28867 \text{ échantillons/seconde}} = 124 \text{ signaux/échantillon}$$

Par conséquent, on devra écrire dans le registre de période une valeur minimale de 124 pour garantir que les DMA audio du système puissent prélever l'échantillon de données suivant. Si la valeur de période est inférieure à 124, pendant que le compte du cycle a atteint 0, les DMA audio n'auront pas eu un temps suffisant pour prélever l'échantillon de données suivant et utiliseront l'échantillon précédent.

28867 échantillons/seconde est aussi le taux d'échantillonnage maximal dans un système PAL. Ainsi, dans un de ces systèmes, on devra écrire une valeur minimale de 123 signaux/échantillon dans le registre de période.

	HORLOGE VALEURS		
	NTSC	PAL	UNITES
Constante d'Horloge	3579545	3546895	signaux par seconde
Intervalle de Synchro	0,279365	0,281937	micro-secondes par intervalle

NOTE

L'intervalle de synchronisation est dérivé de la constante d'horloge, où

$$\text{intervalle de synchro} = \frac{1}{\text{constante d'horloge}}$$

puis on réduit le résultat en micro-secondes. Les "Signaux" et les "Intervalles de synchronisation" se réfèrent à la même chose.

INDICATION DE LA VALEUR PERIODE

Après avoir sélectionné l'intervalle voulu entre échantillons de données, on pourra calculer la valeur à mettre dans le registre de période en utilisant la formule pour la période :

$$\text{Valeur de période} = \frac{\text{intervalle voulu}}{\text{intervalle de synchro}} = \frac{\text{constante d'horloge}}{\text{échantillons par seconde}}$$

A titre d'exemple, disons que l'on souhaite produire une onde sinusoïdale d'un Khz, en utilisant un tableau de huit échantillons de données (quatre mots de données) (cf. Planche 5-3).

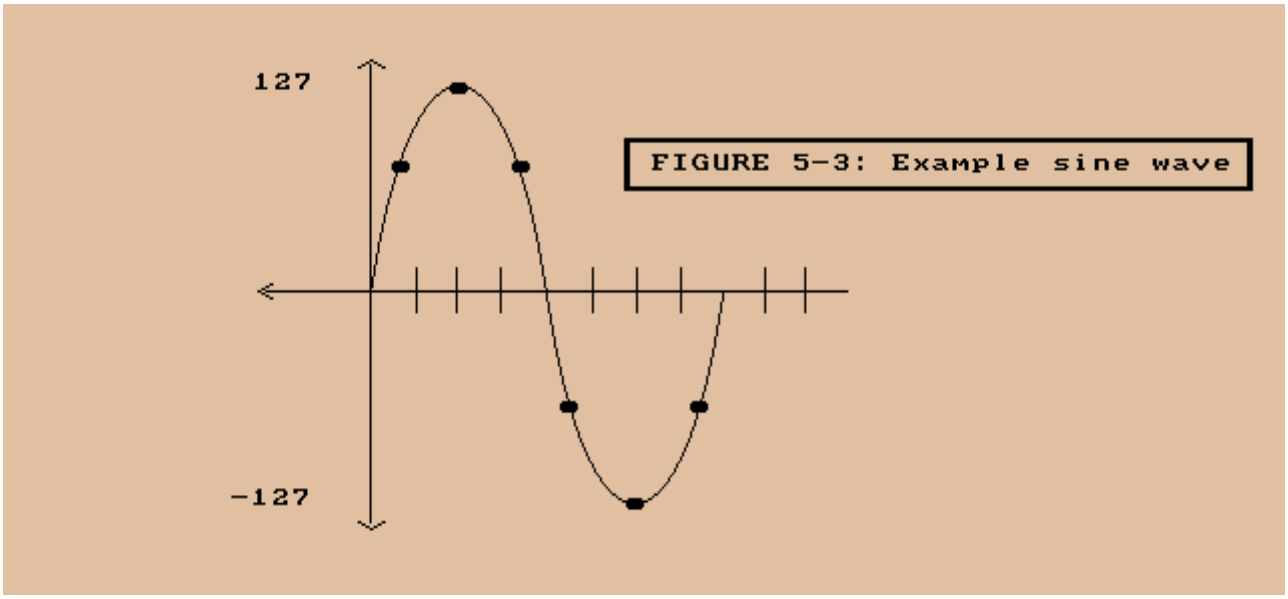


PLANCHE 5-3 : Exemple d'Onde Sinusoïdale

Valeurs échantillonnées : 0
90
127
90
0
-90
-127
-90

Pour envoyer en sortie la série de huit échantillons à 1 Khz (1000 cycles par seconde), chaque cycle complet sera envoyé sur la sortie en 1/1000ème de seconde. Par conséquent, chaque valeur individuelle devra être prélevée en 1/8ème de ce temps. Ce qui équivaut à 1000 micro-secondes par forme d'onde ou à 125 micro-secondes par échantillon. Pour produire correctement une telle forme d'onde, la valeur de période devrait être :

$$\text{Valeur Période} = \frac{125 \text{ micro-sec./échant.}}{0{,}279365 \text{ micro-sec./interv.}} = 447 \text{ Intv.s synchro/échant.}$$

Pour initialiser le registre de période, on doit écrire la valeur de période dans le registre AUDxPER, où "x" est le numéro du canal qu'on est en train d'employer. Par exemple, les instructions suivantes montrent comment écrire une valeur de période de 447 dans le registre de période du canal 0.

```
SETAUDOPERIOD:
        LEA    CUSTOM,a0
        MOVE.W #447,AUD0PER(a0)
```

Pour produire des sons de bonne qualité, en évitant la distorsion, on devra observer les limitations sur les valeurs de période dont nous parlons dans la section "Production d'un Son de Bonne Qualité".

Pour la relation entre période et hauteur musicale, voyez la section à la fin de ce chapitre qui contient un listage de la gamme musicale tempérée.

ENTENDRE LA FORME D'ONDE

Après avoir défini l'emplacement des données audio, leur longueur, le volume et la période, on peut entendre la forme d'onde en ouvrant le DMA du canal audio concerné. Cette action fait démarrer la sortie des sons. Une fois qu'il a commencé, le DMA continue tant qu'on ne l'arrête pas expressément. Ainsi, la forme d'onde continue à être jouée et rejouée en produisant une sonorité stable. Le système utilise la valeur des registres d'adresse chaque fois qu'il rejoue la forme d'onde.

Pour tout DMA audio qui doit intervenir, on devra valider le bit DMAEN de DMACON. Lorsque DMAEN et AUDxEN sont tous les deux validés, le DMA ouvre le canal x. Tous ces bits et leur signification sont indiqués dans le Tableau 5-3.

TABLEAU 5-3 : Bits de Validation du Canal Audio et du DMA

REGISTRE DMACON

Bit	Nom	Fonction
15	SET/CLR	Quand on écrit un 1 dans ce bit, celui-ci initialise tous les bits de DMACON dont la position du bit correspondante est aussi à 1, en négligeant tous les autres bits.
9	DMAEN	Tout accès direct mémoire ne peut se se produire que si ce bit est à 1.
3	AUD3EN	Valide le canal audio 3.
2	AUD2EN	Valide le canal audio 2.
1	AUD1EN	Valide le canal audio 1.
0	AUD0EN	Valide le canal audio 0.

Par exemple, si on utilise le canal 0, alors on écrit un 1 dans le bit 9 pour valider le DMA et un 1 dans le bit 0 pour valider le canal audio, comme nous le montrons ci-dessous :

```
BEGINCHAN0:
        LEA    CUSTOM,a0
        MOVE.W #(DMAF_SETCLR!DMAF_AUD0!DMAF_MASTER),DMACON(a0)
```

ARRET DU DMA AUDIO

A chaque instant, on peut arrêter le canal en écrivant un 0 dans le bit de AUDxEN. Cependant, il ne sera pas possible de reprendre la sortie au même point de la forme d'onde en écrivant seulement un 1 dans le bit en question. La validation d'un canal audio fait presque toujours commencer la sortie des données à partir du début de la liste de données sur qui pointent les registres d'adresse du canal en question. Si le canal est invalidé pour un court instant (moins de deux périodes d'échantillonnage), il se maintient et peut continuer ainsi à partir d'où il a été arrêté.

L'exemple qui suit indique comment arrêter le DMA audio d'un canal.

```
STOPAUDCHAN0:
    LEA    CUSTOM,a0
    MOVE.W #(DMAF_AUD0),DMACON(a0)
```

RECAPITULATION

Voici les étapes nécessaires à suivre pour produire une sonorité stable :

1. Définir la forme d'onde.

2. Créer l'ensemble des données contenant les doublets des échantillons de données (mots de données). D'ordinaire, un ensemble de données contient la définition d'une seule forme d'onde.

3. Mettre en place les registres d'adresse.

AUDxLCH (les trois bits de poids fort)
AUDxLCL (les trois bits de poids faible)

4. Mettre en place le registre de longueur, AUDxLEN, avec le nombre de mots de données à prélever avant de commencer à l'adresse courante de AUDxLC.

5. Mettre en place le registre de volume, AUDxVOL.

6. Mettre en place le registre de période, AUDxPER

7. Démarrer le DMA audio en écrivant un 1 dans le bit 9, DMAEN, en même temps qu'un 1 dans le bit SET/CLR et un 1 dans la position du bit de AUDxEN du canal ou des canaux qu'on souhaite ouvrir.

EXEMPLE

Dans cet exemple, qui rassemble tous les segments de programme des sections précédentes, on entendra une onde sinusoïdale sur le canal 0. L'exemple suppose l'accès exclusif au 'hardware' audio et ne fonctionnera pas dans un environnement multitâche.

```
MAIN:
    LEA CUSTOM,a0           ; Adresse de base des puces spécialisées
    LEA SINEDATA(pc),a1     ; Adresse de données sur le registre d'adresse audio 0

WHEREODATA:
    MOVE.L a1,AUD0LCH(a0)   ; Le 68000 écrit dans celui-ci, comme s'il s'agissait d'un registre
                            ; 32 bits, à l'emplacement destiné aux bits de poids faible
                            ; (commun à toutes les adresses et aux registres de pointeur du système).

SETAUDOLENGTH:
    MOVE.W #4,AUD0LEN(a0) ; Mettre en place la longueur en mots
```

```
SETAUDOVOLUME:
    MOVE.W #64,AUD0VOL(a0)          ; Utiliser le volume maximal

SETAUDOPERIOD:
    MOVE.W #447,AUD0PER(a0)

BEGINCHAN0:
    MOVE.W #(DMAF_SETCLR!DMAF_AUD0!DMAF_MASTER),DMACON(a0)
    RTS                             ; Retourner au programme principal
    DS.W 0                          ; S'assurer que c'est divisible par deux

SINEDATA:
    DC.B 0, 90, 127, 90, 0, -90, -127, -90

    END
```

PRODUCTION DE SONS COMPLEXES

En plus des sonorités simples, il est possible de créer des sons plus complexes, comme ceux provenant de notes de musique différentes, reliées ensemble dans une mélodie à une voix, ceux de différentes notes jouées en même temps ou même des sons modulés.

RACCORDER LES SONS

On raccorde les sons en écrivant dans le registre d'adresse et dans celui de longueur, en faisant démarrer la sortie audio et en écrivant à nouveau dans ces registres pour préparer la forme d'onde successive qu'on veut raccorder à la précédente. Tout ceci est rendu facile par la synchronisation des interruptions audio et par l'existence de registres de sauvegarde. Les registres d'adresse et de longueur sont lus par le canal DMA avant que ne commence la sortie audio. Ensuite le canal DMA stocke les valeurs dans les registres de sauvegarde. Dès que les registres originaux ont été lus par le canal DMA, on peut en modifier la valeur sans déranger l'opération qu'on a lancée avec le contenu original du registre. Par conséquent, il est possible d'écrire dans le contenu de ces registres, de lancer une sortie audio et, ensuite, de réécrire dans les registres pour préparer la prochaine forme d'onde qu'on veut raccorder à la première.

Les interruptions interviennent immédiatement après que le canal DMA audio a lu les registres d'adresse et de longueur et stocké leur valeur dans les registres de sauvegarde. Dès que l'interruption s'est manifestée, on peut réécrire dans les registres en question l'adresse et la longueur du segment de la forme d'onde suivante. Cette combinaison de registres de sauvegarde et de synchronisation des interruptions permet de garder un pied en dehors du canal DMA audio, tout en permettant d'avoir un son continu et uniforme en sortie.

Si on ne réécrit pas dans les registres, le système rejoue la forme d'onde courante. A chaque fois que le compteur de longueur atteint zéro, les deux registres, celui d'adresse et celui de longueur, sont rechargés avec les mêmes valeurs pour pouvoir continuer à avoir une sortie audio.

EXEMPLE

Cet exemple détaille l'action du DMA audio du système en mode pas-à-pas.

Supposons qu'on veuille raccorder une forme d'onde sinusoïdale et une forme d'onde triangulaire bout-à-bout pour un effet audio particulier, en faisant l'alternance entre les deux. La séquence qui suit montre l'action du programme et son interaction avec le système DMA audio. L'exemple présuppose que la période, le volume et la longueur des données soient les mêmes pour l'onde sinusoïdale et pour l'onde triangulaire.

PROGRAMME D'INTERRUPTION

Si (onde = triangulaire)
    écrire dans AUD0LCL l'adresse des données de l'onde sinusoïdale

Autrement si (onde = sinusoïdale)
    écrire dans AUD0LCL l'adresse des données de l'onde triangulaire

PROGRAMME PRINCIPAL

1. Mettre en place le volume, la période et la longueur.
2. Ecrire dans AUD0LCL l'adresse des données de l'onde sinusoïdale.
3. Faire démarrer le DMA.
4. Continuer avec quelque chose d'autre.

REPONSE DU SYSTEME

Dès que le DMA démarre,

a. Copier dans le registre de longueur de "sauvegarde" à partir d'AUDOLEN.

b. Copier dans le registre d'adresse de "sauvegarde" à partir d'AUDOLCL (il sera utilisé comme pointeur pour indiquer le mot de données courant à prélever).

c. Créer une interruption pour le 68000 l'informant qu'il a des copies de travail complètes du prélèvement des registres d'adresse et de longueur.

d. Démarrer le prélèvement des données audio et faire chaque prélevement dans la plage de temps impartie aux DMA.

ENTENDRE PLUSIEURS SONS EN MEME TEMPS

On peut entendre des sons multiples soit en utilisant plusieurs canaux de manière indépendante, soit en additionnant des échantillons en différents ensembles de données qui joueront l'ensemble des données additionnées par le biais d'un même canal.

Étant donné que les quatre canaux audio sont programmables indépendamment, chaque canal possède son propre ensemble de données; ainsi, on pourra entendre un son ou une note de musique différents dans chaque canal.

MODULATION DU SON

Pour avoir des effets audio plus complexes, on peut se servir d'un canal audio pour en moduler un autre. Ceci accroît la gamme et le type d'effets qu'on peut produire. On peut moduler la fréquence et l'amplitude d'un canal ou réaliser les deux types de modulation en même temps.

La modulation d'amplitude affecte le volume de la forme d'onde. On l'utilise souvent pour créer des effets de trémolo ou de vibrato. La modulation de fréquence affecte la période de la forme d'onde. Bien que la forme d'onde elle-même reste inchangée, la modulation de fréquence en accroît ou en diminue la hauteur.

Le système utilise un canal pour en moduler un autre lorsqu'on relie deux canaux ensemble. Les bits 'attach' du registre ADKCON contrôlent comment sont interprétées les données en provenance d'un canal audio (cf. le tableau ci-dessous). D'ordinaire, chaque canal produit un son quand on le valide. Si on met à 1 le bit 'attach' d'un canal audio, ce canal cesse de produire un son et ses données sont utilisées pour moduler le son du canal suivant qui a le numéro d'ordre plus élevé. Si on emploie un canal comme modulateur, les mots qui constituent son ensemble de données ne sont plus traités comme des octets individuellement. Au contraire, ils sont utilisés comme des mots destinés au "modulateur". Les mots de données en provenance du canal modulateur sont écrits dans les registres correspondants du canal modulé chaque fois que le registre de période du canal modulateur décroche.

Pour moduler seulement l'amplitude de la sortie audio, on doit relier un canal en tant que modulateur de volume. Il faut définir l'ensemble de données du canal modulateur comme une suite de mots, chacun contenant l'information pour le volume en respectant le format suivant :

BITS	FONCTION
15 - 7	Inutilisés
6 - 0	Information pour le volume, V6-V0

Pour moduler seulement la fréquence, on doit relier un canal comme modulateur de période. Il faut définir l'ensemble de données du canal modulateur comme une suite de mots contenant l'information sur la période en respectant le format suivant :

BITS	FONCTION
15 - 0	Information sur la période, P15-P0

Si l'on veut moduler la période et le volume ensemble dans un même canal, il faut relier le canal en tant que modulateur de période et de fréquence à la fois. Par exemple, si on emploie le canal 0 pour moduler la période et la fréquence du canal 1, il faudra valider deux bits 'attach' - le bit 0 pour moduler le volume et le bit 4 pour moduler la période. Lorsque le volume et la période sont modulés ensemble, les mots du jeu de données du canal modulateur sont définis alternativement comme information sur le volume et sur la période.

L'ensemble de données d'exemple du Tableau Table 5-4 montre la différence d'interprétation des données quand un canal est employé directement pour l'audio, quand il est relié comme modulateur de volume, quand il est relié comme modulateur de période et quand il est relié comme modulateur de période et de volume en même temps.

TABLEAU 5-4 : Interprétation des Données en Mode "Attach"

DATA WORDS	INDEPENDANT (NE MODULE RIEN)	MODULATION DE PERIODE ET VOLUME	MODULATION DE PERIODE	MODULATION DE VOLUME
Mot 1	|donnée|donnée|	|volume de l'autre canal|	|période|	|volume|
Mot 2	|donnée|donnée|	|période de l'autre canal|	|période|	|volume|
Mot 3	|donnée|donnée|	|volume de l'autre canal|	|période|	|volume|
Mot 4	|donnée|donnée|	|période de l'autre canal|	|période|	|volume|

La longueur des ensembles de données du modulateur et celle des canaux modulés sont totalement indépendantes.

Les canaux sont reliés par le système dans un ordre prédéfini, comme nous le montrons dans le Tableau 5-5. Pour relier un canal en tant que modulateur, on met à 1 son bit 'attach'. Si on met à 1 le bit 'attach' du volume ou celui de la période d'un canal, la sortie audio du canal sera invalidée; le canal sera relié au canal suivant ayant le numéro d'ordre plus élevé, comme nous l'indiquons dans le Tableau 5-5. Comme un canal relié module toujours le canal suivant ayant le numéro d'ordre le plus élevé, il n'est pas possible de relier le canal 3 à un autre canal. En écrivant un 1 dans les bits de modulation du canal 3, on ne fait qu'invalider la sortie audio du canal en question.

TABLEAU 5-5 : Connexion de Canaux pour une Modulation

REGISTRE ADKCON

Bit	Nom	Fonction
7	ATPER3	Utiliser le canal audio 3 pour ne rien moduler (invalide la sortie audio du canal 3)
6	ATPER2	Utiliser le canal audio 2 pour moduler la période du canal 3
5	ATPER1	Utiliser le canal audio 1 pour moduler la période du canal 2
4		ATPER0 Utiliser le canal audio 0 pour moduler la période du canal 1
3		ATVOL3 Utiliser le canal audio 3 pour ne rien moduler (invalide la sortie audio du canal 3)
2		ATVOL2 Utiliser le canal audio 2 pour moduler le volume du canal 3
1		ATVOL1 Utiliser le canal audio 1 pour moduler le volume du canal 2
0		ATVOL0 Utiliser le canal audio 0 pour moduler le volume du canal 1

PRODUCTION DE SONS DE BONNE QUALITE

Quand on essaie de créer des sons de bonne qualité, il faut prendre en considération les facteurs suivants :

- o Transitions de la forme d'onde.
- o Taux d'échantillonnage.
- o Bon rendement.
- o Réduction du bruit.
- o Soin à éviter la distorsion.
- o Limitations du filtre passe bas.

REALISATION DES TRANSITIONS D'UNE FORME D'ONDE

Pour éviter des sons désagréables lorsqu'on passe d'une forme d'onde à une autre, il nous faut faire des transitions en douceur. On peut éviter les "petits bruits secs" en s'assurant que les formes d'onde commencent et se terminent approximativement à la même valeur. On peut éviter les "gros bruits secs" en faisant commencer une forme d'onde seulement au passage à zéro de la courbe. On peut éviter les "bruits lourds et sourds" en arrangeant l'amplitude moyenne de chaque onde pour qu'elle se trouve à la même valeur. L'amplitude moyenne est la somme des octets de la forme d'onde divisée par le nombre d'octets de la forme d'onde elle-même.

TAUX D'ECHANTILLONNAGE

Si on a besoin d'une grande précision de fréquence en sortie, il peut arriver de trouver que la fréquence qu'on veut produire se trouve quelque part entre deux taux d'échantillonnages disponibles, mais non suffisamment proche de l'un des deux taux pour qu'elle réponde à nos besoins. Dans ces cas, il se peut qu'on ait la nécessité de régler la longueur de la table de données audio outre à devoir altérer le taux d'échantillonnage.

Pour les très hautes fréquences, on pourrait aussi avoir besoin d'utiliser les tables de données audio qui contiennent plus d'un cycle complet de la forme d'onde pour reproduire la fréquence voulue de façon plus précise, comme l'illustre la Planche 5-4.

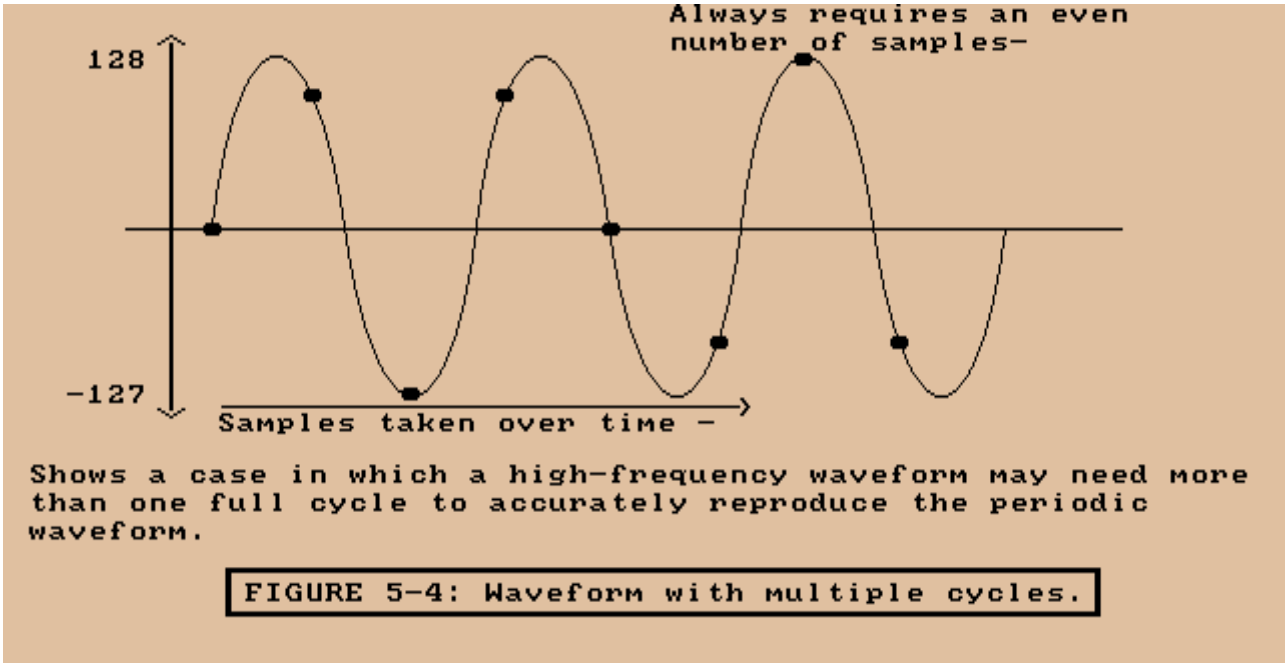


PLANCHE 5-4 : Forme d'Onde avec Plusieurs Cycles

RENDEMENT

La gestion des canaux audio implique une certaine quantité de travail en supplément. Si l'on essaie de produire une synthèse audio continue et sans heurts, on devra essayer d'éviter autant que possible le travail supplémentaire occasionné par le contrôle du système. En règle de base, plus le tampon audio qu'on fournit au système est important, moins souvent il sera nécessaire d'interrompre le système pour réinitialiser les pointeurs au début du tampon suivant et, par la même occasion, les interactions avec le système seront moindres par rapport à ce qu'elles devraient. S'il n'existe qu'un tampon pour la forme d'onde, le 'hardware' réinitialise les pointeurs automatiquement et, par conséquent, il n'y aura pas de travail de programmation supplémentaire à faire.

La section "Raccorder les Sons" illustrait comment on pouvait raccorder des "bouts" de sons ensemble en réagissant aux interruptions et en modifiant les valeurs des registres d'adresse pour abouter les sons. Si un système est chargé lourdement, il est possible qu'une réaction à une interruption n'arrive pas à temps pour garantir une transition audio sans heurts. Par conséquent, là où on aura besoin d'une sortie sans heurts, il sera opportun d'utiliser une table audio beaucoup plus grande. Cette table profite aussi de l'aptitude qu'offre le DMA audio qui réduit le nombre d'interruptions que doit prendre en charge le 68000.

REDUCTION DU BRUIT

Pour réduire le niveau de bruit et produire un son propre, il faut essayer d'utiliser toute l'étendue de -128 à 127, quand on représente une forme d'onde. Ceci réduit une grande partie du bruit (erreurs de quantification) qui s'ajoute au signal dès qu'on emploie un nombre plus important de bits de précision. Le bruit de quantification est provoqué par l'introduction de l'erreur d'arrondi. Si on essaie de reproduire un signal, tel qu'une onde sinusoïdale, on peut représenter l'amplitude de chaque échantillon avec seulement un nombre réduit de bits de précision. La différence entre le nombre véritable et l'approximation fournie représente l'erreur d'arrondi ou bruit.

En doublant l'amplitude, on aura la moitié moins de bruit car la grandeur des pas de la forme d'onde reste inchangée et que, par conséquent, celle-ci est une fraction plus petite de l'amplitude.

En d'autres termes, si on essaie de représenter une forme d'onde en utilisant, par exemple, une fourchette de +3 à -3 seulement, l'importance de l'erreur en sortie serait considérablement plus grande que si l'on utilisait une fourchette de 127 à -128 pour représenter le même signal. En proportion, la valeur numérique employée pour représenter l'amplitude d'une forme d'onde aura une erreur moins importante. Dès qu'on augmente le nombre de niveaux d'échantillons possibles, on diminue la grandeur relative de chaque pas et, par conséquent, on diminue l'ordre de grandeur de l'erreur.

Pour produire des sons de faible intensité, on doit continuer à définir une forme d'onde en utilisant la totalité de l'étendue, et on doit régler le volume. Ceci maintient le même niveau de précision (rapport signal/bruit) pour les sons de faible et forte intensité.

LA DISTORSION

Si on se sert de l'échantillonnage pour produire une forme d'onde, un effet de bord se produit quand le taux d'échantillonnage a des "battements" ou se combine avec la fréquence qu'on veut produire. Ceci produit deux fréquences supplémentaires, l'une égale au taux d'échantillonnage plus la fréquence voulue et l'autre égale au taux d'échantillonnage moins la fréquence voulue. Ce phénomène s'appelle distorsion.

On élimine la distorsion si le taux d'échantillonnage dépasse la fréquence de sortie d'au moins 7 Khz. Ceci met la fréquence de battement en dehors de l'intervalle du filtre passe-bas, en écrêtant les fréquences indésirables. La Planche 5-5 montre un tracé dans le domaine de fréquences d'un filtre passe-bas utilisé dans le système pour empêcher la distorsion.

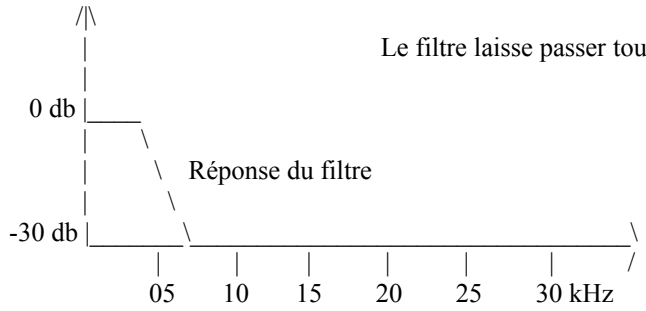


PLANCHE 5-5 : Tracé dans le Domaine de Fréquences du Filtre Passe-bas.

La Planche 5-6 montre qu'il est permis d'utiliser un taux d'échantillonnage de 12 KHz pour produire une forme d'onde de 4 KHz. Les deux fréquences de battement sont en dehors de l'étendue du filtre, comme indiqué par les calculs suivants :

12+4= 16 KHz
12-4= 8 KHz

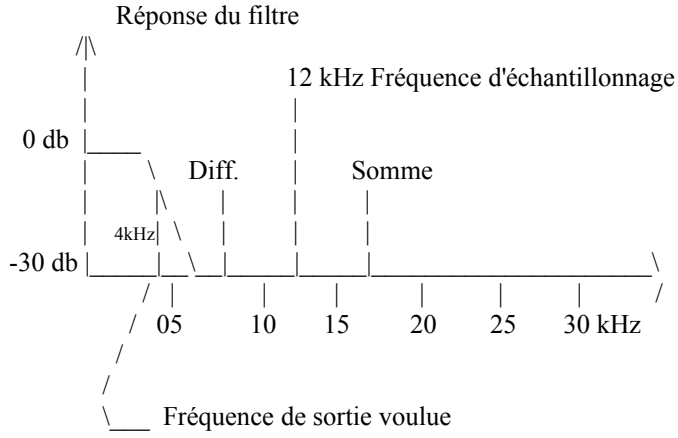


PLANCHE 5-6 : Sortie sans Bruit (Sans Distorsion)

On pourra s'apercevoir dans la Planche 5-7 qu'il est inacceptable d'utiliser un échantillonnage de 10 KHz pour produire une forme d'onde de 4KHz. L'une des fréquence de battement (10 - 4), se trouvant à l'intérieur de l'étendue du filtre, permet à une partie de la fréquence indésirable de se manifester dans la sortie audio.

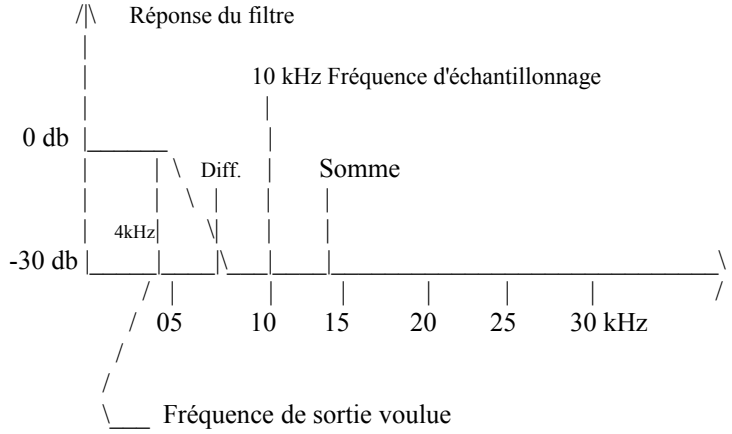


PLANCHE 5-7 : Distorsion

Tout ceci donne naissance à l'équation suivante qui indique que la fréquence d'échantillonnage doit dépasser la fréquence de sortie d'au moins 7 KHz pour que la fréquence de battement se trouve au-dessus de la plage de coupure du filtre qui évite la distorsion :

Taux minimal d'échantillonnage = Elément de fréquence plus élevée + 7 KHz

Dans l'équation, l'élément fréquence est défini en tant que "élément de fréquence plus élevée" car on peut produire des formes d'onde complexes avec de multiples éléments de fréquence, au lieu de produire une onde sinusoïdale simple.

FILTRE PASSE-BAS

Le système comprend un filtre qui élimine la distorsion telle que nous l'avons définie ci-dessus. Ce filtre devient actif autour de 4 HHz et commence à atténuer (couper) régulièrement le signal. D'ordinaire, il n'est pas possible d'entendre clairement des fréquences supérieures à 7 KHz. Par conséquent, on prendra la réponse en fréquence la plus complète dans la gamme de fréquences de 0 à 7 kHz. Si l'on est en train de créer des fréquences de 0 à 7 KHz, on devra sélectionner un taux d'échantillonnage qui ne soit pas inférieur à 14 KHz, ce qui correspond à une période d'échantillonnage dans la fourchette 124 à 256.

A une période d'échantillonnage de 320 environ, on commence à perdre les valeurs de fréquence plus élevées entre 0 KHz et 7 KHz, comme illustré par la Planche 5-6.

TABLEAU 5-6 : Relation entre Fréquence et Taux d'Échantillonnage

	Période d'échant.	**Taux d'échant.**	**Fréquence de Sortie Maximale (KHz)**
Taux d'échant. maxi.	124	29	7
Taux d'échant. mini. pour sortie 7 KHz	256	14	7
Taux d'échant. trop bas pour sortie 7 KHz	320	11	4

Dans les A2000 avec deux circuits de fond de panier par couche et dans les A500 plus récents, il existe un bit de contrôle qui permet à la sortie audio de court-circuiter le filtre passe-bas. Ce bit de contrôle est le même bit de sortie du CIA 8520 qui contrôle la luminosité du voyant rouge de mise en route. Le fait de court-circuiter le filtre permet d'avoir des sons de meilleure qualité dans certaines applications, mais alors, le cas échéant, on aura besoin d'un filtre externe avec une fréquence de coupure appropriée.

UTILISATION DIRECTE DE LA SORTIE AUDIO (SANS DMA)

Il est possible de créer des sons en écrivant un seul mot de données audio dans les adresses de sortie audio, au lieu de mettre en place en mémoire une liste de données audio. Cette méthode de contrôle de la sortie coûte très cher en temps de calcul au processeur et, par conséquent, n'est pas recommandée.

Pour utiliser une sortie audio directe, il ne faut pas valider le DMA du canal audio qu'on compte employer; ceci modifie la synchronisation des interruptions. L'interruption normale intervient après lecture d'une adresse de données; en sortie audio directe, l'interruption intervient après l'envoi d'un mot de données en sortie.

A la différence de ce qui se produit avec la sortie de données automatique contrôlée par les DMA, en sortie audio directe, si l'on n'écrit pas un nouveau jeu de données dans les adresses de sortie, avant que deux intervalles d'échantillonnage ne se soient écoulés, la sortie audio cesse de se modifier. La dernière valeur continuera à rester en sortie du convertisseur numérique/analogique.

On met en place de la même manière les registres de période et de volume.

LA GAMME MUSICALE TEMPEREE

Le Tableau 5-7 donne une approximation proche de la gamme tempérée sur une octave pour un échantillon d'une taille de 16 bits. La colonne "Période" donne la valeur de période à écrire dans les registres de période. Le registre de longueur AUDxLEN devra être mis à 8 (seize octes = 8 mots). L'échantillon devrait représenter un cycle de la forme d'onde.

TABLEAU 5-7 : Octave Tempérée d'un Échantillon de 16 Octets

Période NTSC	Période PAL	Note	Fréquence Idéale	Fréquence NTSC réelle	Fréquence PAL réelle
254	252	A	880.0	880.8	879.7
240	238	A#	932.3	932.2	931.4
226	224	B	987.8	989.9	989.6
214	212	C	1046.5	1045.4	1045.7
202	200	C#	1108.7	1107.5	1108.4
190	189	D	1174.7	1177.5	1172.9
180	178	D#	1244.5	1242.9	1245.4
170	168	E	1318.5	1316.0	1319.5
160	159	F	1396.9	1398.3	1394.2
151	150	F#	1480.0	1481.6	1477.9
143	141	G	1568.0	1564.5	1572.2
135	133	G#	1661.2	1657.2	1666.8

Le Tableau ci-dessus indique les valeurs de période à utiliser avec un échantillon de 16 bits pour faire des sons dans la deuxième octave au-dessus du C central. Pour générer des sons dans des octaves inférieures, on peut utiliser deux méthodes : doubler la valeur de la période ou doubler la taille de l'échantillon.

Quand on double la période, le temps entre chaque échantillon est doublé de telle sorte que l'échantillon aura besoin de deux fois plus de temps pour se faire entendre. Ce qui signifie que la fréquence du son généré est coupée en deux ce qui donne l'octave inférieure suivante. Ainsi, si vous jouez un C avec une valeur de période de 214, puis vous jouez ce même échantillon avec une valeur de période de 428, on entendra ce dernier dans l'octave inférieure suivante.

De la même manière, quand on double la taille de l'échantillon, celui-ci prend deux fois plus de temps à restituer la totalité de l'échantillon et la fréquence du son généré se situe dans l'octave inférieure suivante. Ainsi, si l'on a un échantillon de 8 et un autre de 16 bits de la même forme d'onde joués à la même vitesse, on entendra l'échantillon de 16 bits dans l'octave inférieure (plus grave).

L'échantillon d'une gamme tempérée représente typiquement le cycle entier d'une note. Pour éviter la distorsion, avec ces échantillons, on ne devrait utiliser que des valeurs de période se situant dans l'intervalle 124-256. Les périodes de 124 à 256 correspondent aux taux d'écoute dans l'intervalle 14-28 KHz échantillons/seconde qui fait un usage effectif du filtre de coupure à 7 KHz de l'Amiga dont la fonction est d'empêcher le bruit. Pour rester dans cette fourchette, on aura besoin d'un échantillon différent par octave.

S'il n'est pas possible d'utiliser un échantillon différent pour chaque octave, on devra régler la valeur de période dans l'intervalle total de 124 à 65536. Pour le programmeur, c'est facile à faire, mais cela peut produire un bruit indésirable dans les hautes fréquences du son résultant. Pour plus de renseignements à ce sujet, lire la section "La Distorsion".

Les valeurs du Tableau 5-7 ont été générées en employant la formule ci-dessous. Pour calculer le son généré avec une période et une taille d'échantillon données, utilisez :

$$\text{Fréquence} = \frac{\text{Constante d'Horloge}}{\text{Octets de l'Echant.} * \text{Période}} = \frac{3579545}{16 * \text{Période}} = 880.8 \text{ Hz}$$

Dans un système NTSC, la constante d'horloge est égale à 3579545 signaux par seconde. Dans un système PAL, cette constante est de 3546895 signaux. Les octets de l'échantillon sont le nombre d'octets d'un cycle de l'échantillon de la forme d'onde. (La constante d'horloge dérive de la valeur de l'horloge du système divisée par deux. Cette valeur varie si l'on utilise une horloge système externe, comme celle d'un 'genlock'.)

En utilisant la formule ci-dessus, on peut générer les valeurs nécessaires d'un échantillon arbitraire quelconque dans la gamme tempérée. Le Tableau 5-8 donne une approximation proche d'une gamme tempérée sur cinq octaves en employant cinq échantillons. On dérive les valeurs en utilisant la formule ci-dessus. Remarquez que, dans chaque octave, les valeurs de période sont identiques et que la taille de l'échantillon est de moitié. Les échantillons listés représentent une forme d'onde triangulaire simple.

TABLEAU 5-8 : Gamme Tempérée sur Cinq Octaves

Période NTSC	Période PAL	Note	Fréquence Idéale	Fréquence NTSC réelle	Fréquence PAL réelle
254	252	A	55.00	55.05	54.98
240	238	A#	58.27	58.26	58.21
226	224	B	61.73	61.87	61.85
214	212	C	65.40	65.34	65.35
202	200	C#	69.29	69.22	69.27
190	189	D	73.41	73.59	73.30
180	178	D#	77.78	77.68	77.83
170	168	E	82.40	82.25	82.47
160	159	F	87.30	87.39	87.13
151	150	F#	92.49	92.60	92.36
143	141	G	98.00	97.78	98.26
135	133	G#	103.82	103.57	104.17

Taille de l'échantillon = 256 octets, AUDxLEN = 128

254	252	A	110.00	110.10	109.96
240	238	A#	116.54	116.52	116.43
226	224	B	123.47	123.74	123.70
214	212	C	130.81	130.68	130.71
202	200	C#	138.59	138.44	138.55
190	189	D	146.83	147.18	146.61
180	178	D#	155.56	155.36	155.67
170	168	E	164.81	164.50	164.94
160	159	F	174.61	174.78	174.27
151	150	F#	184.99	185.20	184.73
143	141	G	196.00	195.56	196.52
135	133	G#	207.65	207.15	208.35

Taille de l'échantillon = 128 octets, AUDxLEN = 64

254	252	A	220.00	220.20	219.92
240	238	A#	233.08	233.04	232.86
226	224	B	246.94	247.48	247.41
214	212	C	261.63	261.36	261.42
202	200	C#	277.18	276.88	277.10
190	189	D	293.66	294.37	293.23
180	178	D#	311.13	310.72	311.35
170	168	E	329.63	329.00	329.88
160	159	F	349.23	349.56	348.55
151	150	F#	369.99	370.40	369.47
143	141	G	392.00	391.12	393.05
135	133	G#	415.30	414.30	416.70

Taille de l'échantillon = 64 octets, AUDxLEN = 32

Période NTSC	Période PAL	Note	Fréquence Idéale	Fréquence NTSC réelle	Fréquence PAL réelle
254	252	A	440.0	440.4	439.8
240	238	A#	466.16	466.09	465.72
226	224	B	493.88	494.96	494.82
214	212	C	523.25	522.71	522.83
202	200	C#	554.37	553.77	554.20
190	189	D	587.33	588.74	586.46
180	178	D#	622.25	621.45	622.70
170	168	E	659.26	658.00	659.76
160	159	F	698.46	699.13	697.11
151	150	F#	739.99	740.80	738.94
143	141	G	783.99	782.24	786.10
135	133	G#	830.61	828.60	833.39

Taille de l'échantillon = 32 octets, AUDxLEN = 16

254	252	A	880.0	880.8	879.7
240	238	A#	932.3	932.2	931.4
226	224	B	987.8	989.9	989.6
214	212	C	1046.5	1045.4	1045.7
202	200	C#	1108.7	1107.5	1108.4
190	189	D	1174.7	1177.5	1172.9
180	178	D#	1244.5	1242.9	1245.4
170	168	E	1318.5	1316.0	1319.5
160	159	F	1396.9	1398.3	1394.2
151	150	F#	1480.0	1481.6	1477.9
143	141	G	1568.0	1564.5	1572.2
135	133	G#	661.2	1657.2	1666.8

Taille de l'échantillon = 16 octets, AUDxLEN = 8

ECHANTILLON DE 256 OCTETS

```
  0   2   4   6   8  10  12  14  16  18  20  22  24  26
 28  30  32  34  36  38  40  42  44  46  48  50  52  54
 56  58  60  62  64  66  68  70  72  74  76  78  80  82
 84  86  88  90  92  94  96  98 100 102 104 106 108 110
112 114 116 118 120 122 124 126 128 126 124 122 120 118
116 114 112 110 108 106 104 102 100  98  96  94  92  90
 88  86  84  82  80  78  76  74  72  70  68  66  64  62
 60  58  56  54  52  50  48  46  44  42  40  38  36  34
 32  30  28  26  24  22  20  18  16  14  12  10   8   6
  4   2   0  -2  -4  -6  -8 -10 -12 -14 -16 -18 -20 -22
-24 -26 -28 -30 -32 -34 -36 -38 -40 -42 -44 -46 -48 -50
-52 -54 -56 -58 -60 -62 -64 -66 -68 -70 -72 -74 -76 -78
-80 -82 -84 -86 -88 -90 -92 -94 -96 -98 -100 -102 -104 -106
-108 -110 -112 -114 -116 -118 -120 -122 -124 -126 -127 -126 -124 -122
-120 -118 -116 -114 -112 -110 -108 -106 -104 -102 -100  -98  -96  -94
 -92  -90  -88  -86  -84  -82  -80  -78  -76  -74  -72  -70  -68  -66
 -64  -62  -60  -58  -56  -54  -52  -50  -48  -46  -44  -42  -40  -38
 -36  -34  -32  -30  -28  -26  -24  -22  -20  -18  -16  -14  -12  -10
  -8   -6   -4   -2
```

ECHANTILLON DE 128 OCTETS

```
  0   4   8  12  16  20  24  28  32  36  40  44  48  52
 56  60  64  68  72  76  80  84  88  92  96 100 104 108
112 116 120 124 128 124 120 116 112 108 104 100  96  92
 88  84  80  76  72  68  64  60  56  52  48  44  40  36
 32  28  24  20  16  12   8   4   0   4   8  12  16  20
 24  28  32  36  40  44  48  52  56  60  64  68  72  76
 80  84  88  92  96 100 104 108 112 116 120 124 -127 -124
-120 -116 -112 -108 -104 -100  -96  -92  -88  -84  -80  -76  -72  -68
 -64  -60  -56  -52  -48  -44  -40  -36  -32  -28  -24  -20  -16  -12
  -8   -4
```

ECHANTILLON DE 64 OCTETS

```
  0   8  16  24  32  40  48  56  64  72  80  88  96 104
112 120 128 120 112 104  96  88  80  72  64  56  48  40
 32  24  16   8   0  -8 -16 -24 -32 -40 -48 -56 -64 -72
-80 -88 -96 -104 -112 -120 -127 -120 -112 -104  -96  -88  -80  -72
-64 -56 -48  40 -32 -24 -16  -8
```

ECHANTILLON DE 32 OCTETS

```
  0  16  32  48  64  80  96 112 128 112  96  80  64  48
 32  16   0 -16 -32 -48 -64 -80 -96 -112 -127 -112  -96  -80
-64   4   8 -32 -16
```

ECHANTILLON DE 16 OCTETS

```
  0  32  64  96 128  96  64  32   0 -32 -64 -96 -127  -96
-64 -32
```

VALEURS EN DECIBELS DES INTERVALLES DE VOLUME

Le Tableau 5-9 fournit les valeurs correspondantes en décibels pour les intervalles de volume du système Amiga.

TABLEAU 5-9 : Valeurs en Décibels des Intervalles de Volume

Volume	Valeur en Décibels	Volume	Valeur en Décibels
64	0.0	32	-6.0
63	-0.1	31	-6.3
62	-0.3	30	-6.6
61	-0.4	29	-6.9
60	-0.6	28	-7.2
59	-0.7	27	-7.5
58	-0.9	26	-7.8
57	-1.0	25	-8.2
56	-1.2	24	-8.5
55	-1.3	23	-8.9
54	-1.5	22	-9.3
53	-1.6	21	-9.7
52	-1.8	20	-10.1
51	-2.0	19	-10.5
50	-2.1	18	-11.0
49	-2.3	17	-11.5
48	-2.5	16	-12.0
47	-2.7	15	-12.6
46	-2.9	14	-13.2
45	-3.1	13	-13.8
44	-3.3	12	-14.5
43	-3.5	11	-15.3
42	-3.7	10	-16.1
41	-3.9	9	-17.0
40	4.1	8	-18.1
39	4.3	7	-19.2
38	4.5	6	-20.6
37	4.8	5	-22.1
36	-5.0	4	-24.1
35	-5.2	3	-26.6
34	-5.5	2	-30.1
33	-5.8	1	-36.1
		0	Moins infini

LES ETATS AUDIO MACHINE

Pour une explication des différents états, référez-vous à la Planche 5-8.
Il existe un état audio machine pour chaque état. La machine possède huit états et est synchronisée sur la cadence de la constante d'horloge (3,58 MHz en NTSC). A la base, trois états sont inutilisés et se limitent à retourner à l'état inactif (000). L'un des chemins en dehors de l'état d'attente est prévu pour les opérations pilotées par interruption (le processeur fournit les données), tandis que l'autre chemin est prévu pour les opérations pilotées par DMA (la puce spécialisée "Agnus" fournit les données).

Dans une opération pilotée par interruption, un transfert à la boucle principale (états 010 et 011) intervient immédiatement après écriture des données par le processeur. Dans l'état 010, l'octet supérieur est envoyé en sortie, ainsi que l'octet inférieur dans l'état 011. Des transitions comme 010011010 interviennent lorsque le compteur de période revient à un.
Le compteur de période est rechargé, lors de ces transitions. Tant qu'une interruption est remise à zéro par le processeur à temps, la machine reste dans la boucle principale. Autrement, elle entre en état d'attente.
Des interruptions sont générées à chaque transition de mot (011010).

Dans une opération pilotée par DMA, se produit une transition à l'état 001 et les demandes DMA sont envoyées à Agnus dès que le DMA est mis en service. A cause du transfert dans et de Agnus ['pipelining'], on devra laisser passer le premier mot de données. Dès que ce mot arrive, on passe à l'état 101; une demande pour le prochain mot de données est déjà partie.
Quand les données arrivent, on passe à l'état 010 et la boucle principale continuera tant qu'on ne mettra pas hors service le DMA. Le compteur de longueur fait un compte à rebours une fois à chaque arrivée d'un mot. Quand il a fini, une demande de redémarrage du DMA arrive à Agnus en même temps que la demande normale. Ceci informe Agnus qu'il doit réinitialiser le pointeur au début de la table de données. En outre, le compteur de longueur est rechargé et une demande d'interruption arrive dès que le compteur de longueur a fini son travail (compteur à 1). La demande arrive juste dès que le dernier mot de la forme d'onde commence sa sortie.

Les demandes du DMA, ainsi que ses demandes de redémarrage sont envoyées à Agnus une fois par ligne horizontale et les données sont renvoyées 14 cycles d'horloge plus tard environ (la durée d'un cycle d'horloge est de 200 nano-secondes).

En mode 'attach', les choses sont un peu différentes. Pour ce qui est du volume, en mode 'attach', le demandes interviennent comme elles le font pendant une opération normale (lors de la transition 011010). Pour ce qui est de la période, en mode 'attach', une suite de demandes intervient lors de la transition 010011. Lorsque la période et le volume 'attach' sont prioritaires, les demandes interviennent lors des deux transitions à la fois.

Si on emploie un taux d'échantillonnage plus élevé que celui employé normalement (approximativement 29 KHz), les deux échantillons du registre tampon seront répétés. Si on court-circuite le filtre de l'Amiga et que le volume est au maximum ($40), on pourra utiliser cette particularité pour réaliser des ondes porteuses jusqu'à 1,79 MHz. La modulation est placée dans la mappe de mémoire, avec des valeurs de signe positif pour les octets pairs et des valeurs de signe négatif pour les octets impairs.

Les symboles utilisés dans le diagramme d'état sont expliqués dans la liste qui suit. Les mots en majuscules indiquent les signaux externes; les noms en minuscules indiquent les signaux locaux.

AUDxON	Le "x" des DMA 'ON' indique le numéro du canal (signal de DMACON).
AUDxIP	Interruption audio en attente (une entrée dans le canal des circuits d'interruption).
AUDxIR	Demande d'interruption audio (une sortie en provenance du canal des circuits d'interruption).
intreq1	Demande d'interruption qui se combine avec intreq2 pour former AUDxIR.
intreq2	Se préparer pour une demande d'interruption. La demande arrivera après la prochaine transition 011 --> 010 dans une opération normale.
AUDxDAT	Signal de chargement des données audio. Charge 16 bits de données dans le canal audio.
AUDxDR	Demande du DMA audio à Agnus pour un mot de données.
AUDxDSR	Demande du DMA audio à Agnus de réinitialiser le pointeur au début du bloc.
dmasen	Valide la demande de redémarrage.
percntrld	Recharge le compteur de période à partir du circuit de mémorisation de sauvegarde, écrit typiquement par le processeur avec AUDxPER (on peut y écrire aussi par le biais d' 'attach')
percount	Faire un compte à rebours dans le compteur de période à un blocage.
perfin	Stopper le compteur de période (valeur = 1).
lencntrld	Recharger le compteur de longueur à partir du circuit de mémorisation de sauvegarde.
lencount	Faire un compte à rebours dans le compteur lors d'une encoche.
lenfin	Arrêt du compteur de longueur (valeur = 1).
volcntrld	Recharger le compteur de volume à partir du circuit de mémorisation de sauvegarde.
pbufld1	Charger le tampon de sortie à partir du circuit de mémorisation qui le contient, circuit écrit par AUDxDAT.
pbufld2	Comme pbufld1, mais seulement pendant 010-->011 avec période en mode 'attach'.
AUDxAV	Volume en mode 'attach'. Envoie les données au circuit de mémorisation de volume du canal suivant au lieu de les envoyer au convertisseur N/A.
AUDxAP	Période en mode 'attach'. Envoie les données au circuit de mémorisation de période du canal suivant au lieu de les envoyer au convertisseur N/A.
penhi	Valider les 8 bits de données de poids fort à envoyer au convertisseur N-->A.

napnav /AUDxAV * /AUDxAP + AUDxAV - aucune partie 'attach' ni même volume en mode 'attach'. Condition à satisfaire pour les demandes d'interruption normales du DMA.

sq2,1,0 Le nom des bascules d'état, du MSB au LSB.

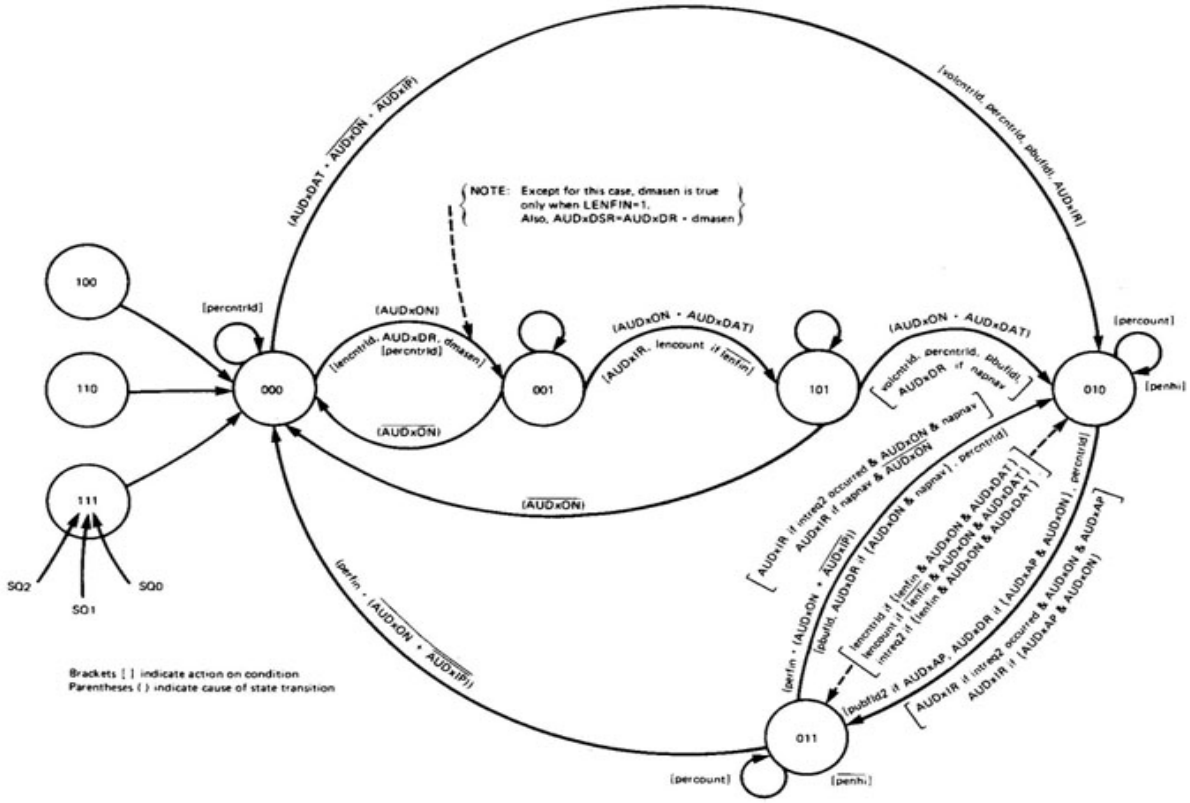


PLANCHE 5-8 : Diagramme des États Audio

Chapitre 6

LE 'HARDWARE' DU BLITTER

INTRODUCTION

Le 'blitter' est l'un des deux co-processeurs de l'Amiga. Faisant partie de la puce Agnus, il est employé pour copier quelque part des blocs de mémoire rectangulaires et pour dessiner des lignes. Quand il copie des blocs de mémoire, il est approximativement deux fois plus rapide que le 68000 et est capable de déplacer presque quatre méga-octets par seconde. Il peut tracer des lignes à une vitesse pouvant aller jusqu'à un million de pixels par seconde.

En mode déplacement de blocs, le 'blitter' a la possibilité d'exécuter toutes les opérations logiques sur trois zones de mémoire source, il a la possibilité de faire des décalages de 1 à 15 bits dans deux zones sources, a la possibilité d'effectuer des remplissages de surfaces fermées et celle de masquer le premier et le dernier mot de chaque ligne de trame. En mode ligne, une ligne peut avoir un modèle quelconque ou il est possible de tracer des lignes n'ayant qu'un pixel par ligne horizontale.

Le 'blitter' ne peut avoir accès qu'à la mémoire 'CHIP' - cette partie de mémoire à laquelle a accès le 'hardware' d'affichage. Tenter d'employer le 'blitter' pour lire ou écrire en mémoire 'FAST' ou dans un autre type de mémoire autre que la mémoire 'CHIP' peut provoquer la destruction de ce qui se trouve en mémoire CHIP.

On appelle "blit" chaque opération du 'blitter' - qu'il s'agisse d'un tracé de ligne ou d'un déplacement de blocs de mémoire. On exécute un 'blit' en initialisant les registres du 'blitter' avec des valeurs appropriées et en faisant ensuite démarrer celui-ci en écrivant dans le registre BLTSIZE. Comme le 'blitter' est un co-processeur asynchrone, le 68000 continue à fonctionner pendant l'exécution du 'blit'.

ORGANISATION DE LA MEMOIRE

Le 'blitter' travaille sur des mots, et non sur des bits. Toutes les données prélevées, modifiées et écrites sont des mots de 16 bits. Le 'blitter', par une programmation soignée, a la possibilité de réaliser plusieurs opérations de type "bit".

Le 'blitter' est particulièrement bien adapté aux opérations graphiques. A titre d'exemple, un écran de 320 sur 200, mis en place pour afficher 16 couleurs, est organisé en quatre plans de bits de 8000 octets chacun. Chaque plan de bits est constitué de 200 lignes de 40 octets ou de 20 mots de 16 bits. (A partir de maintenant, un "mot" voudra dire un mot de 16 bits.)

LES CANAUX DMA

Le 'blitter' a quatre canaux DMA - trois canaux source, étiquetés A, B et C, et un canal cible, qui s'appelle D. Chacun de ces canaux a un pointeur d'adresse indépendant, des registres de données et de modulo et, pour finir, un registre de masque de mot. Les quatre canaux se partagent un seul registre de taille pour le 'blit'.

Chaque registre de pointeur d'adresse est composé de deux mots, BLTxPTH et BLTxPTL. (Ici et par la suite, en se référant à un registre, le "x" présent dans le nom devra être remplacé par l'étiquette qui indique le canal, A, B, C ou D.) Les deux mots de chaque registre sont adjacents dans l'espace adressable du 68000, avec en premier le mot d'adresse de poids fort, ainsi on peut les écrire à partir du processeur avec une seule opération d'écriture 32 bits. On devrait écrire dans les registres de pointeur avec une adresse en octets. Comme le 'blitter' ne travaille que sur des mots, le bit d'adresse le moins significatif sera ignoré. Comme on ne peut avoir accès qu'à de la mémoire de type 'CHIP', certains bits plus significatifs seront ignorés à leur tour. Sur des machines avec 512 Ko de mémoire 'CHIP', les 13 bits les plus significatifs sont ignorés. Les machines futures auront plus de mémoire 'CHIP' et, alors, beaucoup moins de bits seront ignorés. Dans ces registres, on devra toujours écrire une adresse valide, paire, se situant en mémoire 'CHIP'.

NOTE

Assurez-vous d'avoir mis à zéro tous les bits inutilisés des registres des puces spécialisées. Ces bits pourront être utilisés dans des versions ultérieures de ces puces. Le fait de ne pas mettre à zéro ces bits pourra donner des résultats inattendus sur les machines à venir.

Chaque canal DMA peut être validé ou invalidé séparément. Les bits de validation sont les bits SRCA, SRCB, SRCC et DEST du registre de contrôle 0 (BLTCON0).

Une fois le canal invalidé, le système n'exécutera aucun cycle mémoire pour ce même canal et, pour un canal source, la valeur constante stockée dans le registre de données de ce canal, sera utilisée à chaque cycle du 'blitter'. Dans ce but, chacun des trois canaux source possède des registres de données pré-chargeables, BLTxDAT.

Les images en mémoire, sont stockés d'ordinaire de façon linéaire; chaque mot de données d'une ligne est placé à une adresse supérieure par rapport au mot se trouvant à sa gauche. Chaque ligne est donc la continuation "plus un" de la ligne précédente (cf. Planche 6-1).

```
20  21  22  23  24  24  26
27  28  29  30  31  32  33
34  35  36  37  38  39  40
41  42  43  44  45  46  47
48  49  50  51  52  53  54
55  56  57  58  59  60  61
```

Planche 6-1 : De quelle Façon sont Stockes les Images en Mémoire

La mappe de la Planche 6-1 représente un seul plan de bits (un seul bit de couleur) d'un dessin situé aux adresses de mots de 20 à 61. Chacune de ces adresses a accès à un seul mot (16 pixels) d'un seul plan de bits. Si ce dessin avait demandé seize couleurs, il aurait fallu quatre plans de bits comme celui-ci en mémoire et il aurait fallu quatre opérations de copie (transfert) pour transférer tout le dessin.

Le 'blitter' est très efficace quand il copie de tels blocs car il suffit de lui fournir l'adresse initiale (20), l'adresse cible et la taille du bloc (hauteur = 6, largeur = 7). Ensuite, celui-ci transfère les données, un mot à la fois, dès que le bus de données est disponible. Dès que le transfert est complet, le 'blitter' le signale au processeur par le biais d'un drapeau et d'une interruption.

NOTE

Cette opération de copie (transfert) se fait en mémoire et peut modifier ou ne pas modifier, lors du transfert, la mémoire qu'utilise l'affichage.

Tous les 'blits' de copie de données sont exécutés comme des rectangles de mots d'une hauteur et d'une largeur données. Tous les quatre canaux DMA utilisent un seul registre de taille 'blit', BLTSIZE, utilisé soit pour la largeur soit pour la hauteur. La largeur peut prendre des valeurs de 1 à 64 mots (de 16 à 1024 bits). La hauteur peut varier de 1 à 1024 lignes. La largeur est stockée dans les six bits les moins significatifs du registre BLTSIZE. Si l'on stocke un valeur égale à 0, le système utilise une largeur égale à 64 mots. C'est le seul paramètre du 'blitter' qu'on doit donner en mots.

La hauteur est stockée dans les dix bits supérieurs du registre BLTSIZE, et zéro représentera une hauteur de 1024 lignes.
Ainsi, le plus grand 'blit' possible avec le 'blitter' actuel de l'Amiga est de 1024 par 1024 pixels. Cependant, les opérations de masquage et celles de décalage, peuvent exiger qu'on prélève un mot supplémentaire à chaque ligne de balayage de trame, ce qui donne une largeur horizontale maximale utile de 1008 pixels.

NOTE

Pour insister sur le paragraphe précédent : la largeur d'un 'blit' se donne en mots et un zéro représente 64 mots. La hauteur d'un 'blit' se donne en lignes et un zéro représente 1024 lignes.

Le 'blitter', en outre, offre des facilités, les modulo, pour avoir accès à des dessins de dimensions inférieures à celles d'un plan de bits pris dans son ensemble. Chaque canal DMA possède un registre modulo 16 bits, qui s'appelle BLTxMOD. A mesure que chaque mot est prélevé (ou écrit) dans un canal validé, le registre de pointeur d'adresse est augmenté de deux pas (deux octets, ou un seul mot). Après chaque ligne, le 'blit' est terminé, la valeur modulo signée 16 bits de ce canal DMA vient s'ajouter au pointeur d'adresse. (Une ligne est définie par sa largeur, stockée dans BLTSIZE.)

NOTE

Les valeurs modulo s'expriment en octets et non en mots. Etant donné que le 'blitter' a seulement la possibilité de travailler avec des mots, le bit le moins significatif est ignoré. La valeur est étendue avec son signe à toute la largeur des registres de pointeur d'adresse. Les modulo de valeur négative peuvent s'avérer utiles de plusieurs façons, telle la répétition d'une ligne si l'on met le modulo à la valeur négative de la largeur d'un plan de bits.

A titre d'exemple, supposons que l'on veuille travailler sur une portion de mappe de bits de 320 sur 200 pixels qui a commencé à la ligne 13, octet 12 (où l'un et l'autre partent de zéro) et que la portion ait une largeur de 10 octets. Nous devrions initialiser le registre de pointeur à l'adresse du plan de bits, plus 40 octets par ligne par 13 lignes, plus 12 octets pour avoir la position horizontale correcte. Nous devrions mettre la largeur à 5 mots (10 octets). A la fin de chaque ligne, nous devrions sauter 30 octets pour avoir le début de la ligne suivante, nous devrions, par conséquent, employer une valeur modulo de 30. D'une façon générale, la largeur (en mots) par deux, plus la valeur modulo (en octets) devrait être égale à toute la largeur, en octets, du plan de bits qui contient l'image.

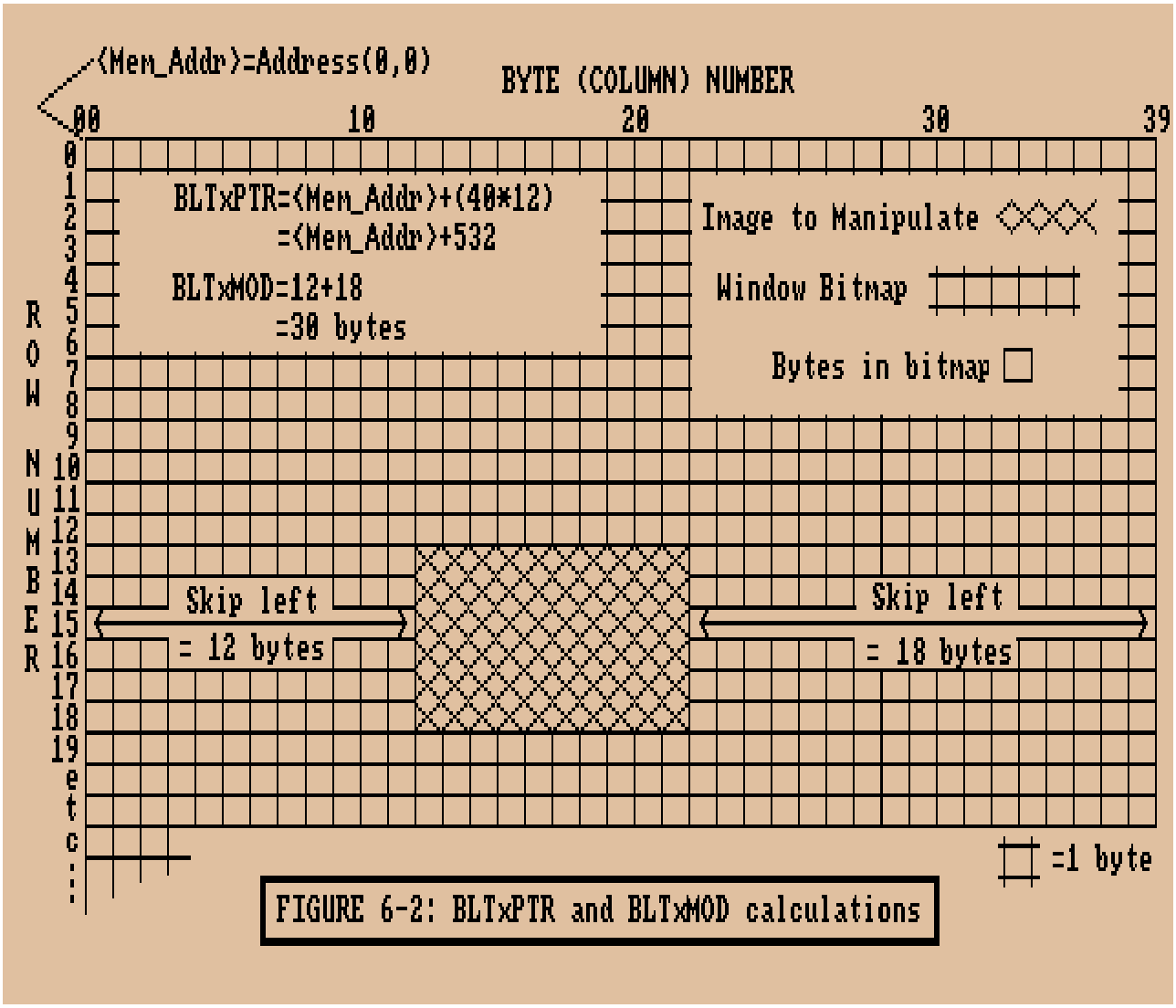


PLANCHE 6-2 : Calculs pour BLTxPTR et BLTxMOD

NOTE

On peut employer le 'blitter' pour traiter des parties linéaires plutôt que des parties rectangulaires, en mettant la valeur horizontale ou verticale de BLTSIZE à 1.

Comme chaque canal DMA a son propre registre de modulo, on a la possibilité de transférer des données dans des plans de bits de différentes largeurs. Ceci est très utile, quand on déplace des petits dessins dans des plans de bits d'écran de dimensions plus grandes.

GENERATEUR DE FONCTIONS

Le 'blitter' a la possibilité de réunir des données à partir de trois canaux DMA source de 256 manières différentes, pour générer les valeurs stockées par le canal DMA cible. Ces sources pourraient être un plan de bits provenant de trois images graphiques indépendantes. Tant que chacune de ces sources sera une surface rectangulaire composée d'une multitude de points, on pourra appliquer une même opération logique à chaque point de toute la surface. Ainsi, pour pouvoir définir une opération logique du 'blitter', il est seulement indispensable de prendre en considération ce qui se produirait avec toutes les liaisons possibles d'un bit avec celui de chacune des trois sources.

Avec trois bits, il y a huit combinaisons de valeurs possibles, pour chacune desquelles il nous faudra indiquer le bit cible correspondant sous forme de zéro ou de un. On peut visualiser ceci avec une table de vérité ordinaire, comme nous l'indiquons ci-dessous. Nous avons listé les trois canaux source et les valeurs possibles d'un seul bit à partir de chacun des trois autres bits.

A	B	C	D	Position	BLTCON0 MINTERM
0	0	0	?	0	$\overline{A}\overline{B}\overline{C}$
0	0	1	?	1	$\overline{A}\overline{B}C$
0	1	0	?	2	$\overline{A}B\overline{C}$
0	1	1	?	3	$\overline{A}BC$
1	0	0	?	4	$A\overline{B}\overline{C}$
1	0	1	?	5	$A\overline{B}C$
1	1	0	?	6	$AB\overline{C}$
1	1	1	?	7	ABC

Cette information est rassemblée dans un format standard, l'octet de contrôle LF du registre BLTCON0. Cet octet programme le 'blitter' pour que celui-ci exécute l'une des 256 opérations logiques possibles sur les trois sources pour un 'blit' donné.

Pour calculer l'octet de contrôle LF de BLTCON0, on remplit la table de vérité avec les valeurs voulues pour D et on lit la valeur de fonction de bas en haut dans la table.

Par exemple, si nous voulions mettre tous les bits de la cible là où le bit source A correspondant est 1 ou le bit source B correspondant est 1, nous devrions remplir les quatre dernières entrées de la table de vérité avec des 1 (car le bit de A est mis à 1) et la troisième, quatrième, septième et huitième entrée avec des 1 (car le bit de B est mis à 1) et toutes les autres (la première et la deuxième) avec des 0, car ni A ni B ne sont mis à 1. Ensuite, nous lirions la table de vérité de bas en haut, et trouverons 11111100, soit $FC [1].

[1] "$" indique la notation hexadécimale.

Comme autre exemple, un octet de contrôle de $80 (= 1000 0000 en binaire) active seulement les bits des points du rectangle cible où les bits correspondant des sources A, B et C sont tous activés (ABC = 1, bit 7 de LF activé). Tous les autres points du rectangle correspondant aux autres liaisons pour A, B et C seront des 0. Ceci se produit parce que les bits de 6 à 0 de l'octet de contrôle LF, qui indiquent la sortie D dans ces situations, sont mis à 0.

ETUDE DE L'OCTET DE CONTROLE LF AVEC LES 'MINTERMS'

Une approche pour étudier l'octet de contrôle LF utilise les équations logiques. Chaque ligne de la table de vérité correspond à un 'minterm' qu'on peut définir comme une assignation particulière de valeurs aux bits A, B et C. Par exemple, habituellement, on écrit le premier 'minterm' $\overline{A}\overline{B}\overline{C}$ ou "not A and not B and not C". Le dernier s'écrit ABC.

NOTE

Les deux termes adjacents sont soumis à un AND et les deux termes qui sont séparés par un "+" sont soumis à un OR. "AND" a une priorité plus élevée, par conséquent, AB + BC est égal à (AB) + (BC).

On peut écrire n'importe quelle fonction sous forme de somme de 'minterms'. Si nous voulions calculer la fonction où D est à 1 quand le bit de A est mis à 1 et que le bit de C est nul ou quand le bit de B est mis à 1, nous avons la possibilité de l'écrire sous le forme $A\overline{C} + B$ ou "A and not C or B". Etant donné que "1 and A" est égal à "A" :

$D = A\overline{C} + B$

$D = A(1)\overline{C} + (1)B(1)$

Etant donné que A et $\overline{A}$ sont vrais tous les deux (1 = A + $\overline{A}$), et de même pour B et C; nous pouvons développer l'équation ci-dessus comme ceci :

$D = A(1)\overline{C} + (1)B(1)$

$D=A(B+\overline{B})\overline{C}+(A+\overline{A})B(C+\overline{C})$

$D=AB\overline{C}+A\overline{B}\overline{C}+AB(C+\overline{C})+\overline{A}B(C+\overline{C})$

$D=AB\overline{C}+A\overline{B}\overline{C}+ABC+AB\overline{C}+\overline{A}BC+\overline{A}B\overline{C}$

Après avoir procédé à quelques simplifications : nous nous retrouvons avec les cinq 'minterms' suivants :

$A\overline{C}+B=AB\overline{C}+A\overline{B}\overline{C}+ABC+\overline{A}BC+\overline{A}B\overline{C}$

Ceux-ci correspondent aux positions 6, 4, 7, 3 et 2 du bit de BLTCON0, en accord avec notre table de vérité, que nous mettrons alors en place, tandis que nous remettrons à zéro le reste.

Le large choix d'opérations logiques autorise des techniques graphiques sophistiquées. Par exemple, on a la possibilité de déplacer le dessin d'une voiture au milieu de quelques immeubles pré-existants avec peu de 'blits'. La création de cet effet demande des dessins de voiture, des immeubles (ou un arrière-plan) et un "masque" voiture

qui contienne des bits mis à 1, là où le dessin de la voiture n'est pas transparent. On peut visualiser ce masque comme l'ombre de la voiture à partir d'une source de lumière qui se trouve dans la même position que le spectateur.

NOTE

Le masque de la voiture n'a besoin que d'un seul plan de bits, et ceci sans aucun rapport avec la profondeur du plan de bits de l'arrière-plan. On peut utiliser ce masque pour chacun des plans de bits de l'arrière-plan, à tour de rôle.

Pour animer la voiture, sauvegardez d'abord le dessin d'arrière-plan où l'on devra placer la voiture. Copiez, ensuite, la voiture dans son premier emplacement avec un autre 'blit'. Maintenait, l'image est prête pour l'affichage. Pour créer l'image suivante, sauvegardez la prochaine section d'arrière-plan où se trouvera la voiture et redessinez la voiture, en vous servant de trois 'blits' différents. (Cette technique fonctionne mieux avec des 'blits' synchronisés sur le faisceau électronique ou en créant un double tampon.)

Pour sauvegarder l'arrière-plan de manière temporaire, copiez un rectangle de celui-ci (à partir du canal A, par exemple) dans un tampon de sauvegarde (en utilisant le canal D). Dans ce cas, la fonction qu'on devra utiliser est "A", la fonction standard de copie. Dans le Tableau 6-1, nous remarquons que le code LF correspondant a une valeur de $F0.

Pour dessiner la voiture, nous pourrions utiliser le canal DMA de A pour prélever les véritables données de la voiture, le canal DMA de C pour prélever l'arrière-plan et le canal de D pour écrire la nouvelle image.

NOTE

Nous devons prélever l'arrière-plan cible avant d'y écrire car seule une partie du mot cible pourrait avoir besoin de modifications et il n'existe aucune possibilité d'écrire seulement dans une partie du mot.

Lors du 'blit' de la voiture dans l'arrière-plan, nous pourrions avoir envie d'utiliser une fonction qui, là où le masque de la voiture (prélevé avec le canal DMA de A) a un bit mis à 1, nous ferait passer à travers les données de la voiture, à partir de B, et là où A n'a pas de bit mis à 1, nous ferait passer à travers l'arrière-plan d'origine, à partir de C. La fonction correspondante, à laquelle on fait souvent référence comme à la fonction

"cookie-cut" [découpe pépère], est $AB+\overline{A}C$, qui fonctionne avec une valeur de code LF égale à $CA.

Pour restaurer l'arrière-plan et le préparer pour la prochaine séquence, nous copierons l'information sauvegardée ci-dessus lors de la première étape, avec la fonction de copie standard ($F0).

Si on décale les données et le masque à un nouvel emplacement et que l'on répète les trois étapes ci-dessus à plusieurs reprises, la voiture semblera rouler à travers l'arrière-plan (les immeubles).

NOTE

Il se peut que celle-ci ne soit pas la meilleure méthode d'animation, en relation avec l'application, mais nous emploierons souvent la fonction 'cookie-cut'.

Le Tableau 6-1 liste quelques unes des fonctions les plus courantes, ainsi que leur valeur, pour une référence simple.

Tableau 6-1 : Tableau des Valeurs Minterms Courantes

Equation Sélectionnée	**Code LF de BLTCON0**	**Equation Sélectionnée**	**Code LF de BLTCON0**
D = A	$F0	D = AB	$C0
D = $\overline{A}$	$0F	D = $\overline{AB}$	$30
D = B	$CC	D = $\overline{A}$B	$0C
D = $\overline{B}$	$33	D = $\overline{AB}$	$03
D = C	$AA	D = BC	$88
D = $\overline{C}$	$55	D = B$\overline{C}$	$44
D = AC	$A0	D = $\overline{B}$C	$22
D = A$\overline{C}$	$50	D = $\overline{AC}$	$11
D = $\overline{A}$C	$0A	D = A+$\overline{B}$	$F3
D = $\overline{AC}$	$05	D = $\overline{A}$+$\overline{B}$	$3F
D = A+B	$FC	D = A+$\overline{C}$	$F5
D = $\overline{A}$+B	$CF	D = $\overline{A}$+$\overline{C}$	$5F
D = A+C	$FA	D = B+$\overline{C}$	$DD
D = $\overline{A}$+C	$AF	D = $\overline{B}$+$\overline{C}$	$77
D = B+C	$EE	D = AB+$\overline{A}$C	$CA
D = $\overline{B}$+C	$BB		

ETUDE DE L'OCTET DE CONTROLE LF AVEC LES DIAGRAMMES DE VENN

Une autre manière d'arriver à une fonction particulière est l'utilisation
des diagrammes de Venn :

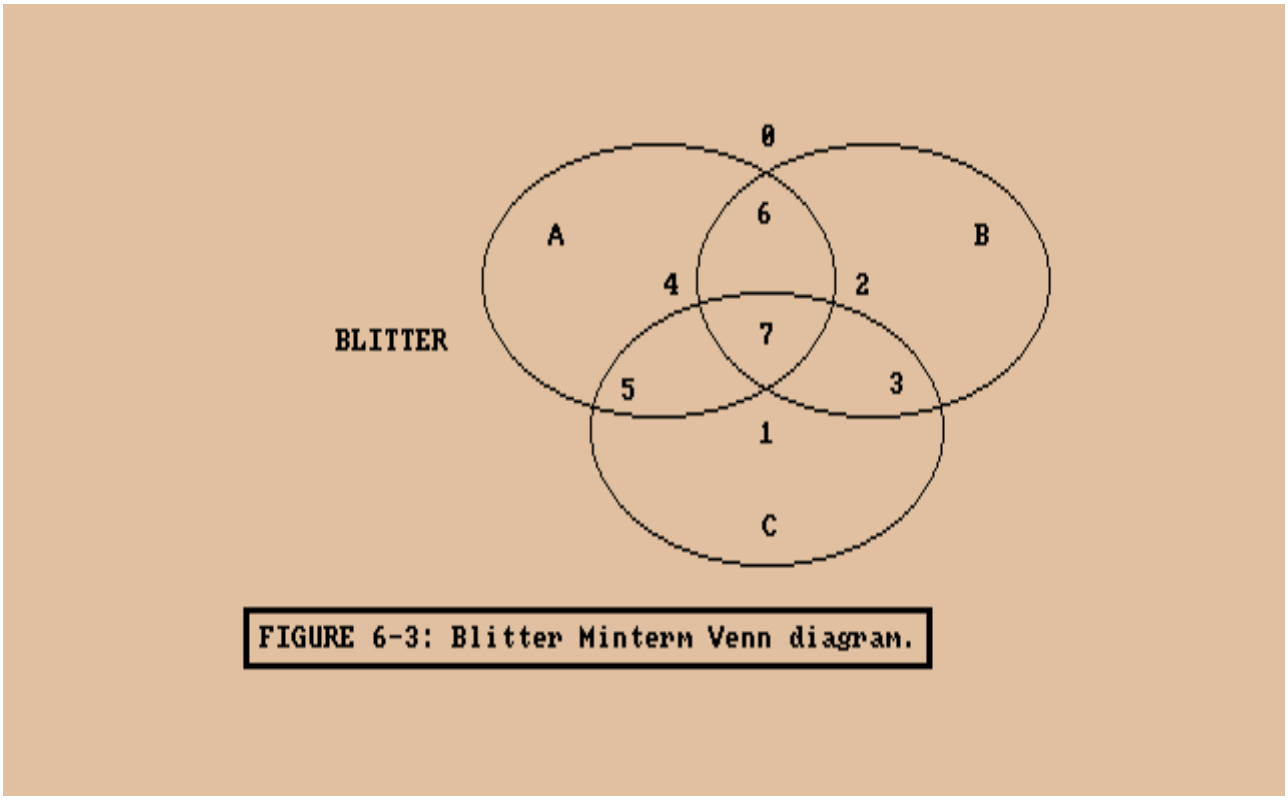


FIGURE 6-3: Blitter Minterm Venn diagram.

PLANCHE 6-3 : Diagramme de Venn pour les 'Minterms' du Blitter

1. Pour sélectionner une fonction D=A (ce qui signifie, cible = A seulement source), on ne sélectionne que les 'minterms' qui se trouvent entièrement enfermés dans le cercle A de la planche ci-dessus. Celui-ci est l'ensemble des 'minterms' 7, 6, 5 et 4. Dès qu'on les écrit comme une suite de 1 pour les 'minterms' sélectionnés et de 0 pour les 'minterms' non sélectionnés, on obtient la valeur :

Numéro du Minterm	7 6 5 4 3 2 1 0
Minterms Sélectionnés	1 1 1 1 0 0 0 0
	F 0 égal à $F0

2. Pour sélectionner une fonction qui est une liaison entre de deux sources, chercher les 'minterms' dans les deux cercles (leur intersection). Par exemple, la liaison AB (A "and" B) est représentée par la partie commune aux deux cercles A et B, ou 'minterms' 7 et 6.

Numéro du Minterms	7 6 5 4 3 2 1 0
Minterms Sélectionnés	1 1 0 0 0 0 0 0
	C 0 égal à $C0

3. Pour utiliser une fonction qui est l'inverse, ou "not", de l'une des sources, comme $\overline{A}$, prendre tous les 'minterms' qui se trouvent à l'extérieur du cercle représenté par A dans la planche ci-dessus. Dans ce cas, nous aurons les 'minterms' 0, 1, 2 et 3.

Numéro du Minterms	7 6 5 4 3 2 1 0
Minterms Sélectionnés	0 0 0 0 1 1 1 1
	0 F égal à $0F

4. Pour relier les 'minterms', ou leur appliquer une opération "or", il faut faire une opération "or" sur leurs valeurs. Par exemple, l'équation AB+BC devient :

Numéro du Minterms	7 6 5 4 3 2 1 0
AB	1 1 0 0 0 0 0 0
BC	1 0 0 0 1 0 0 0
AB+BC	1 1 0 0 1 0 0 0
	C 8 égal à $C8

DECALAGES ET MASQUES

Jusqu'à maintenant nous avons traité du 'blitter' seulement pour déplacer des mots en mémoire et pour les réunir avec des opérations logiques. Ceci suffit pour déplacer des images graphiques, tant que celles-ci se trouvent dans la même position par rapport au début d'un mot. Si le pixel le plus à gauche de notre image de voiture se trouvait au deuxième pixel à partir de la gauche, nous pourrions dessiner facilement cette image à l'écran dans n'importe quelle position où le pixel le plus à gauche commence lui aussi deux pixels après à partir du début d'un mot. Mais, souvent, nous avons besoin de dessiner cette même voiture décalée de quelques pixels à gauche ou à droite. Pour pouvoir le faire, les canaux DMA possèdent un déphaseur en cylindre qui a la possibilité de décaler une image de 0 à 15 bits.

Cette opération de décalage n'est aucunement limitée; elle ne prend pas plus de temps pour exécuter un 'blit' avec décalage qu'un 'blit' sans décalage, à la différence du décalage exécuté avec le 68000. Normalement, ce décalage se fait vers la droite. Le déphaseur permet de déplacer des images dans les limites d'un pixel, même si les pixels sont adressés par 16 à la fois par chaque adresse de mot de l'image du plan de bits.

Ainsi, si les données d'entrée sont décalées à droite, qu'est-ce qui est décalé de la gauche ? Pour le premier mot du 'blit', on décale des zéros; puis, pendant le même 'blit', dans chaque mot qui suit, on décale les données qui ont été décalées dans le mot précédent.

On met en place la valeur de décalage du canal de A avec les bits 15 à 12 de BLTCON0; on met en place la valeur de décalage de B avec les bits 15 à 12 de BLTCON1. Pour plusieurs opérations, on utilisera la même valeur pour les deux décalages. Pour des décalages d'une valeur supérieure à 15 bits, il faut charger le pointeur du registre d'adresse de la zone cible avec une adresse supérieure; pour un décalage de 100 bits, il faudrait avancer le pointeur sur la zone cible de 100/16 ou de 6 mots (12 octets) et opérer un décalage à droite des 4 bits qui restent.

A titre d'exemple, disons qu'on va faire un 'blit' d'une largeur de trois mots et qu'on va utiliser un décalage de 4 bits. Pour simplifier, on présume qu'on fait une copie directe de A à D. Le premier mot qu'on écrira dans D, sera le premier mot prélevé dans A, décalé à droite de quatre bits,

avec des zéros qui ont été décalés de la gauche. Le deuxième mot sera le deuxième mot prélevé dans A, avec les quatre bits les moins significatifs du premier mot décalés. Ensuite, nous écrirons le premier mot de la deuxième ligne, prélevé dans A, décalé de quatre bits, avec les quatre bits les moins significatifs du dernier mot de la première ligne décalés. Cette opération continuera tant que le 'blit' n'aura pas pris fin.

Par conséquent, dans les 'blits' décalés, nous ne prenons que les zéros décalés du premier mot de la première ligne. Dans toutes les autres lignes, le 'blitter' décalera les bits qui ont été décalés dans la ligne précédente. Dans plusieurs applications graphiques, ceci n'est pas souhaitable. Pour cette raison, le 'blitter' offre la possibilité de masquer le premier et le dernier mot de chaque ligne en provenance du canal DMA de A. Ainsi, il est possible d'extraire des données rectangulaires d'une zone source dont les bords gauche et droit se trouvent à un mot limite. Ces deux registres, destinés aux masques du premier et du dernier mot du canal de A du 'blitter', s'appellent BLTAFWM et BLTALWM. Si l'on ne s'en sert pas, il faut les initialiser, tous les deux, avec la valeur ($FFFF).

NOTE

Les fontes texte de l'Amiga sont stockées dans une mappe de bits compactée. Chaque caractère d'une fonte est extrait en utilisant le 'blitter', et en masquant les bits dont on ne veut pas. On peut ensuite placer un caractère à n'importe quel alignement de pixel en le décalant avec une valeur appropriée.

Avant de faire les décalages, on fait une opération logique "and" entre les masques et les données de la zone source. Un bit de la zone source A n'apparaîtra réellement, lors de l'opération logique, que s'il y a un bit à 1, dans le masque du premier mot. On fait une opération logique "and" avec BLTAWFM et le premier mot de chaque ligne, puis on fait cette même opération avec BLTALWM et le dernier mot. Si la largeur d'une ligne est d'un seul mot, on applique les deux masques simultanément.

Les masques sont aussi utiles pour extraire une partie de "colonnes" d'un plan de bits donné. Disons que nous avons, par exemple, un rectangle dessiné auparavant et qui contient du texte et du graphisme d'une largeur de 23 pixels. Le bord le plus à gauche est le bit le plus à gauche dans sa mappe de bits et celle-ci a une largeur de deux mots. Nous voudrions faire le rendu de ce rectangle en le faisant commencer à la position pixel 5 dans notre mappe de bits d'écran, sans déranger l'organisation de ce qui se trouve à l'extérieur du rectangle.

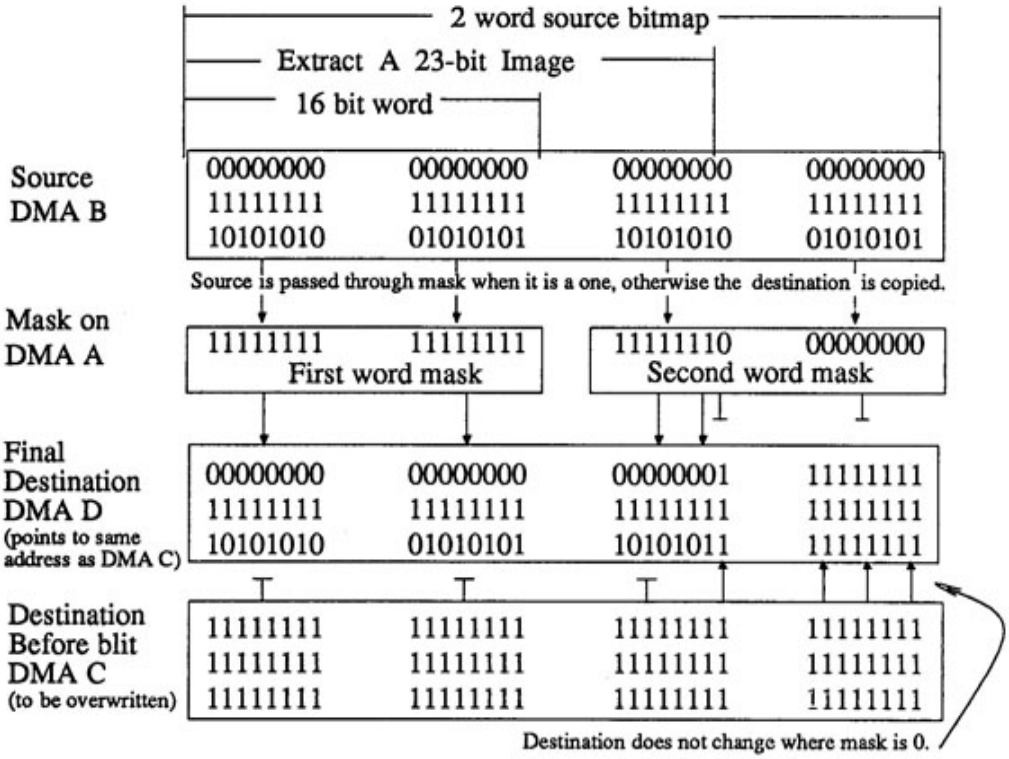


Figure 6-4: Extracting a Range of Columns

PLANCHE 6-4 : Extraction d'une Suite de Colonnes

Pour ce faire, nous faisons pointer le canal DMA de B sur la mappe de bits qui contient l'image source et le canal DMA de D sur la mappe de bits de l'écran. Nous utilisons un décalage de 5. En outre, nous faisons pointer le canal DMA de C sur la mappe de bits de l'écran. Nous utilisons une largeur de 'blit' de deux mots. Ce dont nous avons besoin est une simple opération de copie, si l'on excepte que nous souhaiterions isoler les les cinq premiers bits du premier mot et les quatre derniers bits (2 fois 16, moins 23, moins 5) du dernier mot. Le canal DMA de A nous vient en aide. Nous chargeons auparavant le registre de données de de A avec $FFFF (tous les bits) et nous nous servons d'un masque pour le premier mot avec les cinq bits les moins significatifs mis à zéro ($07FF) et d'un masque des derniers quatre mots les moins significatifs mis à zéro ($07FF). Nous ne validons pas le canal DMA de A, mais seulement les canaux B, C et D, étant donné que nous voulons nous servir de ce canal pour faire un simple masque de ligne. A ce moment, nous voudrons transférer les données de B (la source) aux endroits où le canal de A est à 1 (pour un 'minterm' de AB) et transférer les données cible originales (dans le canal de C) aux endroits où le canal A est à 0 (pour un 'minterm' de $\overline{A}C$), en nous servant de notre fonction 'cookie-cut' classique AB+AC, c'est-à-dire $CA.

NOTE

Même si le canal de A est invalidé, nous nous en servons dans notre fonction logique et en chargeons auparavant les registres de données. L'invalidation d'un canal met simplement hors service les prélèvements en mémoire du canal; toutes les autres opérations continuent à être exécutées, seulement à partir d'une valeur de constante stockée dans le registre de données du canal.

Une manière alternative, mais plus subtile, d'atteindre le même but est d'utiliser un décalage de A de cinq, un masque du premier mot de tous les autres bits et un masque du dernier mot avec les neuf bits les plus à droite mis à zéro. Tous les autres registres restent inchangés.

NOTE

Assurez-vous que vous chargez les registres de données immédiats du 'blitter' seulement après avoir mis en place la valeur de décalage de BLTCON0/BLTCON1, car le fait de charger d'abord les registres de données conduira à des résultats imprévisibles. Par exemple, si une personne avant moi a laissé BSHIFT à "4" et que je charge BDATA avec un "1" et que je change ensuite BSH1 à 2, le BDATA résultant, qui sera utilisé, sera "1<<4", et non "1<<2". L'action de charger un des registres de données "dessine" et décale les données dans la machine.

MODE DESCENDANT (DESCENDING MODE)

Notre 'blit' de copie mémoire fonctionne bien si la source ne chevauche pas la cible. Si nous voulons déplacer une image d'une ligne vers le bas (dans la direction des adresses hautes), cependant, nous nous heurtons à un problème - nous écrivons par-dessus la deuxième ligne que nous avons pris la peine de copier ! Le 'blitter' possède un mode de travail - l'adressage par décrémentation - qui résout joliment ce problème.

On met en route l'adressage par décrémentation en mettant à 1 le bit un de BLTCON1 (défini comme BLITREVERSE). Si on utilise ce mode, les pointeurs d'adresse seront décrémentés de deux (octets) au lieu d'être incrémentés de deux, lors d'un prélèvement de mot. En outre, les valeurs modulo seront soustraites au lieu d'être additionnées. Ensuite, les décalages se feront vers la gauche au lieu de se faire vers la droite, le masque du premier mot masque le dernier mot d'une ligne (qui est aussi le premier mot à être prélevé) et le masque du dernier mot masquera le premier mot de la ligne.

Ainsi, pour une copie de mémoire ordinaire, la seule différence existante, dans la mise en route du 'blitter' (si l'on néglige les décalages et les opérations de masque), sera dans le fait qu'on initialise les registres de pointeur d'adresse pour les faire pointer sur le dernier mot d'un bloc, plutôt que sur le premier. Les valeurs modulo, la taille du 'blit' et tous les autres paramètres devront être mis en place comme auparavant.

NOTE

Ce mode diffère de la pré-décrémentation/post-incrémentation du 68000, où un registre d'adresse doit être initialisé pour pointer sur le mot qui suit le dernier mot, au lieu de pointer sur le dernier mot.

L'adressage par décrémentation est aussi nécessaire pour le remplissage des surfaces, dont nous parlons ci-dessous, dans une autre section.

COPIE DE SECTIONS ARBITRAIRES

L'une des utilisations les plus courantes du 'blitter' est celle qui consiste à déplacer des rectangles de données arbitraires d'un plan de bits à un autre ou à des positions différentes à l'intérieur d'un même plan de bits. D'ordinaire, ces rectangles se trouvent à des coordonnés de bits arbitraires, de telle sorte que les décalages et les masques sont indispensables. Il existe d'autres complications. Il faudra peut-être faire plusieurs lectures et différentes expériences avant que cette section ne soit bien comprise dans le détail.

Une image source qui n'occupe que deux mots peut s'étendre sur trois, si on la copie avec certains décalages. Notre rectangle ci-dessus d'une largeur de 23 pixels, par exemple, lorsqu'il est décalé de 12 bits, s'étendra sur trois mots. A l'opposé, une image qui s'étend sur trois mots peut convenir à deux mots, lors de certains décalages. Dans de tels cas, on devra utiliser la taille du 'blit' supérieure, pour que source et cible puissent s'adapter dans une même taille. On devra mettre des masques adaptés pour masquer les données indésirables.

Ci-dessous, quelques lignes d'indications pour faire la copie d'une section arbitraire.

1. Utiliser le canal DMA de A, invalidé, pré-chargé avec tous ses bits à zéro, ainsi que les valeurs de masque et de décalage appropriées, pour masquer la fonction 'cookie-cut'. Utiliser le canal de B pour prélever les données source, le canal de C pour prélever les données cible et le canal de D pour écrire les données cible. Utiliser la fonction 'cookie-cut' $CA.

2. En cas de décalages, utiliser toujours l'adressage par incrémentation, si le décalage des bits se fait sur la droite, et utiliser l'adressage par décrémentation si le décalage des bits se fait sur la gauche.

NOTE

Ces décalages sont ceux qui concernent la position des bits du bord le plus à gauche à l'intérieur d'un mot, et non les décalages absolus, comme on l'a expliqué précédemment.

3. Si la source et la cible se chevauchent, utiliser l'adressage d'incrémentation, si la cible a une adresse mémoire plus basse (la plus haute dans l'affichage) et, autrement, utiliser l'adressage d'incrémentation.

4. Si la source s'étend sur plus de mots que la cible, utiliser la même valeur de décalage pour le canal de A et pour le canal source B et mettre les masques du dernier et du premier mot comme s'ils devaient masquer les données source de B.

5. Si la cible s'étend sur plus de mots que la source, utiliser une valeur de décalage de zéro pour le canal de A et mettre les masques du dernier et du premier mot comme s'ils devaient masquer les données cible D.

6. Si la source et la cible s'étendent sur un même nombre de mots, utiliser le canal de A pour masquer soit la source, comme au point 4, soit la cible, comme au point 5.

NOTE

Les conditions 2 et 3 peuvent se trouver en contradiction, si on essaie de déplacer une image d'un pixel en bas et à droite. Dans ce cas, nous voudrions nous servir de l'adressage de décrémentation pour que notre cible n'écrive pas par-dessus notre source avant que l'on n'ait utilisé la source, mais nous voudrions aussi utiliser l'adressage d'incrémentation pour les décalages à droite. Dans certains cas, il est possible de se référer au point 2 ci-dessus en faisant des masques astucieux. Mais, parfois, le fait de masquer seulement le premier ou le dernier mot peut ne pas suffire; il peut être nécessaire de masquer plus de 16 bits à un bout ou à l'autre. Dans ce cas, on peut construire en mémoire un masque pour une seule ligne de trame et valider le canal DMA de A pour qu'il prélève ce masque de manière explicite. En mettant la valeur modulo de A à la valeur négative de la largeur du masque, celui-ci sera prélevé à chaque ligne de manière répétitive.

MODE REMPLISSAGE DE SURFACES

En plus de copier des données, le 'blitter' peut simultanément exécuter une opération de remplissage pendant la copie. Cette opération ne présente qu'une seule restriction - la surface à remplir doit d'abord être définie par des lignes avec un tracé sans grain ayant un seul bit mis à 1 pour la ligne horizontale. Pour cette opération, on dispose d'un mode de tracé de ligne particulier. Utiliser un 'blit' de copie ordinaire (ou tout autre 'blit', étant donné que les remplissages de surface viennent après tous les décalages, les masques et les combinaisons logiques des sources).
On devra utiliser l'adressage de décrémentation. Mettez à un, dans BLTCON1, soit le bit de validation de remplissage inclusif (FILL OR, ou bit 3), soit le bit de validation de remplissage exclusif (FILL XOR, ou bit 4).
Le mode de remplissage inclusif remplit entre les lignes, en laissant les lignes intactes.
Le mode de remplissage exclusif remplit entre les lignes, en laissant intactes les lignes qui entourent le bord droit des surfaces qu'il a remplies, mais en effaçant les lignes qui entourent le bord gauche.
Le remplissage exclusif réduit d'un pixel les surfaces qu'il a remplies par rapport au même modèle rempli avec un remplissage inclusif.

Par exemple le modèle :

00100100-00011000

rempli avec un remplissage inclusif, donne :

00111100-00011000

tandis qu'avec un remplissage exclusif, on aura pour résultat :

00011100-00001000

(Naturellement, les remplissages se font toujours sur la totalité des mots 16 bits.)

Il existe un autre bit (FILL_CARRYIN ou bit 3 de BLTCON1) qui force le remplissage d'un surface "en dehors" des lignes; pour l'exemple ci-dessus, avec un remplissage inclusif, voici ce qu'on aurait en sortie :

11100111-11111111

avec un remplissage exclusif, on aurait en sortie :

11100011-11110111

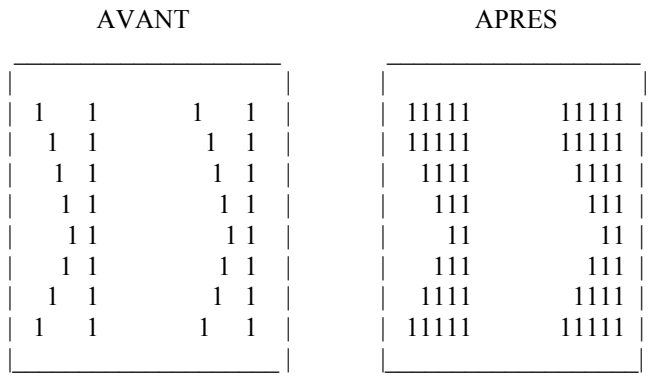


PLANCHE 6-5 : Utilisation du Bit FCI - le Bit est à 0

Si le bit FCI est un 1, au lieu d'être un 0, la surface en dehors des lignes est remplie avec des 1 et celle à l'intérieur des lignes est laissée avec des 0 entre les lignes.

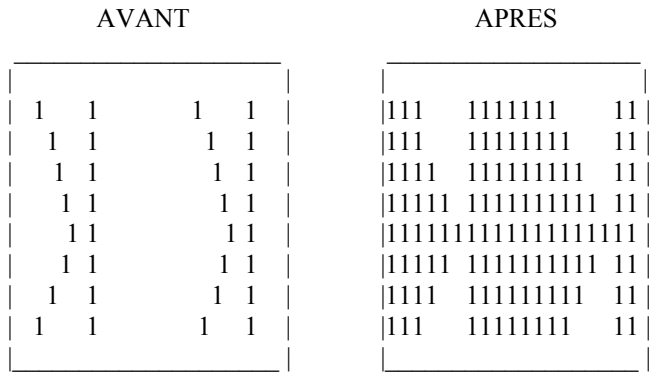


PLANCHE 6-6 : Utilisation du Bit FCI - le Bit est à 1

Si l'on veut produire des sommets à un point, très aigus, on devra valider le mode remplissage exclusif. La Planche 6-7 montre qu'un sommet à un seul point est produit en utilisant le mode remplissage exclusif.

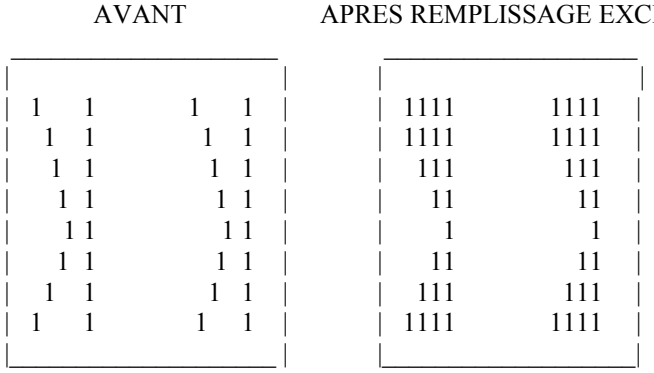


PLANCHE 6-7 : Exemple de Sommet à un Seul Point

Le 'blitter' utilise le bit 'carry-in' de remplissage comme début de l'état de remplissage en commençant du bord le plus à droite de chaque ligne. Le 'blitter' permute l'état de remplissage de chaque bit "1" de la surface source, soit en remplissant, soit en ne remplissant pas l'espace avec des uns. Ce travail continue dans chaque ligne tant qu'on n'atteint pas le bord gauche du 'blit', où le remplissage s'arrête.

LE DRAPEAU 'DONE' DU 'BLITTER'

Quand on écrit dans le registre BLTSIZE, le 'blit' est lancé. Le processeur, cependant, ne s'arrête pas pendant que le 'blitter' travaille; ils peuvent travailler tous les deux en même temps, ce qui contribue pour beaucoup à la vitesse évidente de l'Amiga. L'utilisation du 'blitter' n'exige pas de soins particuliers.

Il y a un drapeau 'done' du 'blitter', qu'on appelle aussi le drapeau d'indisponibilité du 'blitter' : c'est le drapeau DMAF_BLTDONE dans DMACONR. Ce drapeau est mis à 1, lorsque le 'blit' progresse.

NOTE

Si on a démarré un 'blit' mais on lui a verrouillé l'accès en mémoire à cause, par exemple, des prélèvement destinés à l'affichage, ce bit peut ne pas se trouver encore à 1. D'autre part, le processeur, totalement débloqué, peut se trouver en train de tourner en dehors de la mémoire FAST ou en dehors de son antémémoire interne, par conséquent il continuera à avoir des cycles de mémoire disponibles.

La solution est de lire la mémoire 'chip' ou de lire l'adresse du registre 'hardware' avec le micro-processeur avant de tester ce bit. On peut faire cela facilement avec la séquence suivante :

```
btst.b #DMAB_BLTDONE-8,DMACONR(a1)
btst.b #DMAB_BLTDONE-8,DMACONR(a1)
```

où "a1" a été pré-chargé avec l'adresse des registres 'hardware'. Le premier "test" du bit 'done' du 'blitter' peut ne pas retourner un résultat correct, mais le deuxième le retournera.

NOTE

On a "corrigé" le démarrage du bit 'busy' du 'blitter' avec Fat Agnus, pour que ce bit passe à 1 dès qu'on écrit dans BLTSIZE pour lancer le 'blit', au lieu de passer à 1 quand le 'blitter' commence son premier cycle DMA. Cependant, toutes les nouvelles machines n'utiliseront pas ces nouvelles puces, il est par conséquent mieux de se fier à la méthode de test exposée auparavant.

LE MULTITACHE ET LE 'BLITTER'

Quand un 'blit' avance, on ne devrait écrire dans aucun des registres du 'blitter'. Pour des détails, sur l'arbitrage des accès du 'blitter' dans le système, référez-vous au ROM Kernel Manual. Lisez, tout particulièrement, la discussion sur les fonctions OwnBlitter() et DisownBlitter(). Même après avoir "maîtrisé" le 'blitter', un 'blit' peut être encore en train de se terminer, par conséquent, on devra vérifier l'état du drapeau 'done' du 'blitter' avant de l'utiliser, même la première fois. On recommande l'usage de la fonction WaitBlit() du ROM Kernel.

On devrait aussi tester le drapeau 'done' du 'blitter' avant de se servir des résultats d'un 'blit'. Celui-ci peut ne pas être terminé, les données donc peuvent ne pas être encore prêtes. Ce qui peut causer des difficultés pour trouver l'origine des bogues car un 68000 peut être assez lent pour qu'un 'blit' se termine sans que le 68000 puisse tester le drapeau 'done', tandis qu'un 68020, peut-être en tournant en dehors de son antémémoire, peut réussir à atteindre les données avant que le 'blitter' n'ait fini de les écrire.

Disons que nous avons une sous-routine qui affiche une boîte de texte au-dessus d'autres images temporairement. Cette sous-routine aurait la possibilité d'allouer un 'chunk' de mémoire qui pourrait contenir l'image originale de l'écran, pendant que nous affichons notre boîte de texte. En se terminant, cette sous-routine aurait la possibilité de faire un 'blit' sur les images originales et de libérer ensuite la mémoire réservée. Si on libère cette mémoire avant d'avoir testé le drapeau 'done' du 'blitter', d'autres processus pourraient réserver cette mémoire et y stocker des nouvelles données avant la fin du 'blit', en se débarrassant de la source du 'blitter' et en restaurant ainsi toutes les images d'écran.

DRAPEAU D'INTERRUPTION

Le 'blitter' possède aussi un drapeau d'interruption qui est mis à 1 à la fin d'un 'blit'. Ce drapeau, INTF BLIT, s'il est validé, peut générer une interruption du 68000. Pour plus de renseignements sur les interruptions, voir le Chapitre 7, "Le 'Hardware' de Contrôle du Système".

LE DRAPEAU ZERO

Le 'blitter' possède aussi un drapeau 'zero' qu'on peut tester pour déterminer si l'opération logique sélectionnée donne un résultat de zéro pour tous les bits cible, même si ces bits cible n'ont pas été écrits car le canal DMA de D était invalidé. Cette caractéristique est souvent utile pour détecter la collision en exécutant un "and" logique sur les images source pour tester le chevauchement. Si les images ne se chevauchent pas, le drapeau 'zero' donnera vrai.

Le drapeau 'zero' est valide seulement quand le 'blitter' a terminé son opération et on peut le lire à partir du bit DMAF_BLTNZERO du registre DMACONR.

REGISTRE PIPELINE

Le 'blitter' exécute plusieurs opérations à chaque cycle - décalant et masquant les mots source, la liaison logique des sources et le remplissage de la surface et la détection d'un zéro en sortie. Pour valider autant de choses qui doivent se mettre en place si rapidement le 'blitter' fonctionne comme un pipeline. Ce qui signifie -qu'au lieu d'exécuter toutes les opérations citées précédemment en un cycle de 'blitter'-, qu'il disperse ces opérations en deux cycles de 'blitter'. (Ici, nous employons le mot "cycle" dans un sens approximatif pour simplifier.) Pour éclaircir cela, on peut imaginer le 'blitter' comme deux circuits connectés en série. A chaque cycle, une nouvelle suite d'opérations source arrive en entrée et le premier circuit exécute des opérations sur ces données. Il passe ensuite les données à moitié traitées au deuxième circuit pour que celui-ci finisse de les traiter lors du cycle suivant, lorsque le premier circuit sera occupé à travailler sur la prochaine série de données. Chaque série de données occupe deux "cycles" pour transiter dans les deux circuits, une série de données ainsi recouverte pourra être pompée à chaque cycle.

Ce qui veut dire que les deux premières séries de données source sont prélevées avant qu'on ait écrit dans la première cible. Ceci permet de décaler une mappe de bits d'un mot à droite en utilisant, par exemple, le mode d'adressage par incrémentation, même si, normalement, des parties de la cible pourraient être effacées avant d'être prélevées.

Tableau 6-2 : Séquence Typique d'un Cycle du 'Blitter'

Code USE dans BLTCON0	Canaux Actifs	Séquence du Cycle
F	A B C D	A0 B0 C0 - A1 B1 C1 D0 A2 B2 C2 D1 D2
E	A B C	A0 B0 C0 A1 B1 C1 A2 B2 C2
D	A B D	A0 B0 - A1 B1 D0 A2 B2 D1 - D2
C	A B	A0 B0 - A1 B1 - A2 B2
B	A C D	A0 C0 - A1 C1 D0 A2 C2 D1 - D2
A	A C	A0 C0 A1 C1 A2 C2
9	A D	A0 - A1 D0 A2 D1 - D2
8	A	A0 - A1 - A2
7	B C D	B0 C0 - - B1 C1 D0 - B2 C2 D1 - D2
6	B C	B0 C0 - B1 C1 - B2 C2
5	B D	B0 - - B1 D0 - B2 D1 - D2
4	B	B0 - - B1 - - B2
3	C D	C0 - - C1 D0 - C2 D1 - D2
2	C	C0 - C1 - C2
1	D	D0 - D1 - D2
0	aucun	

Notes pour le Tableau ci-dessus :

o Aucun remplissage.

o Aucune activité concurrente dans le bus.

o 'Blit' de trois mots.

o Une opération typique comprend un prélèvement de toutes les sources deux fois pour que la première cible soit disponible. Ceci est dû au phénomène de pipeline interne. On doit faire attention au chevauchement des zones cible et source.

NOTE

Ce Tableau ne veut être qu'une illustration de l'ordre typique des cycles du 'blitter' dans le bus. Les cycles du bus sont réservés de manière dynamique en s'appuyant sur le mode opérationnel du 'blitter'; activité concurrente du bus à partir du processeur, des plans de bits, et d'autres canaux DMA; et d'autres facteurs encore. Commodore Amiga ne garantit pas cette précision ni une future compatibilité avec ce diagramme. Nous nous réservons le droit d'apporter des améliorations au produit et de faire des modifications dans cette partie sans information préalable.

MODE LIGNE

Outre les fonctions décrites ci-dessus, le 'blitter' a la possibilité de tracer des lignes d'après des modèles. Le mode ligne se sélectionne en mettant à 0 (LINEMODE) dans BLTCON1, qui modifie la signification de quelques autres bits dans BLTCON0 et BLTCON1. En mode ligne, le 'blitter' a la possibilité de tracer des lignes d'une longueur de 1024 pixels, de les dessiner dans une grande variété de modes et même de pouvoir dessiner des lignes d'une manière particulière pour le simple remplissage d'une surface.

Plusieurs registres du 'blitter' servent à des choses différentes en mode ligne. Consulter l'Appendice A pour une description plus détaillée de l'utilisation de ces registres et des bits de contrôle en mode ligne.

En mode ligne, le 'blitter' trace une ligne d'un point à un autre, qu'on peut considérer comme un vecteur. La direction du vecteur peut s'étendre dans l'un des huit octants ci-dessous. (Dans le diagramme qui suit, on utilise la convention standard de l'Amiga, avec x qui croît sur la droite et y qui croît vers le bas.) Le nombre entre parenthèses est le numéro de l'octant; le deuxième nombre représente la valeur qu'on devrait placer dans les bits 4 à 2 de BLTCON1.

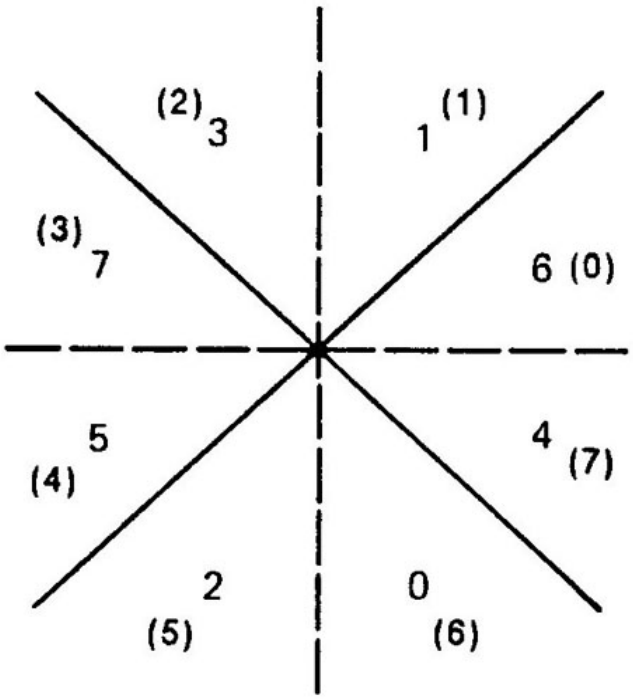


PLANCHE 6-8 : Octants pour le Tracé d'une Ligne

Le tracé de ligne qui se base sur des octants est une simplification qui profite de la symétrie existant entre x et -x et y et -y. Le Tableau ci-dessous liste le nombre d'octants et les valeurs correspondantes :

Tableau 6-3 : Bits de Code BLTCON1 pour Tracer les Octants d'une Ligne

Bits de Code BLTCON1	Octant #
1 1 0	0
0 0 1	1
0 1 1	2
1 1 1	3
1 0 1	4
0 1 0	5
0 0 0	6
1 0 0	7

Nous initialisons les bits de 4 à 2 de BLTCON1 en respectant le tableau ci-dessus. Maintenant, nous introduisons les variables dx et dy et les mettons respectivement aux valeurs absolues entre les coordonnés x et les coordonnés y des points terminaux de la ligne.

```
dx = abs (x2 - x1)
dy = abs (y2 - y1)
```

Maintenant, nous les réajustons, si cela est nécessaire, pour que dx soit supérieur à dy.

```
if (dx < dy)
   {
   temp = dx;
   dx = dy;
   dy = temp;
   }
```

Comme alternative, mettre dx et dy comme suit :

```
dx = max(abs(x2 - x1), abs(y2 - y1)) ;
dy = min(abs(x2 - x1), abs(y2 - y1)) ;
```

Ces calculs ont pour effet de "normaliser" notre ligne dans l'octant 0; étant donné que nous avons déjà informé le 'blitter' sur l'octant réel qu'il doit employer, celui-ci n'aura aucune difficulté à tracer la ligne.

Nous initialisons le pointeur de registre de A à 4 * dy - 2 * dx. Si cette valeur est négative, nous mettrons le bit de signe (SIGNFLAG dans BLTCONl), autrement nous le mettrons à zéro. Nous mettons le registre de modulo de A à 4 * (dy - dx) et celui de B à 4 * dy.

On devra pré-charger le registre de données de A avec $8000. On devra mettre les deux masques de mot à $FFFF. On devra mettre la valeur de décalage de A à la coordonné x du premier point (x1) modulo 15.

On devra initialiser le registre de données avec le modèle de la ligne de texture, s'il y en a un, ou avec $FFFF pour une ligne pleine. On devra mettre la valeur de décalage de B au numéro du bit où commence la texture de la ligne (zéro indique le bit le moins significatif).

On devra initialiser les registres de pointeur de C et de D avec le mot qui contient le premier pixel de la ligne; on devra mettre les registres de modulo de C et de D à la largeur du plan de bits en octets.

On devra mettre à un les bits SRCA, SRCC et DEST de BLTCON0, et le drapeau SRCB à zéro. On devra réinitialiser OVFLAG. Si l'on ne désire qu'un seul bit par ligne horizontale, on devra mettre le bit ONEDOT de BLTCON1; autrement, on devra le réinitialiser.

La fonction logique reste la même. Le canal DMA de C représente la zone source originale, le canal A le bit qu'on doit mettre dans la ligne et le canal B le modèle à dessiner. Ainsi, pour dessiner une ligne, la fonction $AB+\overline{A}C$ sera la plus courante.
Pour dessiner une ligne en utilisant le mode 'or' exclusif on peut effacer facilement une ligne en la redessinant à nouveau, on peut employer la fonction $AB\overline{C}+\overline{A}C$.

Nous mettons la hauteur du 'blit' à la largeur de la ligne, qui est égale à dx + 1. La largeur devra être mise à deux pour tous les tracés de ligne. (Naturellement, on ne devra pas écrire dans le registre BLTSIZE jusqu'au dernier moment, quand tous les autres registres seront remplis.)

RECAPITULATION DU REGISTRE EN MODE LIGNE

Mise en place préliminaire :

La ligne va de (x1 ,y1) à (x2,y2).

dx = max (abs (x2 - x1), abs (y2 - y1))
dy = min (abs (x2 - x1), abs (y2 - y1))

Mise en place du registre :

BLTADAT = $8000
BLTBDAT = modèle de la ligne de texture ($FFFF pour une ligne pleine)

BLTAFWM = $FFFF
BLTALWM = $FFFF

BLTAMOD = 4 * (dy-dx)
BLTBMOD = 4 * dy
BLTCMOD = largeur du plan de bits en octets
BLTDMOD = largeur du plan de bits en octets

BLTAPT = (4 * dy) - (2 * dx)
BLTBPT = inutilisé
BLTCPT = mot renfermant le premier pixel de la ligne
BLTDPT = mot renfermant le premier pixel de la ligne

BLTCON0 bits 15-12 = x1 modulo 15
BLTCON0 bits SRCA, SRCC et SRCD = 1
BLTCON0 bit SRCB = 0
si mode ligne 'or' exclusif :
  alors octet de contrôle LF de BLTCON0 = $AB\overline{C} + \overline{A}C$
  autrement octet de contrôle LF de BLTCON0 = $AB + \overline{A}C$

BLTCON1 bit LINEMODE = 1
BLTCON1 bit OVFLAG = 0
BLTCON1 bits 4-2 = numéro de l'octant d'après le tableau
BLTCON1 bits 15-12 = bit de démarrage pour la texture de la ligne (0 = dernier bit significatif)
si (((4 * dy) - (2 * dx)) < 0) :
  alors bit SIGNFLAG de BLTCON1 = 1
  autrement bit SIGNFLAG de BLTCON1 = 0
si un pixel/ligne :
  alors bit ONEDOT de BLTCON1 = 1
  autrement bit ONEDOT de BLTCON1 = 0

BLTSIZE bits 15-6 = dx + 1
BLTSIZE bits 5-0 = 2

NOTE

On doit activer le registre BLTSIZE en dernier car celui-ci fait commencer
un 'blit'.

VITESSE D'EXECUTION DU 'BLITTER'

La vitesse d'exécution du 'blitter' dépend uniquement du nombre de canaux DMA qui sont validés. En ligne générale, on pourrait employer un canal DMA, mais tant que celui-ci ne sera pas validé, il ne comptera pas. Le cycle minimum du 'blitter' est de quatre impulsions d'horloge; le cycle maximum est de huit. L'emploi du registre de A est toujours disponible.
L'emploi du registre de B ajoute toujours deux impulsions d'horloge au cycle du 'blitter'.
L'emploi de C ou celui de D est toujours disponible, mais leur emploi en même temps ajoute deux autres impulsions d'horloge.
Ainsi, un cycle de copie, qui utilise A et D, prendra quatre impulsions d'horloge par cycle; un cycle de copie, qui utilise B et D, prendra six impulsions d'horloge par cycle et une copie de bits généralisée qui utilise B, C et D prendra huit impulsions d'horloge par cycle.
Lorsqu'on se trouve en mode ligne, chaque pixel prend huit impulsions d'horloge.

Pour les Amiga en NTSC, la vitesse d'horloge du système est de 7.16 MHertz (pour les Amiga en PAL, elle est de 7.09 MHertz). L'horloge du 'blitter' est l'horloge du système. Pour calculer le temps total requis par un 'blit' en micro-secondes, en faisant abstraction du temps de mise en route et des conflits entre DMA, il faut employer l'équation suivante (en NTSC) :

$$t = \frac{n * H * W}{7.16}$$

En PAL :

$$t = \frac{n * H * W}{7.09}$$

où, respectivement, t est le temps en micro-secondes, n est le nombre d'impulsions d'horloge par cycle, tandis que H et W sont la hauteur et la largeur (en mots) du 'blit'.

Par exemple pour copier un plan de bits d'écran de 320 par 200 dans un autre plan de bits, nous pourrions choisir d'utiliser les canaux de A et de D. Ce qui demanderait quatre impulsions d'horloge par cycle du 'blitter', pour un total de

$$\frac{4 * 200 * 20}{7.16} = 2235 \text{ micro-secondes.}$$

Ces mesures ne prennent pas en compte le temps de mise en route du 'blitter' qui est égal au temps nécessaire pour calculer et charger les registres du 'blitter' et démarrer le 'blit'. Ces mesures ignorent aussi les conflits entre DMA.

OPERATIONS DU 'BLITTER' ET DMA DU SYSTEME

Les operations du 'blitter' affectent les performances du reste du système. Dans les sections qui suivent, nous expliquons de quelle manière sont affectées les performances du système par la priorité d'accès à la mémoire du système du 'blitter', par le temps imparti aux DMA dans la plage de temps, par le partage du bus entre le 68000 et le 'hardware' qui s'occupe de l'affichage, par les opérations du 'blitter' et du Copper ainsi que par les différentes tailles d'affichage des aires de jeu.

Le 'blitter' exécute ses différentes opérations de stockage, de modification et de prélèvement de données par le biais de séquences DMA et partage les accès à la mémoire avec les autres périphériques du système. Chaque périphérique qui accède à la mémoire possède un niveau de priorité qui lui est assigné, indiquant son importance par rapport aux autres périphériques.

Les DMA du lecteur de disquette, de l'audio, de l'affichage et des sprites ont tous un niveau de priorité très élevé. Dans certaines circonstances, les DMA de l'affichage sont prioritaires par rapport aux DMA des sprites. Chacun de ces quatre périphériques possède une plage de temps lors du balayage horizontal du faisceau électronique. Si un périphérique ne demande pas la plage de temps qui lui est impartie, ce temps est disponible pour d'autres utilisations. Ces périphériques ont une priorité imposée d'avance car des cycles DMA ratés peuvent entraîner une perte de données, du bruit dans la sortie audio ou des interruptions d'affichage.

Le Copper possède le niveau de priorité suivant car il doit exécuter ses opérations en même temps que la trame d'affichage pour rester synchronisé avec le faisceau électronique d'affichage qui balaie tout l'écran.

Des priorités de niveau plus bas sont assignées au 'blitter' et au 68000, dans cet ordre. Le 'blitter' a une priorité de plus haut niveau car il exécute la copie et la modification de données, ainsi que les opérations de tracé de ligne plus vite que le 68000.

Pendant une ligne de balayage horizontale (environ 63 micro-secondes), il y a 227,3 "impulsions d'horloge couleur", c'est-à-dire des cycles d'accès en mémoire. La durée approximative d'un cycle mémoire est de 280 nano-secondes. Cet ensemble de 227,5 cycles par ligne horizontale comprend soit le temps d'affichage que celui de non affichage. Sur le temps total, 226 cycles sont disponibles pour être réservés aux différents périphériques qui ont besoin d'avoir accès à la mémoire.

Voici de quelle manière est réservée la plage de temps impartie dans chaque ligne horizontale :

4 cycles pour le rafraîchissement de la mémoire
3 cycles pour les DMA du lecteur de disquettes
4 cycles pour les DMA audio (2 octets par canal)
16 cycles pour les DMA des sprites (2 mots par canal)
80 cycles pour les DMA des plans de bits (plages de temps de numéro pair ou impair selon les dimensions de l'affichage qu'on utilise)

La Planche 6-9 montre une ligne de balayage horizontale complète et la manière dont on réserve les cycles d'horloge.

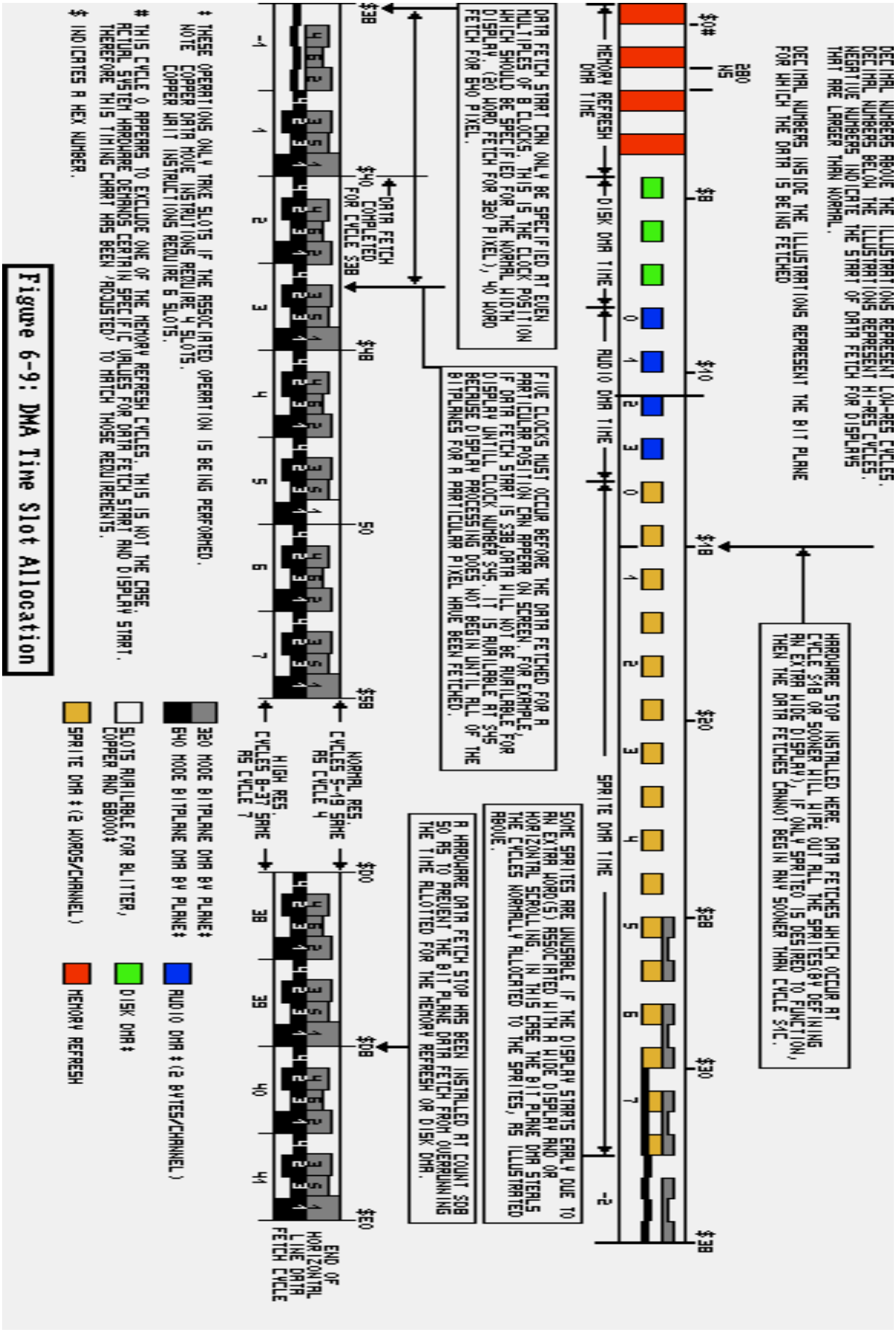


PLANCHE 6-9 : Réservation de la Plage de Temps pour les DMA

Le 68000 n'emploie que les cycles d'accès mémoire de numéro pair. Le 68000 dépense la moitié environ du temps complet d'une instruction processeur pour faire des opérations internes et l'autre moitié pour ses accès en mémoire. Par conséquent, cette réservation de cycles de mémoire alternés laisse croire au 68000 qu'il a accès à la mémoire sans arrêt et donc il continue à fonctionner à vitesse maximale.

Certaines instructions du 68000 ne s'accordent pas parfaitement avec la réservation des cycles pairs et provoquent la perte de quelques cycles. S'il rate des cycles, le 68000 doit attendre jusqu'à la prochaine plage de temps pour pouvoir disposer de mémoire avant de continuer. Cependant, la plupart des instructions ne fait pas rater de cycles et donc le 68000 fonctionne à vitesse maximale la plupart du temps, s'il n'y a pas d'interférence avec les DMA du 'blitter'.

La Planche 6-10 illustre un cycle normal du 68000.

NOTE

Avec l'Amiga, on ne devrait jamais utiliser l'instruction 'test-and-set' (TAS) du 68000; le cycle indivisible d'écriture-modification-lecture, utilisé seulement dans cette instruction, ne s'adapte pas dans l'une des plages de temps d'accès en mémoire des DMA.

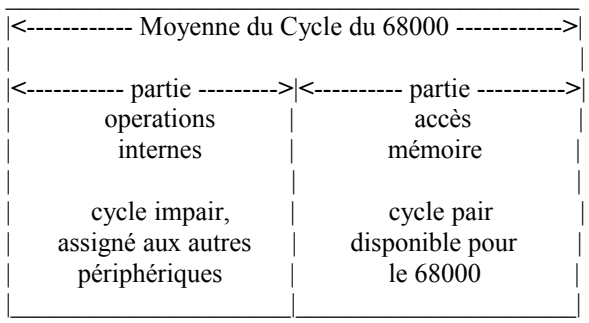


Planche 6-10 : Cycle Normal du 68000

Si l'affichage contient quatre, ou moins de quatre, plans de bits en basse résolution, on peut garantir au 68000 des cycles de mémoire alternés (s'il est, à la fois, prêt à demander un cycle et s'il est dans une zone de priorité maximale). Cependant, s'il y a plus de quatre plans de bits, les DMA des plans de bits commenceront, lors de l'affichage, à voler des cycles au 68000.

Pendant le temps d'affichage d'un affichage avec six plans de bits (en basse résolution, d'une largeur de 320 pixels), les DMA des plans de bits prendront 160 plages du temps imparti au 68000 à chaque ligne horizontale. Comme on peut le voir dans la Planche 6-11, les DMA des plans de bits volent 50 % de la plage impartie au processeur que celui-ci aurait pu utiliser s'il y avait eu un affichage avec quatre plans de bits.

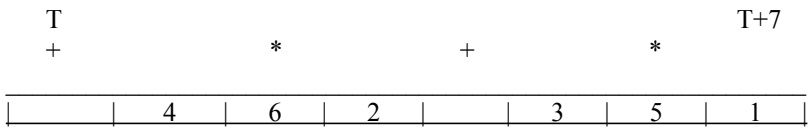


Planche 6-11 : Plages de Temps Utilisées par un Affichage avec Six Plans de Bits

Si on indique quatre plans de bits en haute résolution (d'une largeur de 400 pixels), les DMA des plans de bits ont besoin de toutes les plages de temps de mémoire disponibles pour ne prélever que les 40 mots de données de chaque ligne des quatre plans de bits (40 * 4 = 160 plages de temps). Ce qui verrouille effectivement le 68000 (aussi bien que le 'blitter' et le Copper) pour des accès en mémoire pendant l'affichage, en dehors des suppressions verticale et horizontale.

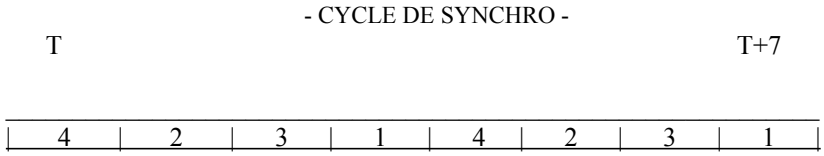


Planche 6-12 : Plages de Temps Utilisées par un Affichage en Haute Résolution

Chaque ligne d'un affichage normal, de dimensions maximales, contient 320 pixels en mode basse résolution ou 640 pixels en mode haute résolution. Ainsi, on prélèvera soit 20 soit 40 mots pendant le temps d'affichage d'une ligne horizontale. Si l'on veut faire défiler une aire de jeu, il faudra prélever un mot supplémentaire par ligne en mémoire.

La taille d'un affichage est réglable (cf. le Chapitre 3, "Le 'Hardware des Aires de Jeu") et les DMA des plans de bits ont la priorité sur les DMA des sprites. Comme nous l'indiquons dans la Planche 6-9, des affichages plus grands peuvent bloquer un ou plusieurs sprites d'un numéro d'ordre élevé, tout particulièrement lors d'un déroulement.

Comme mentionné ci-dessus, le 'blitter', normalement, a une priorité plus élevée que le processeur pour les cycles DMA. Cependant, il y a certains cas où le 'blitter' et le 68000 peuvent partager des cycles mémoire. Si on lui en donne la possibilité, le 'blitter' volera chaque cycle de mémoire disponible. Les DMA du lecteur de disquettes, de l'audio et de l'affichage ont la priorité sur le 'blitter', celui-ci donc ne peut pas les bloquer dans les accès au bus. Selon le réglage du bit de mode DMA du 'blitter', à qui on fait référence d'habitude comme au bit "blitter-nasty" [bit dangereux du 'blitter'], on peut bloquer l'accès au bus au processeur. Ce bit s'appelle BLITHOG et se trouve dans le registre DMACON.

Si DMAF_BLITHOG est à 1, le 'blitter' conservera le bus pour tout cycle de mémoire disponible. Potentiellement, ceci pourrait se produire pour chaque cycle.

Si DMAF_BLITHOG est à 0, le superviseur des DMA contrôlera les demandes de cycles du 68000. Si on ne satisfaisait pas à ces demandes, pendant trois cycles consécutifs de mémoire, le 'blitter' libérerait le bus pendant un cycle.

SCHEMA FONCTIONNEL PAR BLOCS DU 'BLITTER'

La Planche 6-13 indique les blocs de construction principaux d'un seul bit pendant une opération du 'blitter' d'une largeur de 16 bits. Le schéma ne couvre pas le 'hardware' qui trace la ligne.

- Le coin supérieur gauche montre comment on applique le premier -et le dernier- masque de mot aux données en provenance de la source A. Quand le 'blit' réduit à la largeur d'un seul mot, on applique les deux masques.

- Le dessin du déphaseur (en haut à droite et au centre à gauche) illustre comment on prend 16 bits de données à une position indiquée à l'intérieur du registre 32 bits, en s'appuyant sur les valeurs de décalage de B ou de A indiquées dans BLTCON0 et dans BLTCON1.

- Le générateur de 'minterms' (au centre à droite) illustre comment les bits de sélection d'un 'minterm' soit autorisent soit inhibent l'utilisation d'un 'minterm' déterminé.

- Le dessin montre comment fonctionne l'opération de remplissage sur les données générées par les combinaisons du 'minterm'. Ces opérations peuvent s'exécuter en même temps que d'autres opérations logiques complexes.

- En haut, le dessin montre qu'on peut éviter d'écrire les données générées pour la zone cible, dans une cible, en utilisant un des bits de contrôle du 'blitter'.

- On ne montre pas dans ce schéma la logique de détection du zéro qui s'occupe de chaque bit généré pour la zone cible. Si, pour la cible, on génère des bits à 1, la logique indique si la surface du 'blit' contenait au moins un bit à 1 (la détection du zéro indique que l'assertion est fausse).

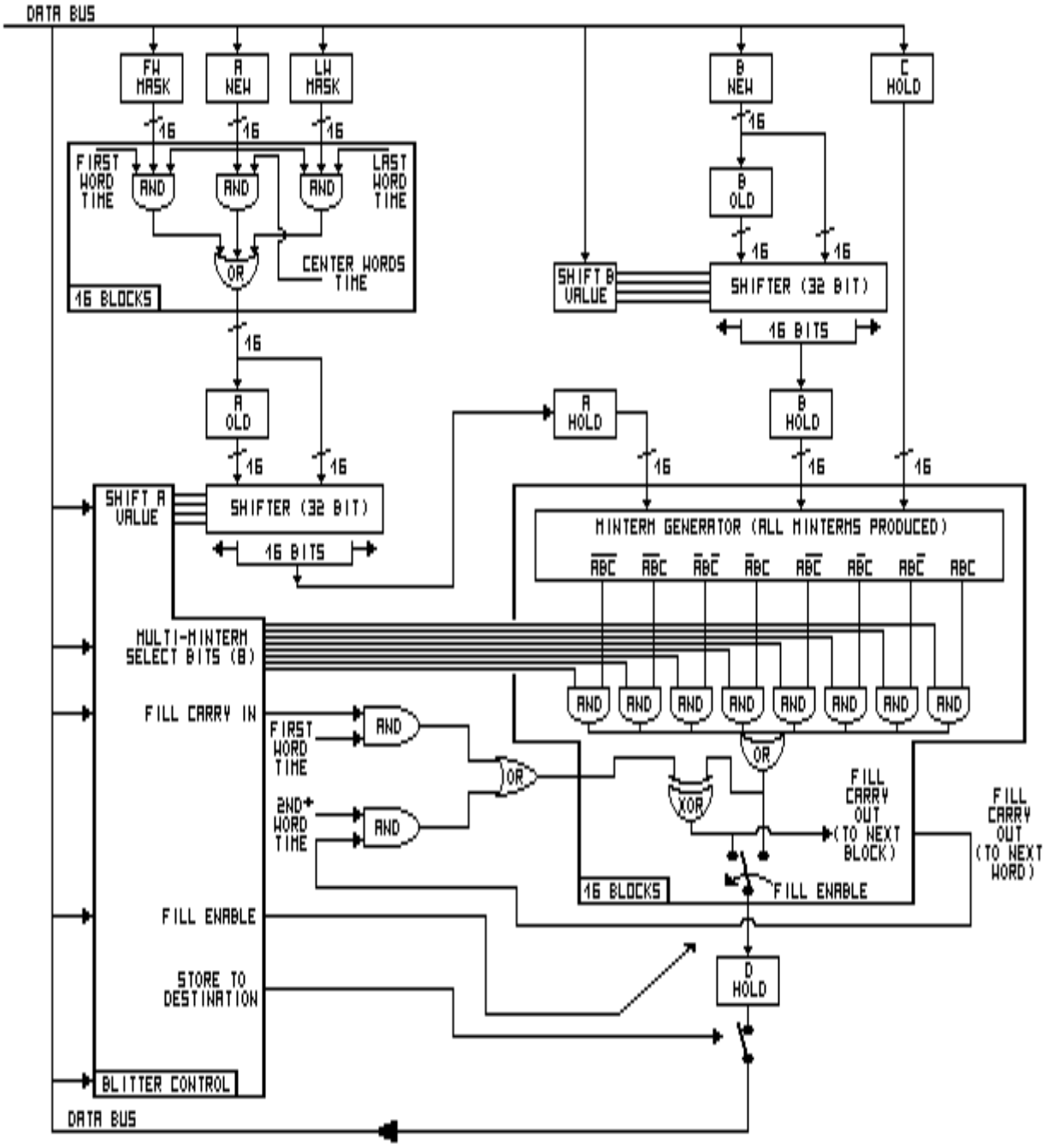


FIGURE 6-13: Blitter Block Diagram

PLANCHE 6-13 : Schéma Fonctionnel par Blocs du 'Blitter'

LES POINTS CLE DU 'BLITTER'

C'est une liste de quelques points clé dont il faudra se souvenir quand on programme le 'blitter'.

- o Ecrire BLTSIZE en dernier; l'écriture de ce registre déclenche le 'blit'.

- o Modulo et pointeurs sont en octets; la largeur est en mots et la hauteur est en pixels. Le bit le moins significatif de tous les pointeurs et de tous les modulo est ignoré.

- o L'ordre des opérations du 'blitter' est le suivant : masquage, décalage, liaisons logiques des sources, remplissage de surfaces et mise en place du drapeau zéro.

- o En adressage par décrémentation, le 'blitter' augmente les pointeurs, additionne les modulo et décale à droite.

- o En adressage par incrémentation, le 'blitter' diminue les pointeurs, soustrait les modulo et décale à gauche.

- o Le remplissage de surfaces fonctionne correctement en mode adressage par décrémentation.

- o Tester BLTDONE avant d'écrire dans les registres du 'blitter' ou se servir des résultats d'un 'blit'.

- o Les décalages se font sur des données immédiates dès que celle-ci sont chargées.

EXEMPLE : ClearMem

```
;
;   Exemple à propos du 'Blitter' - réinitialisation de mémoire
;

        include 'exec/types.i'
        include 'hardware/custom.i'
        include 'hardware/dmabits.i'
        include 'hardware/blit.i'
        include 'hardware/hw examples.i"

        xref   _custom
;
; Attendre que le 'blit' précédent soit terminé
;
waitblit:
        btst.b #DMAB_BLTDONE-8,DMACONR(a1)
waitblit2:
        btst.b #DMAB_BLTDONE-8,DMACONR(a1)
        bne    waitblit2
        rts
;
; Cette routine utilise un effet de bord du 'blitter'. Lorsque chaque
; 'blit' est fini, le pointeur du 'blitter' pointe sur
; le prochain mot auquel on devra appliquer le 'blit'.
```

```
;
; Quand cette routine prend fin, le dernier 'blit' a commencé et pourrait
; ne pas être fini, assurez-vous donc que vous appelez le waitblit ci-dessus
; avant de présumer que les données sont réinitialisées.
;
; a0   pointeur sur le premier mot à réinitialiser
; d0 - nombre d'octets à réinitialiser (le nombre doit être pair)
;
        xdef   clearmem
clearmem:
        lea    custom,a1              ; Prendre pointeur sur les puces
        bsr    waitblit               ; S'assurer que le blit précédent est fini
        move.l a0,BLTDPT(a1)          ; Mettre en place le pointeur de D sur la zone à réinitialiser
        clr.w  BLTDMOD(a1)            ; Réinitialiser le modulo de D (ne pas sauter d'octets)
        asr.l  #1,d0                  ; Prendre le nombre de mots dans le nombre d'octets
        clr.w  BLTCON1(a1)            ; Pas de mode particulier
        move.w #DEST,BLTCON0(a1)      ; Ne valide que la cible
;
; D'abord nous nous occupons des 'blits' plus petits
;
        moveq  #$3f,d1                ; Masquer les mots modulo 64
        and.w  d0,d1
        beq    dorest                 ; aucun ?  Bien, faire un 'blit'
        sub.l  d1,d0                  ; autrement supprimer le reste
        or.l   #$40,d1                ; mettre la hauteur à 1, la largeur à n
        move.w d1,BLTSIZE(a1)         ; déclencher le 'blit'
;
; Ici, nous nous occupons des mots qui restent, en tant que 'chunks' de
; 128 Ko
;
dorest:
        move.w #$ffc0,d1              ; chercher des bits plus élevés
        and.w  d0,d1                  ; extraire 10 bits supplémentaires
        beq    dorest2                ; il y en a encore ?
        sub.l  d1,d0                  ; extraire ceux dont nous de nous occuper
        bsr    waitblit               ; attendre que le 'blit' précédent soit totalement terminé
        move.w d0,BLTSIZE(a1)         ; faire un autre 'blit'
dorest2:
        swap   d0                     ; encore ?
        beq    done                   ; non.
        clr.w  d1                     ; faire un 'blit' de 1024x64 (128 Ko)
keepon:
        bsr    waitblit               ; terminer ce 'blit'
        move.w d1,BLTSIZE(a1)         ; et à nouveau, 'blit'
        subq.w #1,d0                  ; encore un ?
        bne    keepon                 ; continuer à tourner.
done:
        rts                           ; terminé. Le 'blit' continue à avancer.
        end
```

EXEMPLE : Ligne Simple

```
;
; Cet exemple utilise le mode tracé de ligne du 'blitter'
; pour dessiner une ligne. On dessine une ligne sans modèle
; et on applique un 'blit' 'or' simple dans un seul plan de bits.
;
; Entrées : d0=x1 dl=y1 d2=x2 d3=y2 d4=width a0=aptr
;
        include 'exec/types.i'
        include 'hardware/custom.i'
        include 'hardware/blit.i'
        include 'hardware/dmabits.i'
        include 'hardware/hw_examples.i'
;
        xref    _custom
;
        xdef    simpleline
;
; Notre point d'entrée.
;
simpleline:
        lea     custom,a1               ; mettre en place le registre d'adresse utilisateur
        sub.w   d0,d2                   ; calculer dx
        bmi     xneg                    ; si négatif, l'octant pourra être [3,4,5,6]
        sub.w   d1,d3                   ; calculer l'octant dy, l'octant pourra être [1,2,7,8]
        bmi     yneg                    ; si négatif, l'octant sera [7,8]
        cmp.w   d3,d2                   ; comparer |dx|,|dy| l'octant sera [1,2]
        bmi     ygtx                    ; si y > x, c'est l'octant 2
        moveq.l #OCTANT1+LINEMODE,d5            ; autrement c'est l'octant 1
        bra     lineagain               ; aller à la section commune
ygtx:
        exg     d2,d3                   ; X doit être plus grand que Y
        moveq.l #OCTANT2+LINEMODE,d5            ; nous nous trouvons dans l'octant 2
        bra     lineagain               ; et à nouveau à la section commune
yneg:
        neg.w   d3                      ; calculer valeur absolue de (dy)
        cmp.w   d3,d2                   ; compare |dx|,|dy|, l'octant sera [7,8]
        bmi     ynygtx                  ; si y > x, c'est l'octant 7
        moveq.l #OCTANT8+LINEMODE,d5            ; autrement, c'est l'octant 8
        bra     lineagain
ynygtx:
        exg     d2,d3                   ; X doit être plus grand que Y
        moveq.l #OCTANT7+LINEMODE,d5            ; nous nous trouvons dans l'octant 7
        bra     lineagain
xneg:
        neg.w   d2                      ; dx était négatif !  l'octant pourra être [3,4,5,6]
        sub.w   d1,d3                   ; nous calculons dy
        bmi     xyneg                   ; si négatif, l'octant sera [5,6]
        cmp.w   d3,d2                   ; autrement, l'octant sera [3,4]
        bmi     xnygtx                  ; si y > x, c'est l'octant 3
        moveq.l #OCTANT4+LINEMODE,d5            ; autrement 4
        bra     lineagain
xnygtx:
        exg     d2,d3                   ; X doit être plus grand que Y
        moveq.l #OCTANT3+LINEMODE,d5            ; nous nous trouvons dans l'octant 3
        bra     lineagain
```

```
xyneg:
        neg.w   d3                      ; y était négatif, dans [5,6]
        cmp.w   d3,d2                   ; y > x ?
        bmi     xynygtx                 ; si oui, c'est l'octant 6
        moveq.l #OCTANT5+LINEMODE,d5            ; autrement, c'est l'octant 5
        bra     lineagain
xynygtx:
        exg     d2,d3                   ; X doit être plus grand que Y
        moveq.l #OCTANT6+LINEMODE,d5            ; nous nous trouvons dans l'octant 6
lineagain:
        mulu.w  d9,d1                   ; Calculer y1 * largeur
        ror.l   #4,d0                   ; transférer les quatre bits supérieurs dans le mot de poids fort
        add.w   d0,d0                   ; multiplier par 2
        add.l   d1,a0                   ; ptr += (xl >> 3)
        add.w   d0,a0                   ; ptr += yl * largeur
        swap    d0                      ; prendre les quatre bits de xl
        or.w    #$BFA,d0                ; ou avec VSEA, USEC, USED, F=A+C
        lsl.w   #2,d3                   ; y = 4 * y
        add.w   d2,d2                   ; X = 2 * X
        move.w  d2,d1                   ; mettre en place le mot pour la largeur
        lsl.w   #5,d1                   ; décaler de cinq à gauche
        add.w   #$42,d1                 ; et ajouter 1 à la hauteur, 2 à la
                                        ; largeur
        btst    #DMAB_BLTDONE-8,DMACONR(al)  ; vérification de sécurité
waitblit:
        btst    #DMAB_BLTDONE-8,DMACONR(a1) ; attendre le 'blitter'
        bne     waitblit
        move.w  d3,BLTBMOD(a1)   ; modulo de B = 4 * Y
        sub.w   d2,d3
        ext.l   d3
        move.l  d3,BLTAPT(a1)        ; pointeur de A = 4 * Y - 2 * X
        bpl     lineover                ; si négatif
        or.w    #SIGNFLAG,d5            ; mettre le bit de signe dans conl
lineover:
        move.w  d0,BLTCON0(a1)    ; écrire les registres de contrôle
        move.w  d5,BLTCON1(a1)
        move.w  d4,BLTCMOD(a1)   ; modulo de C = largeur du plan de bits
        move.w  d4,BLTDMOD(a1)   ; modulo de D = largeur du plan de bits
        sub.w   d2,d3
        move.w  d3,BLTAMOD(a1)   ; modulo de A = 4 * Y - 4 * X
        move.w  #$8000,BLTADAT(a1)          ; données de A  = 0x8000
        moveq.l #-1,d5                  ; Mettre en place les masques pour toutes les données
        move.l  d5,BLTAFWM(a1)    ; nous avons la possibilité de mettre deux masques en même temps
        move.l  a0,BLTCPT(a1)         ; Pointeur sur le premier pixel à mettre à 1
        move.l  a0,BLTDPT(a1)
        move.w  d1,BLTSIZE(a1)        ; Démarrer le 'blit'
        rts                             ; et retour, le 'blit' continue à progresser
```

EXEMPLE : Rotation de Bits

```
;
; Ici, nous faisons faire une rotation aux bits. Ce code prend une simple
; ligne de trame d'un plan de bits, et lui applique une rotation dans un
; tableau de mots de 16 bits, en mettant à 1 les bits de chaque mot du
; du tableau, qu'on aura indiqués, en relation avec le bit correspondant
; de la ligne de trame. Nous utilisons le mode ligne avec modèles pour que
; l'effet soit magique.
;
; Entrées : d0 contient le nombre de mots de la ligne de trame. d1
; contient le numéro du bit à mettre à 1 (0..15). a0 contient un
; pointeur sur les données de trame et al contient un pointeur sur le
; tableau que nous remplissons; au minimum, le tableau doit avoir une
; taille de (d0)*16 mots (ou de (d0)*32 octets)).
;
        include 'exec/types.i'
        include 'hardware/custom.i'
        include 'hardware/blit.i'
        include 'hardware/dmabits.i'
        include 'hardware/hw_examples.i'

        xref    _custom
;
        xdef    rotatebits
;
; Notre point d'entrée.
;
rotatebits:
        lea     custom,a2               ; Nous avons besoin d'avoir accès aux registres utilisateur
        tst     d0                      ; s'il n'y a pas de mots, retour
        beq     gone
        lea     DMACONR(a2),a3          ; prendre l'adresse de dmaconr
        moveq.l #DMAB BLTDONE-8,d2      ; prendre le nombre de bits de BLTDONE
        btst    d2,(a3)                 ; tester pour voir si 'done' est mis à 1

waitl:
        tst     d2,(a3)                 ; tester encore.
        bne     wait1                   ; pas de 'done' à 1 ?  Continuer à attendre
        moveq.l #-30,d3                 ; Mode ligne : ptr de A = 4Y-2X, Y=0; X15
        move.l  d3,BLTAPT(a2)
        move.w  #-60,BLTAMOD(a2)        ; modulo de A = 4Y-4X
        clr.w   BLTBMOD(a2)             ; modulo de B = 4Y
        move.w  #2,BLTCMOD(a2)          ; modulo de C = largeur de la mappe de de bits (2)
        move.w  #2,BLTDMOD(a2)          ; idem
        ror.w   #4,d1                   ; prendre les quatre bits de numéro de bits
        and.w   #$f000,d1               ; les masquer
        or.w    #$bca,d1                ; USEA, USEC, USED, F=AB+~AC
        move.w  d1,BLTCON0(a2)          ; les remplir
        move.w  #$f049,BLTCONl(a2)      ; BSH=15, SGN, LINE
        move.w  #$8000,BLTADAT(a2)      ; Initialiser les données de A pour une ligne
        move.w  #$ffffBLTAFWM(a2)       ; Initialiser les masques
        move.w  #$ffff,BLTALWM(a2)
        move.l  a1,BLTCPT(a2)           ; Initialiser le pointeur
        move.l  a1,BLTDPT(a2)
        lea     BLTBDAT(a2),a4          ; Pour un accès rapide, nous prenons ces
        lea     BLTSIZE(a2),a5          ; adresses
        move.w  #$402,d1                ; Remplir bltsize; largeur-2, hauteur 16
        move.w  (a0)+,d3                ; Prendre le prochain mot
```

```
        bra     inloop                  ; Aller dans la boucle
again:
        move.w  (a0)+,d3                ; Prendre un autre mot
        btst    d2,(a3)                 ; Tester si le 'blit' est terminé
wait2:
        btst    d2,(a3)                 ; Tester à nouveau
        bne     wait2                   ; oh !, pas prêt, retour à la boucle
inloop:
        move.w  d3,(a4)                 ; remplir un nouveau mot pour qu'il soit vertical
        move.w  d1,(a5)                 ; déclencher le 'blit'
        subq.w  #1,d0                   ; est-ce le dernier mot ?
        bne     again                   ; continuer si ce n'est pas le cas
gone:
        rts
        end
```

CHAPITRE 7

LE 'HARDWARE' DE CONTROLE DU SYSTEME

INTRODUCTION

Ce chapitre couvre le 'hardware' de contrôle du système Amiga, y compris les sujets suivants :

- o De quelle manière peut-on indiquer les priorités des aires de jeu qui se rapportent aux sprites.
- o De quelle manière tester les collisions entre objets
- o De quelle manière contrôler l'accès direct à la mémoire (DMA)
- o De quelle manière contrôler et tester les interruptions
- o De quelle manière contrôler la réinitialisation, ainsi qu'une mise sous tension rapide.

LES PRIORITES VIDEO

On peut contrôler les priorités des différents objets présents à l'écran pour avoir l'illusion des trois dimensions. La section ci-dessous montre comment on peut modifier la priorité des aires de jeu relative aux sprites.

LES PRIORITES DES SPRITES DEFINIES D'AVANCE

Il n'est pas possible de modifier les priorités respectives des sprites. Ceux-ci s'afficheront toujours à l'écran dans l'ordre suivant : les sprites qui ont un numéro d'ordre plus bas s'affichent devant (ils ont une priorité supérieure à) les sprites qui ont un numéro d'ordre d'ordre plus élevé. Nous montrons ceci dans la Planche 7-1. Chaque carré représente l'image d'un sprite portant le numéro inscrit dans le carré.

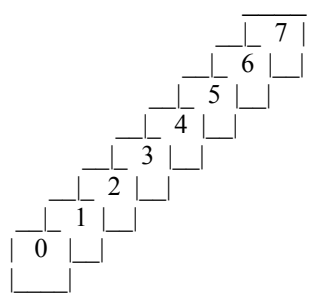


PLANCHE 7-1 : Priorités entre Sprites Fixées d'Avance

COMMENT SONT REGROUPES LES SPRITES

Dans le cas de la collision et de la priorité des aires de jeu seules, les sprites sont traités comme quatre groupes de deux sprites. Ces groupes sont les suivants :

Sprites 0 et 1
Sprites 2 et 3
Sprites 4 et 5
Sprites 6 et 7

COMPRENDRE LES PRIORITES VIDEO

Le concept de priorité vidéo est facile à comprendre si l'on imagine que quatre doigts d'une main représentent les quatre paires de sprites et que les deux doigts de l'autre représentent les deux aires de jeu. Comme il n'est pas possible de modifier la séquence des quatre doigts de la première main, de même il n'est pas possible de modifier la priorité respective des sprites. Cependant, comme on peut entrelacer les deux doigts d'une main de plusieurs façons qui se rapportent aux deux doigts de l'autre main, de même on a la possibilité de placer les aires de jeux devant ou derrière des sprites. Nous illustrons ceci dans la Planche 7-2.

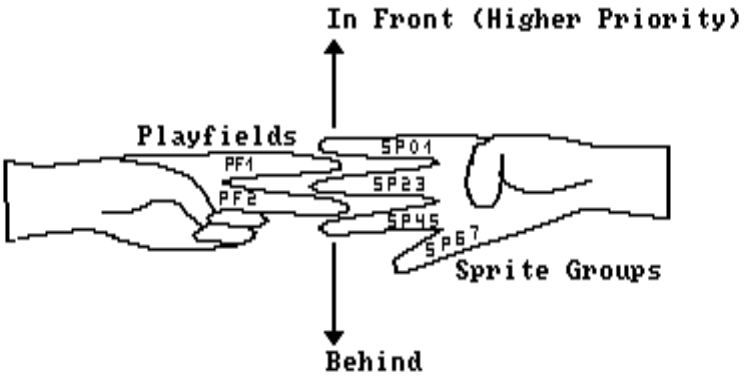


Figure 7-2: Analogy for Video Priority

PLANCHE 7-2 : Analogie en Rapport avec les Priorités Vidéo

Pour chacun des deux "doigts aires de jeu", on peut choisir cinq positions possibles. Par exemple, on peut mettre l'aire de jeu 1 au-dessus des sprites 0 et 1 (0), entre les sprites 0 et 1 et les sprites 2 et (1), entre les sprites 2 et 3 et les sprites 4 et (2), entre les sprites 4 et 5 et les sprites 6 et 7 (3), ou alors au-dessous des sprites 6 et 7 (4). On a les mêmes possibilités avec l'aire de jeu 2.

Les chiffres de 0 à 4 entre parenthèses dans le paragraphe précédent sont les vraies valeurs qu'on utilise pour sélectionner la position des priorités vidéo. Cf., ci-dessous, "Mise en Place du Registre de Contrôle de Priorité".

On a aussi la possibilité de contrôler la priorité de l'aire de jeu 2 par rapport à l'aire de jeu 1. Ceci donne des choix supplémentaires sur la manière dont on peut concevoir les priorités à l'écran.

MISE EN PLACE DU REGISTRE DE CONTROLE DE PRIORITE

Ce registre permet de définir de quelle manière des objets passent l'un devant l'autre ou se cachent l'un sous l'autre. Normalement, l'aire de jeu 1 s'affiche devant l'aire de jeu 2. Le bit PF2PRI inverse cette relation, en donnant plus d'importance à l'aire de jeu 2. On contrôle les priorités vidéo en se servant du bit BPLCON2 (c'est-à-dire, "registre de contrôle du plan de bits numéro 2") comme indiqué dans la Tableau 7-1.

Tableau 7-1 : Bits dans BPLCON2

Bit Numéro	Nom	Fonction
15-7		Inutilisés (les garder à 0)
6	PF2PRI	Priorité de l'aire de jeu 2
5-3	PF2P2 - PF2P0	Mise en place de l'aire de jeu 2 en respectant les sprites
2-0	PF1P2 - PFlP0	Mise en place de l'aire de jeu 1 en respectant les sprites

Les valeurs binaires qu'on donne aux bits PF1P2-PF1P0 déterminent où devra intervenir l'aire de jeu 1 dans la chaîne des priorités, comme le montre le Tableau 7-2. Ceci concorde avec la description donnée dans la section précédente.

NOTE

PF2P2 - PF2P0, bits 5-3, sont les bits de priorité pour les aires de jeu normales (simple, celles qui ne sont pas doubles).

Tableau 7-2 : Priorité des Aires de Jeu Basées sur la Valeur des Bits PF1P2-PF1P0

Valeur	Mise en place (du plus important au moins important)
000	PF1 SP01 SP23 SP45 SP67
001	SP01 PF1 SP23 SP45 SP67
010	SP01 SP23 PF1 SP45 SP67
011	SP01 SP23 SP45 PFl SP67
100	SP01 SP23 SP45 SP67 PFl

Dans ce tableau, PF1 est mis à la place d'aire de jeu 1 et SP01 est mis a la place du groupe de sprites numérotés 0 et 1. SP23 est mis à la place des sprites 2 et 3 en tant que groupe; et SP67 est mis à la place des sprites 6 et 7 en tant que groupe.

Les bits PF2P2-PF2P0 permettent de faire intervenir l'aire de jeu 2 parmi les priorités des sprites exactement de la même manière. Cependant, c'est le bit PF2PRI qui détermine l'aire de jeu qui s'affichera devant l'autre à l'écran. Voici un extrait du contenu possible du registre BPLCON2 qui créerait des choses un peu inhabituelles :

BITS	15-7	PF2PRI	PF2P2-0	PF1P2-0
VALEUR	0s	1	010	000

Ce qui donnerait comme résultat la mise en place d'une priorité aire de jeu / sprite comme :

PF1 SP01 SP23 PF2 SP45 SP67

En d'autres termes, là où les objets passent l'un au-dessus de l'autre, l'aire de jeu 1 se trouvera devant le sprite 0 ou 1; et les sprites de 0 à 3 se trouveront devant l'aire de jeu 2. Toutefois, l'aire de jeu 2 se trouve devant l'aire de jeu 1 dans toute zone où elles se chevauchent et où l'aire de jeu 2 n'est pas bloquée par les sprites de 0 à 3.

La Planche 7-3 montre un emploi de la priorité sprite/aire de jeu. Le seul objet sprite qu'on montre dans le diagramme est le sprite 0. Le sprite peut "voler" dans l'aire de jeu 2, mais, quand il croise l'aire de jeu 1, il disparaît sous celle-ci. Il en résulte un effet vidéo inhabituel provoquant la disparition d'un objet, lorsque celui-ci croise une frontière invisible à l'écran.

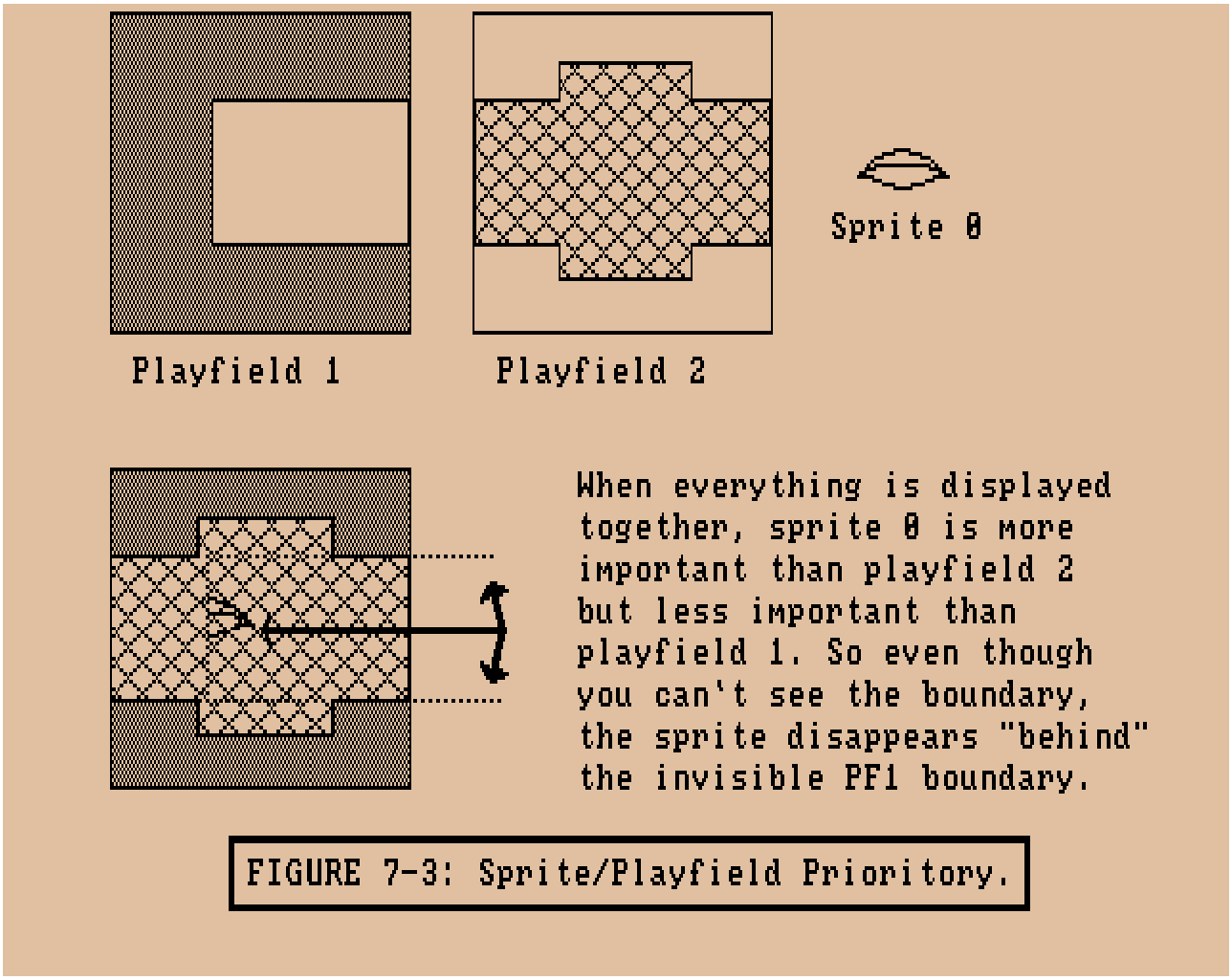


PLANCHE 7-3 : Priorité Sprite/Aire de Jeu

Quand on affiche tout à la fois, le sprite 0 est plus important que l'aire de jeu 2, moins important que l'aire de jeu 1. Par conséquent, même si l'on ne peut pas voir la frontière, le sprite disparaît "au-dessous" de la frontière invisible de PF1.

DETECTION D'UNE COLLISION

On peut se servir du 'hardware' pour détecter des collisions entre un groupe de sprites et un autre groupe, entre un groupe de sprites et l'une des deux aires de jeu, entre les deux aires de jeu ou toute autre combinaison de ces éléments.

Le premier type de collision s'emploie typiquement dans une opération de jeu pour déterminer si un missile est entré en collision avec un joueur qui se déplace. Le deuxième type de collision s'emploie typiquement pour garder un objet qui se déplace à l'intérieur de certaines limites déterminées de l'écran. Le troisième type de collision permet de définir des parties d'une aire de jeu en tant qu'objets individuels, qu'on peut déplacer par le biais du 'blitter'. Ceci s'appelle une animation d'aire de jeu. Si on définit une aire de jeu comme toile de fond ou terrain de jeux et qu'on utilise l'autre aire de jeu pour définir les objets (en plus des sprites), on aura la possibilité de tester des collisions entre les objets-aires de jeu et les sprites ou entre les objets-aires de jeu et la deuxième aire de jeu.

COMMENT AVOIR CONNAISSANCE D'UNE COLLISION

La sortie vidéo se construit quand les données d'entrées en provenance de tous les plans de bits et celles des sprites se réunissent dans un flux de données unique pour l'affichage. Pour chaque position de pixel à l'écran, on affiche la couleur de l'objet qui possède la priorité plus élevée. On détecte des collisions lorsque deux ou plusieurs objets essaient de recouvrir la position d'un même pixel. Cet évènement met à 1 un bit dans le registre de données de collision.

COMMENT INTERPRETER LES DONNEES D'UNE COLLISION

Le registre de données de collision, CLXDAT, est destiné à la seule lecture et son contenu est remis à 0 après avoir été lu. Nous indiquons ses bits dans le Tableau 7-3.

Tableau 7-3 : Bits de CLXDAT

Numéro de Bit	Collisions Enregistrées
15	Inutilisé
14	Sprite 4 (ou 5) avec sprite 6 (ou 7)
13	Sprite 2 (ou 3) avec sprite 6 (ou 7)
12	Sprite 2 (ou 3) avec sprite 4 (ou 5)
11	Sprite 0 (ou 1) avec sprite 6 (ou 7)
10	Sprite 0 (ou 1) avec sprite 4 (ou 5)
9	Sprite 0 (ou 1) avec sprite 2 (ou 3)
8	Plans de bits pairs avec sprite 6 (ou 7)
7	Plans de bits pairs avec sprite 4 (ou 5)
6	Plans de bits pairs avec sprite 2 (ou 3)
5	Plans de bits pairs avec sprite 0 (ou l)
4	Plans de bits impairs avec sprite 6 (ou 7)
3	Plans de bits impairs avec sprite 4 (ou 5)
2	Plans de bits impairs avec sprite 2 (ou 3)
1	Plans de bits impairs avec sprite 0 (ou 1)
0	Plans de bits pairs avec plans de bits impairs

NOTE

Les chiffres entre parenthèses du Tableau 7-3 se réfèrent aux collisions qui seront enregistrées seulement si on veut les afficher. Le registre de contrôle de collision permet soit d'ignorer soit d'inclure les sprites de numéro impair dans la détection de collision.

Remarquez, dans ce tableau, que la détection de collision ne se modifie pas si l'on sélectionne le mode aire de jeu simple ou le mode double aire de jeu. La détection d'une collision ne dépend que des bits réellement présents dans les plans de bits de numéro impair ou pair. Le registre de contrôle de collision indique comment gérer les plans de bits lorsqu'on détecte une collision.

COMMENT EST CONTROLEE LA DETECTION D'UNE COLLISION

Le registre de contrôle de collision, CLXCON, contient des bits qui définissent certaines caractéristiques de détection d'une collision. Nous montrons ces bits dans le Tableau 7-4.

Tableau 7-4 : Bits CLXCON

Numéro du Bit	Nom	Fonction
15	ENSP7	Valider sprite 7 (OU avec sprite 6)
14	ENSP5	Valider sprite 5 (OU avec sprite 4)
13	ENSP3	Valider sprite 3 (OU avec sprite 2)
12	ENSP1	Valider sprite 1 (OU avec sprite 0)
11	ENBP6	Valider plan de bits 6 (concordance requise pour collision)
10	ENBP5	Valider plan de bits 5 (concordance requise pour collision)
9	ENBP4	Valider plan de bits 4 (concordance requise pour collision)
8	ENBP3	Valider plan de bits 3 (concordance requise pour collision)
7	ENBP2	Valider plan de bits 2 (concordance requise pour collision)
6	ENBP1	Valider plan de bits 1 (concordance requise pour collision)
5	MVBP6	Valeur de concordance pour la collision du plan de bits 6
4	MVBP5	Valeur de concordance pour la collision du plan de bits 5
3	MVBP4	Valeur de concordance pour la collision du plan de bits 4
2	MVBP3	Valeur de concordance pour la collision du plan de bits 3
1	MVBP2	Valeur de concordance pour la collision du plan de bits 2
0	MVBP1	Valeur de concordance pour la collision du plan de bits 1

Les bits 15-12 permettent d'indiquer quelles sont les collisions avec une paire de sprites qui doivent inclure le sprite de numéro impair d'une paire de sprites. Les sprites de numéro impair sont toujours inclus dans la détection de collision.
Les bits 11-6 permettent d'indiquer s'il faut inclure ou exclure des plans de bits spécifiques dans la détection de collision.
Les bits 5-0 permettent d'indiquer la polarité des bits (condition vraie-fausse) qui provoqueront une collision. Par exemple, on pourrait vouloir enregistrer des collisions seulement si l'objet entre en collision avec "quelque chose qui soit la couleur verte n.o 2" ou "quelque chose qui soit la couleur bleue". Cette caractéristique, en même temps que les bits de validation de collision, permet d'indiquer des bits bien précis, en même temps que leur polarité, afin d'enregistrer la collision.

NOTE

Ce registre n'est destiné qu'à l'écriture. Si l'on a exclu (invalidé) tous les plans de bits, alors une collision de plans de bits sera toujours détectée.

DETECTION DE LA POSITION DU FAISCEAU

Parfois, on pourrait avoir besoin de synchroniser le processeur 68000 avec le faisceau vidéo qui est en train de construire l'affichage d'écran. Dans certains cas, on pourrait aussi vouloir mettre à jour une partie de la mémoire d'affichage après que le système ait déjà eu accès aux données de la zone d'affichage en mémoire.

Il y a une adresse d'accès au compteur de faisceau qui permet de lire la valeur de son compteur et d'exécuter certaines opérations qui s'appuient sur la position du faisceau.

NOTE

Le Copper a déjà la possibilité de lire la position d'affichage et de faire automatiquement certaines opérations qui s'appuient sur un registre.
Voir la section "Interruptions du Copper", ci-dessous, ainsi que le Chapitre 2, "Le 'Hardware' des Co-processeurs", pour plus de renseignements.

De plus, si l'on utilise un crayon optique avec ce système, on utilise cette même adresse pour lire la position du crayon plutôt que pour lire celle du faisceau vidéo. Nous décrirons ceci dans le détail dans le Chapitre 8, "Le 'Hardware' des Interfaces".

UTILISATION DU COMPTEUR DE POSITION DU FAISCEAU

Il existe quatre adresses d'accès au compteur de position du faisceau. Nous en décrivons l'emploi dans le Tableau 7-5.

Tableau 7-5 : Contenu du Compteur de Position du Faisceau

VPOSR	Seulement en lecture	Lire le bit de poids fort de la position verticale (V8) et le bit de type de trame.
	Bit 15	LOF (Bit de Trame Impaire). Utilisé pour initialiser un affichage entrelacé.
	Bits 14-1	Inutilisés
	Bit 0	Bit de poids fort de la position verticale (V8). Autorise l'affichage du nombre de lignes correct en PAL (313) dans les versions de l'Amiga en PAL.
VHPOSR	Seulement en lecture	Lire la position verticale et horizontale du compteur qui produit le faisceau à l'écran (lit aussi le crayon optique).
	Bits 15-8	Bits de poids faible de la position verticale, bits V7-V0.
	Bits 7-0	Position horizontale, bits H8-H1. La résolution horizontale est égale à 1/160e de la largeur d'écran.
VPOSW	Seulement en écriture	Bits identiques à ceux de VPOSR ci-dessus.
VHPOSW	Seulement en écriture	Bits identiques à ceux de VHPOSR ci-dessus. Utilisés pour la synchronisation du compteur avec les motifs de test des puces.

Comme d'habitude, on peut lire les paires d'adresses VPOSR,VHPOSR et VPOSW, VHPOSW comme des mots longs et y écrire avec des mots longs, les adresses les plus significatives étant VPOSR et VPOSW.

LES INTERRUPTIONS

Ce système accepte intégralement la suite d'interruptions du processeur 68000. Les différents types d'interruptions générées par le 'hardware' sont envoyées aux circuits auxiliaires et sont transformées en six des sept interruptions dont dispose le 68000.

INTERRUPTIONS NON MASQUABLES

L'interruption de niveau 7 est une interruption non masquable et n'est générée nulle part dans le système actuel. Les sorties brutes d'interruption du 68000, de IPL2 à IPL0, sont acheminées vers le connecteur d'expansion et peuvent être employées, pour de raisons de débogage, pour générer des interruptions de niveau 7.

INTERRUPTIONS MASQUABLES

Les interruptions de niveau 1 à 6 sont générées. Les registres de contrôle à l'intérieur des circuits auxiliaires permettent de masquer certaines de ces sources et les empêchent de générer une interruption du 68000.

INTERFACE UTILISATEUR VERS LE SYSTEME D'INTERRUPTIONS

Le 'software' du système a été conçu pour gérer correctement toutes les interruptions 'hardware' du système de niveau 1 à 6. Une suite indépendante de lignes d'entrées, qu'on désigne sous le nom INT2* [1] et INT6* [1], ont été acheminées vers le connecteur d'expansion pour une utilisation des interruptions avec du 'hardware' externe. Ces lignes sont connues comme lignes d'interruption externe de haut niveau et de bas niveau.

Ces lignes d'interruption sont reliées aux circuits auxiliaires et ont la possibilité de créer, respectivement, des interruptions de niveau 2 et de niveau 6. Il est recommandé de profiter des gestionnaires d'interruption présents dans le système d'exploitation lors de l'utilisation de ces lignes d'interruption externes plutôt que de générer des interruptions directement sur les lignes d'interruption du processeur.

REGISTRES DE CONTROLE DES INTERRUPTIONS

Il existe deux registres d'interruption, validation d'interruption (masque) et demande d'interruption (état). Les deux registres possèdent des adresses pour la lecture et pour l'écriture.

Voici les noms des adresses d'interruption

INTENA
: Validation d'interruption (masque) - seulement en écriture. Met à 1 ou à 0 des bits spécifiques d'INTENA.

INTENAR
: Lecture de validation d'interruption (masque) - seulement en lecture. Lit le contenu d'INTENA.

[1] Le signe * indique un signal actif bas.

INTREQ

Demande d'interruption (état) - seulement en écriture. Utilisé par le processeur pour forcer le traitement d'un certain type d'interruptions (interruptions 'software'). Utilisé aussi pour réinitialiser les drapeaux de demande d'interruption une fois terminé le processus d'interruption.

INIEQR

Demande d'interruption (état) - seulement en lecture. Contient les bits qui définissent les articles qui demandent un service d'interruption.

Dans un registre de demande d'interruption la position des bits est l'équivalent direct de ces mêmes positions dans le registre de validation d'interruption. La seule différence existant entre les adresses destinées à la lecture et celles destinées à la lecture -indiquée ci-dessus- est que le bit 15 n'a pas de signification dans les adresses destinées à la lecture.

ACTIVER ET DESACTIVER LES BITS

Ci-dessous, nous expliquons la signification des bits des registres de contrôle d'interruption et comment s'en servir.

ACTIVER ET DESACTIVER

Les registres d'interruption, comme les registres de contrôle des DMA, ont une façon particulière de sélectionner les bits à activer ou à invalider. Le bit 15 de ces registres est le bit SET/CLR.

Si l'on souhaite activer un bit (le mettre à 1), on doit écrire un 1 à la position qu'on veut activer et un 1 à la position 15.

Si l'on souhaite invalider un bit (le mettre à 0), on doit écrire un 0 à la position qu'on veut invalider et un 0 à la position 15.

Les positions 14-0 sont des sélecteurs de bits. On écrit un 1 dans un ou plusieurs bits pour sélectionner un bit ou des bits donnés. En même temps, on écrit un 1 ou un 0 dans le bit 15 selon qu'on veuille activer ou invalider les bits sélectionnés. Les positions 14-0 qui ont une valeur 0 ne seront pas affectées lors de l'écriture. Si on veut mettre à 1 certains bits et en invalider d'autres, on devra écrire deux fois dans le registre (une fois pour activer des bits, une fois pour en invalider d'autres).

VALIDATION PRINCIPALE D'INTERRUPTION

Le bit 14 des registres d'interruption est celui qui valide l'interruption. C'est le bit principal de validation d'interruption. Si ce bit est à 0, il invalide toutes les autres interruptions. On peut vouloir mettre ce bit à 0 pour invalider temporairement toutes les interruptions afin de faire des travaux de traitement particulièrement critiques.

NOTE

Ce bit s'emploie seulement pour valider ou pour invalider. Il ne crée aucune demande d'interruption.

INTERRUPTIONS EXTERNES

Les bits 13 et 3 des registres d'interruption sont réservés aux interruptions externes.

Le bit 13, EXTER, passe à 1 quand la ligne du système INT6* passe à l'état 0 logique. Le bit 13 génère une interruption de niveau 6.

Le bit 3, PORTS, passe à 1 quand la ligne du système INT2* passe à l'état 0 logique. Le bit 3 génère une interruption de niveau 2.

INTERRUPTION DE SUPPRESSION VERTICALE

Le bit 5, VERTB, provoque une interruption à la ligne 0 (début de la suppression verticale) de la trame de l'affichage vidéo. On demande souvent au système d'effectuer plusieurs tâches différentes pendant l'intervalle de suppression verticale. Parmi ces tâches, il y a la remise à jour des différents registres de pointeur, les tâches de réécriture des listes du Copper, si nécessaire, ainsi que d'autres opérations de contrôle du système.

La durée minimale d'une suppression verticale est de 20 lignes de balayage horizontales dans un système NTSC et de 25 lignes dans un système PAL. Cette suite commence à la ligne 0 et se termine à la ligne 20 en NTSC et à la ligne 25 en PAL. Après cette période minimale de suppression verticale, il est possible de contrôler où commence réellement l'affichage en utilisant le registre DIWSTRT (début de la fenêtre d'affichage) pour augmenter de façon concrète la durée de la suppression verticale. Pour plus de renseignements sur DIWSTRT, cf. Chapitre 3, "Le 'Hardware' des Aires de Jeu".

Si on trouve qu'on a besoin d'un temps plus long pendant la suppression verticale, on pourra utiliser le Copper pour créer une interruption de niveau 3. Cette interruption du Copper devrait être chronométrée pour se produire juste après la dernière ligne d'un affichage à l'écran (après le point d'aboutissement de la fenêtre d'affichage qu'on aura défini par le biais du registre DIWSTOP).

INTERRUPTION DU COPPER

Le bit 4, COPER, est employé par le Copper pour faire générer une interruption de niveau 3. Le Copper a la possibilité de modifier le contenu de chacun des bits de ce registre, de même qu'il a la possibilité d'écrire dans la plupart des registres machine. Cependant, ce bit a été réservé pour identifier tout particulièrement le Copper en tant que source d'interruption.

En général, on utilise ce bit quand on veut tester le faisceau d'affichage pour voir s'il a atteint une position d'écran bien déterminée et que l'on souhaite changer en mémoire quelque chose qui s'appuie sur cette occurrence.

INTERRUPTIONS AUDIO

Les bits 10 - 7, AUD3 - 0, sont assignés aux canaux audio. Ils s'appellent AUD3, AUD2, AUD1 et AUD0 et sont assignés, respectivement, aux canaux 3, 2, 1 et 0.

Cette interruption de niveau 4 signale "fin du bloc audio". Si les DMA audio fonctionnent en mode automatique, cette interruption intervient lorsqu'on a eu accès au flux de données audio. En mode manuel, l'interruption se produit lorsque le registre de données audio est prêt à accepter un autre mot de données.

Pour plus de renseignements sur la manière dont est générée l'interruption et sur sa synchronisation, voir le Chapitre 5, "Le 'Hardware' de l'Audio".

INTERRUPTION DU 'BLITTER'

Le bit 6, BLIT, signale que "le 'blitter' a terminé". Si ce bit est à 1, il indique que le 'blitter' a terminé le transfert de données qu'on lui a demandé de faire. Le 'blitter' est maintenant prêt à accepter une autre tâche. Ce bit génère une interruption de niveau 3.

INTERRUPTION LECTEUR DE DISQUETTES

Les bits 12 et 1 des registres d'interruption sont assignés aux interruptions du lecteur de disquettes.

Le bit 12, DSKSYN, indique que le registre de synchronisation a effectué le test de comparaison avec les données de la disquette. Ce bit génère une interruption de niveau 5.

Le bit 1, DSKBLK, indique "fin du bloc disquette". Il est utilisé pour indiquer que la tâche notifiée aux DMA du lecteur de disquette, qu'on a demandée auparavant, est terminée. Ce bit génère une interruption de niveau 1.

Pour plus de renseignements sur le transfert des données de la disquette et sur les interruptions, voir le Chapitre 8, "Le 'Hardware' des Interfaces".

INTERRUPTIONS DU PORT SERIE

Les suivantes interruptions du port série sont associées aux bits des registres d'interruption indiqués.

Le bit 11, RBF (c'est-à-dire, tampon de réception plein), indique que le tampon d'entrée du UART possède des données qu'il est prêt à lire. Ce bit génère une interruption de niveau 5.

Le bit 0, TBE (c'est-à-dire "tampon de transmission vide"), indique que le tampon de sortie du UART a besoin d'autres données et qu'à partir de cet instant on peut écrire des données dans ce tampon. Ce bit génère une interruption de niveau 1.

Priorité 'Hardware'	Description de la Priorité 'Software' d'Exec		Etiquette
1	1	interruption 'software'	SOFTINT
	2	fin du bloc disquette	DSKBLK
	3	tampon de transmission vide	TBE
2	4	INT2 externe et CIAA	PORTS
3	5	co-processeur graphique	COPER
	6	intervalle suppress. vert.	VERTB
	7	fin travail du 'blitter'	BLIT
4	8	canal audio 2	AUD2
	9	canal audio 0	AUD0
	10	canal audio 3	AUD3
	11	canal audio 1	AUD1
5	12	tampon de réception plein	RBF
	13	synchro disquette trouvé	DSKSYNC
6	14	INT6 externe & CIA	EXTER
	15	spécial (validation principale)	INTEN
7	--	interruption non masquable	NMI

Planche 7-4 : Priorité des Interruptions

CONTROLE DES DMA

Plusieurs fonctions différentes d'accès direct mémoire (DMA) se produisent pendant que le système travaille. Il existe une adresse de lecture ainsi qu'une adresse d'écriture dans les registres DMA, par conséquent on peut savoir quels canaux DMA sont validés.

Voici le nom des adresses du registre DMA :

DMACONR - Contrôle de l'Accès Direct Mémoire - seulement en lecture.

DMACON - Contrôle de l'Accès Direct Mémoire - seulement en écriture.

Nous indiquons le contenu du registre dans le Tableau 7-5 (bit activé si validé).

ACCES DU PROCESSEUR A LA MEMOIRE CHIP

Les puces de l'Amiga accèdent directement à la mémoire 'chip', au lieu d'utiliser le mécanisme d'arbitrage traditionnel du bus. Par conséquent, les caractéristiques fournies par le processeur permettant un support multi-processeurs, comme l'instruction TAS ('test and set') du 68000, ne peuvent accomplir la fonction pour laquelle elles ont été prévues et ne sont pas acceptées par l'architecture de l'Amiga.

Tableau 7-6 : Contenu du Registre DMA

Numéro du Bit	Nom	Fonction
15	SET/CLR	Bit de contrôle destiné à la validation et à la réinitialisation. Voir la description du bit 15, dans la sections "Les Interruptions", ci-dessus.
14	BBUSY	Etat d'indisponibilité du 'blitter' - seulement en lecture.
13	BZERO	Etat zéro du 'blitter' - seulement en lecture. Reste à 1, si, pendant ue opération du 'blitter', la sortie de celui-ci est restée toujours à zéro.
12, 11		Inutilisés.
10	BLTPRI	Priorité du 'blitter'. Connu aussi en tant que "blitter-nasty" [bit de danger du 'blitter']. Quand ce bit est à 1, le 'blitter' a une priorité totale (et non partielle) sur le 68000.
9	DMAEN	Validation des DMA. Il s'agit du bit principal de validation des DMA. Il valide les DMA de tous les canaux aux bits 8-0.
8	BPLEN	Validation DMA aire de jeu.
7	COPEN	Validation DMA coprocesseur graphique
6	BLTEN	Validation DMA 'blitter'
5	SPREN	Validation DMA sprites
4	DSKEN	Validation DMA lecteur de disquettes
3-0	AUDxEN	Validation DMA audio pour canaux 3-0 (x = 3 - 0).

Pour plus de renseignements, sur l'utilisation des DMA, voir les chapitres suivants :

Copper	Chapitre 2 "Le 'Hardware' du Coprocesseur""
Bit-planes	Chapitre 3 "Le 'Hardware' des Aires de Jeu"
Sprites	Chapitre 4 "Le 'Hardware' des Sprites"
Audio	Chapitre 5 "Le 'Hardware' Audio"
Blitter	Chapitre 6 "Le 'Hardware' du 'Blitter'"
Disk	Chapitre 8 "Le 'Hardware' des Interfaces"

REINITIALISATION ET OPERATION DE MISE SOUS TENSION RAPIDE

Quand on met sous tension l'Amiga ou on le réinitialise de l'extérieur, la mappe de la mémoire se trouve dans un état particulier. Une copie supplémentaire de la ROM du système répond, en débutant à l'adresse mémoire $00000000. La RAM du système qui, normalement, devrait être en place à cette adresse n'est pas disponible. Sur certains modèles de l'Amiga, des sections de RAM répondent encore. Sur d'autre modèles, il n'y a pas de RAM qui réponde. Le 'software' doit présumer que cette mémoire n'est pas disponible. Le bit OVL des puces 8520 invalide le recouvrement (cf. Appendice F pour l'adresse de ce bit).

La ROM du système de l'Amiga contient un code ID comme premier mot. La valeur du code ID pourra changer à l'avenir. Le deuxième mot de la ROM contient une instruction JMP ($4ef9). Les deux mots suivants sont utilisés comme compteur initial du programme par le processeur 68000.

L'instruction "RESET" du 68000 fonctionne plutôt comme réinitialisation externe ou comme mise sous tension. Toute la mémoire et les cartes AUTOCONFIGTM disparaissent et une image de la ROM apparaît à l'adresse $00000000. La différence réside dans le fait que le CPU continue son exécution avec l'instruction suivante. Étant donné que la RAM ne peut être disponible, il faudra faire très attention en écrivant un code de réinitialisation qui soit capable d'initialiser tous les modèles Amiga de manière effective.

Voici le listage d'un code source du seul code d'initialisation accepté par l'Amiga :

```
; --------- Le *seul* code de réinitialisation accepté ------------
;
        CNOP   0,4      ;IMPORTANT : Doit être divisible par quatre
MagicResetCode:
        lea.l  2,a0     ;Pointeur sur l'instruction JMP au début de la ROM
        RESET           ;Maintenant toute la RAM est libre !
        jmp    (a0)     ;Compter sur l'appel chronologique pour exécuter cette instruction
```

L'instruction RESET doit être exécutée lorsque le CPU se trouve au niveau privilégié du Superviseur. Si on travaille sous Exec, on devra employer le code suivant :

```
_ColdReboot:
        move.l  4,a6                    ;Prendre un pointeur sur ExecBase
        lea.l   MagicResetCode(pc),a5   ;Adresse du code sur lequel on doit se dérouter
        jsr     _LVOSupervisor(a6)      ;Code de démarrage (doit utiliser JSR)
```

CHAPITRE 8

LE 'HARDWARE' DES INTERFACES

INTRODUCTION

Ce chapitre couvre le 'hardware' des interfaces grâce auxquelles l'Amiga communique avec le monde extérieur; nous traitons aussi des caractéristiques suivantes :

- o Deux ports multi-usage pour le contrôle de la souris, des manettes de jeu, du crayon optique
- o Contrôleur de l'unité de disquettes (pour lecteur de disquettes souples et autres périphériques au format MFM et GCR)
- o Clavier
- o Interface parallèle E/S compatible Centronics (pour des imprimantes)
- o Interface série compatible RS232-C (pour des modems externes ou d'autres périphériques série)
- o Connecteurs de sortie vidéo (RVB, monochrome, NTSC, modulateur RF, emplacements vidéo)

INTERFACE DU PORT CONTROLEUR

Chaque Amiga possède deux connecteurs neuf broches qu'on peut employer en entrée et en sortie avec plusieurs types de contrôleurs. La planche ci-dessous montre un des deux connecteurs et la vue de face correspondante d'une fiche de contrôleur typique.

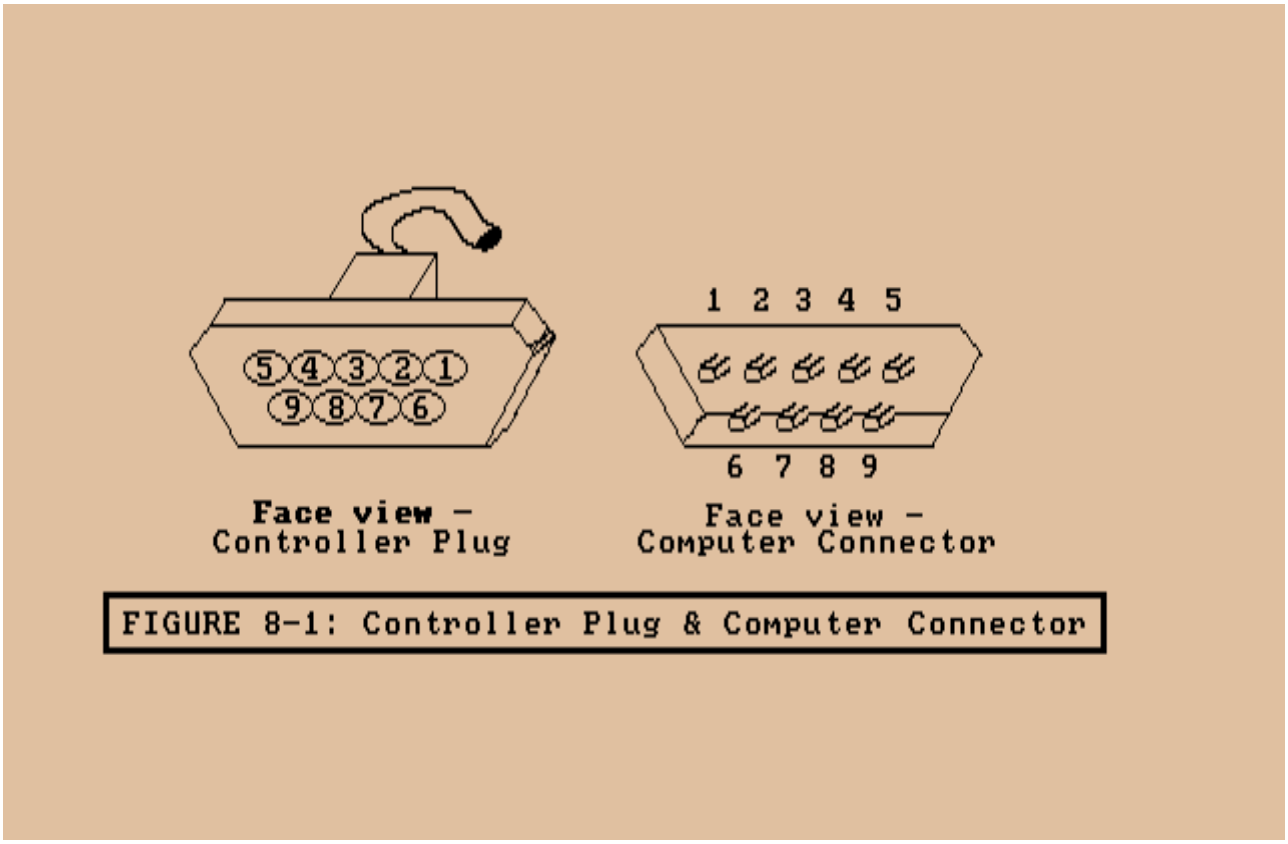


PLANCHE 8-1 : Fiche du Contrôleur et Connecteur de l'Ordinateur

Tableau 8-1 : Connexions Typiques du Contrôleur

Broche	*Joystick*	*Contrôleur de commande Souris, Boule*	*Contrôleur proportionnel (Paire)*	*Joystick proportion. X - Y*	*Crayon Optique*
1	Avant	Impulsion V	---	Bouton 3**	---
2	Arrière	Impulsion H	---	---	---
3	Gauche	Impulsion VQ	Bouton gauche	Bouton 1	---
4	Droite	Impulsion HQ	Bouton droit	Bouton 2	---
5 *	---	Bouton Milieu**	POT droit	POT X	Crayon appuyé sur l'écran
6 *	Bouton 1	Bouton gauche	---	---	Déclencheur de faisceau
7	---	+5V	+5V	+5V	+5V
8	MASSE	MASSE	MASSE	MASSE	MASSE
9 *	Bouton 2**	Bouton droit	POT gauche	POT Y	Bouton 2**

* Ces broches peuvent aussi être configurées en sortie

** Ces boutons sont en option

REGISTRES UTILISES AVEC LE PORT DU CONTROLEUR

JOY0DAT	($DFF00A)	Compteur d'entrée numérique pour souris (port 1)
JOY1DAT	($DFF00C)	Compteur d'entrée numérique pour souris (port 2)
CIAAPRA	($BFE001)	Entrée et sortie pour broche 6 (boutons de mise à feu du port 1 et du port 2)
POT0DAT	($DFF012)	Compteur d'entrée proportionnel (port 1)
POT1DAT	($DFF014)	Compteur d'entrée proportionnel (port 2)
POTGO	($DFF034)	Ecrire les valeurs pour la broche proportionnelle et déclencher les compteurs
POTGOR	($DFF016)	Lire les valeurs pour la broche proportionnelle
BPLCON0	($DFF100)	Le bit 3 valide le verrou du crayon optique
VPOSR	($DFF004)	Lire la position du crayon optique (bits de poids fort)
VHPOSR	($DFF006)	Lire la position du crayon optique (bits de poids faible)

LECTURE DES CONTROLEURS DE SOURIS ET DE BOULE ROULANTE

Les impulsions qui arrivent sur les entrées souris sont converties en valeurs verticale et horizontale séparées. Les registres 8 bits du compteur vertical et horizontal ont la possibilité de vérifier les mouvements de la souris sans l'intervention du processeur.

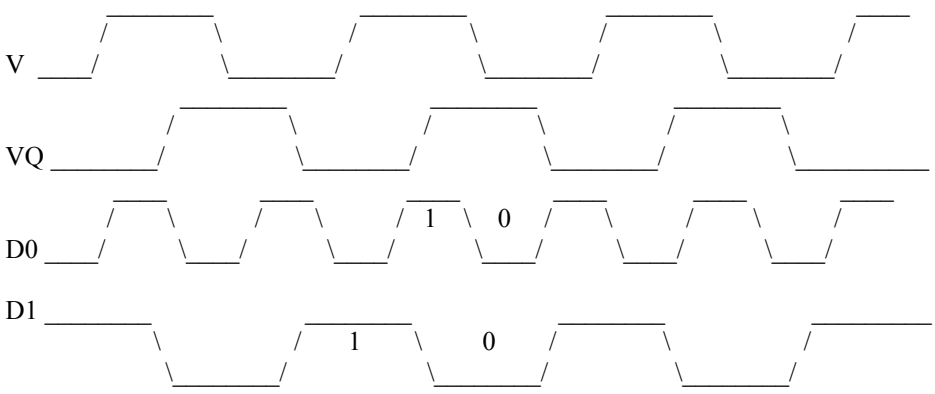
La souris utilise en entrée des impulsions de quadrature. Pour chaque direction, une roue mécanique, à l'intérieur de la souris, produit deux trains d'impulsions, l'un déphasé de 90 degrés par rapport à l'autre (cf. Planche 8-2 pour des détails). La relation entre les phases détermine la direction.

Les compteurs augmentent, si on déplace la souris à droite ou "vers le bas" (vers soi). Les compteurs diminuent, si on déplace la souris à gauche ou "vers le haut" (loin de soi).

IMPULSIONS DE QUADRATURE POUR LA SOURIS

V	VQ	:	D1	D0
0	0	:	1	0
0	1	:	0	1
1	0	:	1	1
1	1	:	0	0

Cas 1 : La valeur du compteur augmente



Cas 2 : La valeur du compteur diminue

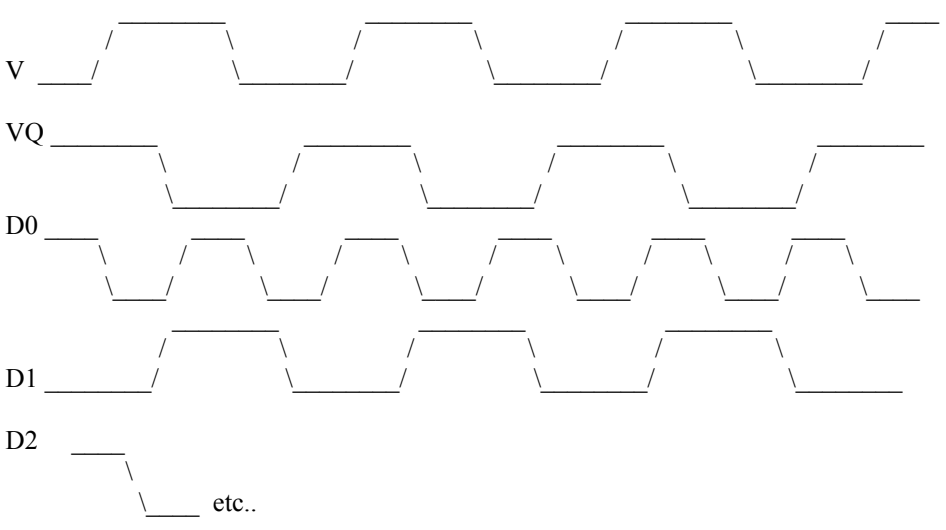


PLANCHE 8-2 : Quadrature pour la Souris

LECTURE DES COMPTEURS

On a la possibilité d'avoir accès au contenu des compteurs en lisant les adresses des registres JOY0DAT et JOY1DAT. Ces adresses contiennent respectivement la valeur des ports 1 et 2.

Le contenu de chacun de ces deux registres 16 bits est le suivant :

Bits 15-8 Valeur verticale souris/boule roulante
Bits 7-0 Valeur horizontale souris/boule roulante

LIMITATIONS DES COMPTEURS

Ces compteurs "rebouclent" en position négative et positive. Si on souhaite employer la souris pour contrôler quelque chose qui va se produire à l'écran, on doit lire les compteurs une fois au minimum à chaque période de suppression verticale et sauvegarder le contenu précédent des registres. Ensuite, on peut faire une soustraction à partir des valeurs qu'on vient de lire pour déterminer la direction du mouvement et la vitesse.

La souris produit un total de 200 impulsions par pouce environ lors d'un déplacement dans une direction horizontale ou verticale. La suppression verticale intervient une fois tous les 1/60èmes de seconde. Si on lit la souris une seule fois, pendant la période de suppression verticale, on trouve très facilement une différence de valeur (par rapport à la valeur précédente) d'au moins 127. Les valeurs du compteur boucleront seulement si l'utilisateur déplace la souris à une vitesse supérieure à 38 pouces par seconde. Les jeux d'action rapides pourraient avoir besoin de lire le registre de la souris deux fois par trame pour prévenir une surcharge du compteur.

Si l'on soustrait la valeur courante de la valeur précédente, la valeur absolue de la différence représentera la vitesse. Le signe de la différence (positif ou négatif) permet de déterminer la direction dans laquelle la souris est en train de se déplacer.

L'emploi de chiffres 8 bits signés est la manière plus facile de calculer la vitesse de la souris. La nouvelle valeur du compteur, moins la valeur précédente, représentera le nombre de pas effectués par le compteur de la souris depuis la dernière vérification. L'exemple que nous donnons dans le Tableau 8-2 présente une méthode alternative. Celle-ci traite les deux valeurs comme des valeurs non signées, s'échelonnant de 0 à 255.
Dans les deux cas, on mesure un total de 100 impulsions.

Tableau 8-2 : Déterminer la Direction de la Souris

Valeur Précédente	*Valeur Courante*	*Valeur de Direction*
200	100	Haut (Gauche)
100	200	Bas (Droite)
200	45	Bas *
45	200	Haut **

NOTES DU TABLEAU 8-1 :

* Comme 200-45 = 155, et que 155 est supérieur à 127, la vraie valeur doit être 255 - (200-45) = 100; la direction est le bas.

** 45-200 = -155. Comme la valeur absolue de -155 est supérieure à 127, la vraie valeur doit être 255 + (-155) = 100; la direction est le haut.

LES BOUTONS DE LA SOURIS

Il y a deux boutons sur la souris standard de l'Amiga. Néanmoins, l'ensemble des circuits de contrôle accepte jusqu'à trois boutons.

o Le bouton gauche de la souris de l'Amiga est relié au CIAAPRA ($BFE001). Le bouton du port 1 est connecté au bit 6, celui du port 2 au bit 7. Pour plus de renseignements, voir l'Appendice qui traite du 8520. Un état logique de 1 signifie "interrupteur ouvert". Un état logique de 0 signifie "interrupteur fermé".

o Le bouton 2 (bouton droit de la souris de l'Amiga) est relié à la broche 9 des ports du contrôleur, la première des deux broches proportionnelles. Pour des détails, voir "Entrée/Sortie Numérique du Port du Contrôleur".

o Le bouton 3, s'il est présent, est relié à la broche 5, l'autre entrée proportionnelle du contrôleur.

LECTURE DES CONTROLEURS NUMERIQUES DE LA MANETTE DE COMMANDE

Les manettes de commande renferment quatre interrupteurs directionnels. Par la manette de contrôle, on a la possibilité d'activer chaque interrupteur chaque interrupteur. Si on appuie sur la manette en diagonale, on active deux interrupteurs adjacents. Dans une manette de commande numérique, la quantité totale de directions possibles est de 8. Toutes les manettes de commande numériques possèdent au moins un bouton de mise à feu.

Les interrupteurs des manettes de commande numériques sont de type normal, ouvert. Quand on appuie sur les interrupteurs, la ligne d'entrée est mise à la masse. Un interrupteur ouvert lit un "1", un interrupteur fermé lit un "0".

Cependant, la lecture des états logiques des données d'entrée de la manette de commande n'est pas si simple car les registres de données des manettes de commande sont les mêmes que ceux qu'on emploie pour les compteurs des contrôleurs de souris et de boule roulante.

Les registres des manettes de commande s'appellent JOY0DAT et JOY1DAT.

Le Tableau 8-2 montre comment interpréter les données de ces registres, après les avoir lues. L'état logique vrai des données de l'interrupteur dans ces registres est égal à "1 = interrupteur fermé".

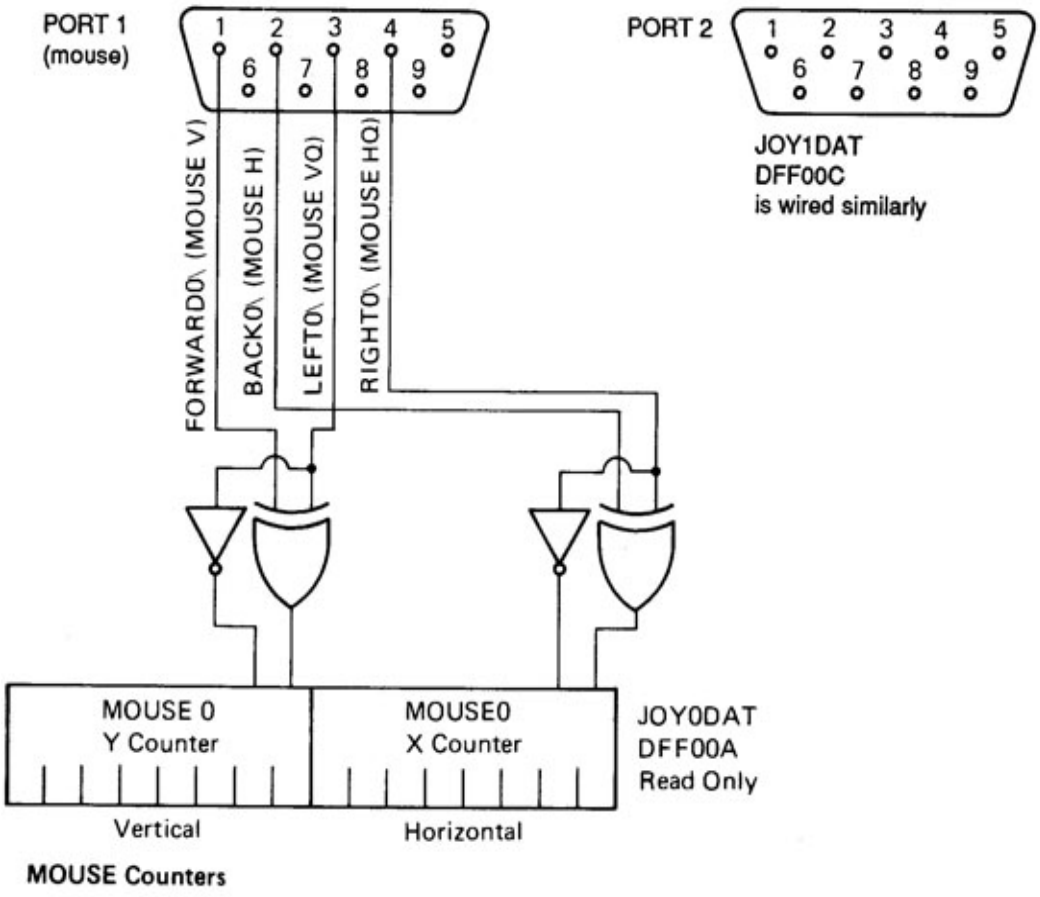


Tableau 8-3: Interprétation des Données dans JOY0DAT et JOY1DAT

Bit de Données	Interprétation
1	Etat logique vrai de l'interrupteur "droit".
9	Etat logique vrai de l'interrupteur "gauche".
1 (XOR) 0	On doit calculer le 'or' exclusif des bits 1 et 0 pour obtenir l'état logique de l'interrupteur "arrière".
9 (XOR) 8	On doit calculer le 'or' exclusif des bits 9 et 8 pour obtenir l'état logique de l'interrupteur "avant".

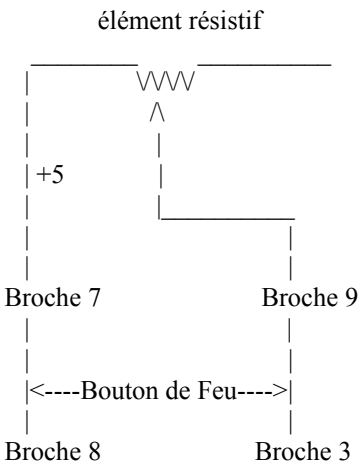
Les boutons de feu des ports 0 et 1 sont connectés aux bits bits 6 et 7 du CIAAPRA ($BFE001). Ici, un 0 indique que l'interrupteur est fermé.

Certaines manettes de commande, mais non toutes, ont un deuxième bouton. Nous encourageons l'emploi de ce bouton, si la fonction contrôlée par le bouton est dédoublée par le biais du clavier ou d'un autre mécanisme. Il est possible de lire ce bouton de la même façon employée pour lire le bouton droit de la souris.

LECTURE DES CONTROLEURS PROPORTIONNELS

Chaque port d'un contrôleur de jeu a la possibilité de gérer deux périphériques à résistance variable. Cette section décrit comment on peut déterminer la position des périphériques d'entrée proportionnels. Il y a deux type courants de contrôleurs proportionnels; la paire de contrôleurs du "manche à balai" et la manette de commande X-Y proportionnelle. La paire de contrôleurs du manche à balai est constituée de deux clôtures individuelles contenant chacune une résistance simple et un bouton de feu, chacune étant connectée à un connecteur d'entrée du port du contrôleur courant. Nous indiquons les connexions typiques dans la Planche 8-3.

MANCHE A BALAI GAUCHE



MANCHE A BALAI DROIT

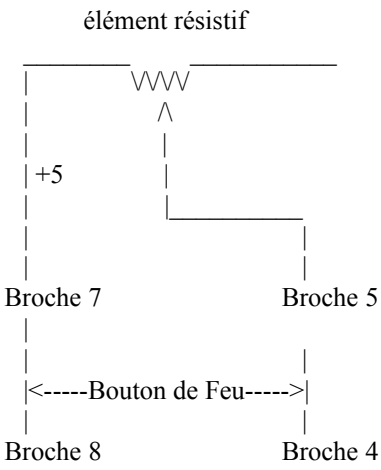


PLANCHE 8-4 : Diagramme de Branchement Typique du Manche à Balai

Dans une manette de commande X-Y proportionnelle, les éléments résistifs sont connectés individuellement aux axes X et Y d'une seule manette du contrôleur.

LECTURE DES BOUTONS DU CONTROLEUR PROPORTIONNEL

Dans les contrôleurs de manches à balai, les lignes de direction gauche et droite de la manette de commande servent de boutons de feu aux manches à balai gauche et droit.

INTERPRETATION DE LA POSITION DU CONTROLEUR

L'interprétation de la position d'un contrôleur proportionnel demande normalement un travail de préparation pendant l'intervalle de suppression verticale.

Pendant cet intervalle, on écrit une valeur dans l'adresse POTGO. pour une manette de commande X-Y standard, cette valeur est de 0001 en hexadécimal. Le fait d'écrire dans ce registre démarre le traitement par l'intermédiaire d'une partie particulière du 'hardware' qui lit les valeurs du potentiomètre et met à zéro les valeurs contenues dans les registres POT (décrits ci-dessous).

L'ensemble des circuits de lecture reste dans un état de réinitialisation pendant les sept ou huit premières lignes du balayage vidéo horizontale. Après cet intervalle de réinitialisation, le circuit permet qu'une charge commence à se former dans le condensateur de synchronisation dont le taux de charge sera contrôlé par la position de la résistance externe du contrôleur. A partir de ce moment, à chaque ligne de balayage horizontale, le circuit compare la charge dans le condensateur de synchro à une valeur initiale. Si la charge se trouve au-dessus de la valeur initiale, la valeur du compteur est conservée jusqu'à ce qu'on fasse apparaître le POTGO suivant.

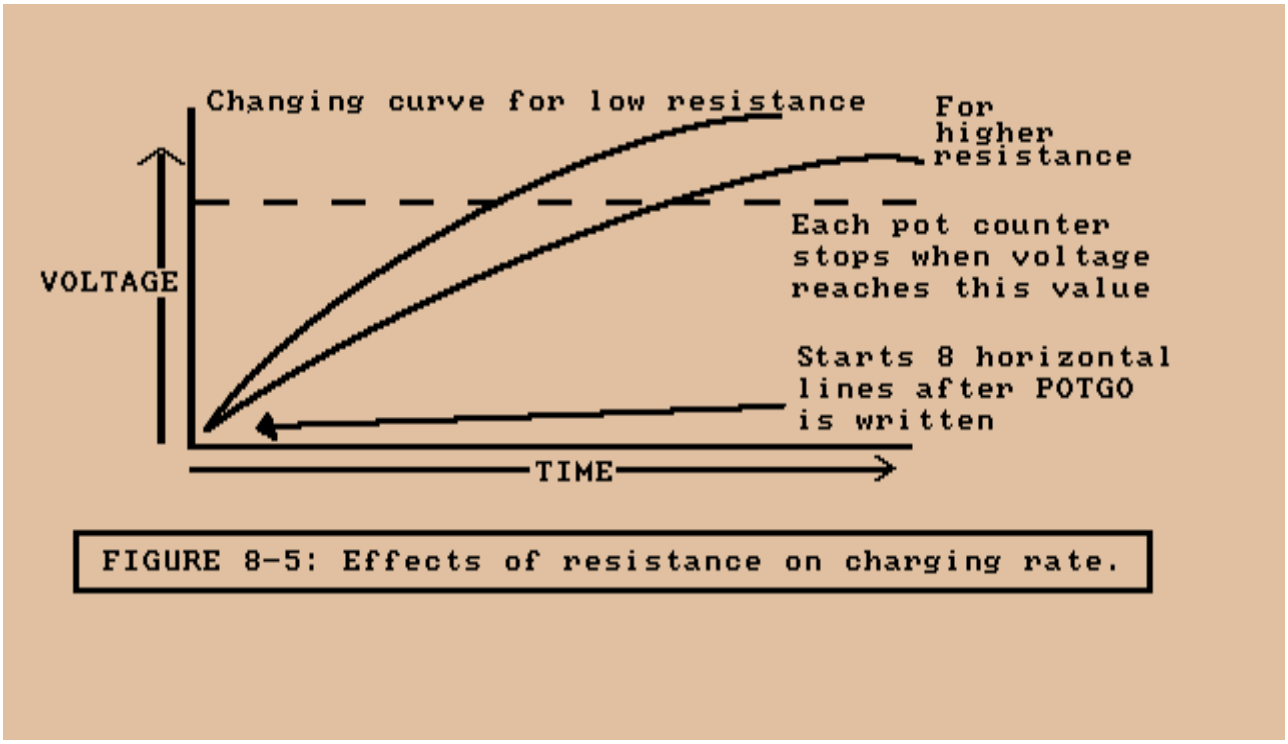


FIGURE 8-5: Effects of resistance on charging rate.

PLANCHE 8-5 : Effets de la Résistance sur le Taux de Charge

Normalement, on fait apparaître POTGO au début d'un écran vidéo, ensuite on lit les valeurs dans les registres POT, pendant la période de suppression verticale suivante, juste avant de faire apparaître un nouveau POTGO.

Rien dans le système protège les compteurs d'un dépassement (le rebouclage se fait après 255). Toutefois, le système est conçu pour que le compteur ne puisse pas se trouver en dépassement de capacité, à l'intérieur d'une plage d'un écran. Ceci permet de savoir de manière certaine si le contrôleur indique un dépassement de capacité.

REGISTRES DU CONTROLEUR PROPORTIONNEL

Les contrôleurs proportionnels emploient les registres suivants :

POT0DAT - données du port 1 (vertical/horizontal)
POT1DAT - données du port 2 (vertical/horizontal)

Positions des bits :

Bits 15-8 valeur POT0Y ou valeur POT1Y
Bits 7-0 valeur POT0X ou valeur POT1X

Toutes les valeurs sont remises à zéro si on écrit dans POTGO avec le bit de poids zéro. On peut lire des valeurs une trame après que l'ensemble des circuits de balayage aient été validés.

SPECIFICATIONS DU POTENTIOMETRE

La résistance des potentiomètres devrait être une résistance linéaire à ruban [1]. D'après la conception du convertisseur analogique/numérique intégré utilisé, sa valeur maximale ne devra pas dépasser 528K (nous conseillons 470K +/- 10 pour cent) pour les deux potentiomètres X et Y. Ceci s'appuie sur un condensateur d'une charge de 0,047 µf, +/- 10 pour cent, pour une durée maximale de charge, à pleine charge, de 16,6 milli-secondes, soit la durée d'une trame vidéo.

Tous les potentiomètres ont un certain "jeu" [2]. Pour obtenir des résultats acceptables, sur une grande base de configurations, il faudra faire la moyenne entre plusieurs lectures de la valeur d'entrée.

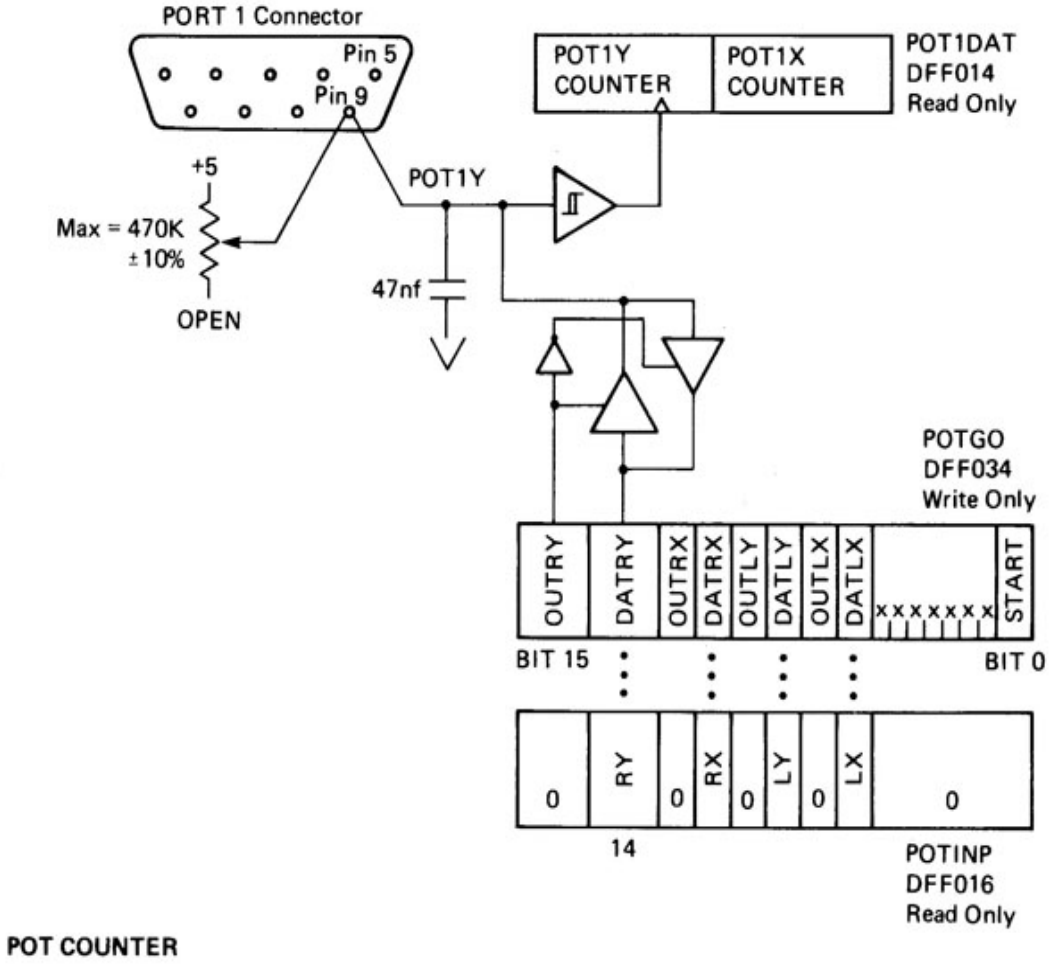


PLANCHE 8-6 : Circuit de Charge du Potentiomètre

[1] J'ai traduit "a linear taper" par résistance linéaire à ruban, mais je ne suis pas certain de ma traduction; en effet, "taper" signifie "cire", "cierge", etc. [NdT].

[2] Le mot "jitter" signifie littéralement "frétillement, gigue"... Je me demande si ma traduction est exacte... [NdT].

LECTURE DU CRAYON OPTIQUE

On a la possibilité de connecter un crayon optique à l'un des ports du contrôleur. Sur l'A1000, on doit connecter le crayon optique au port 1. Si on change de port, on doit faire des modifications internes mineures. Sur l'A500 et sur l'A2000 le port par défaut est le port 2. A l'intérieur de la machine, un cavalier permet de sélectionner le port 1. Quel qu'il soit le port, la conception du crayon optique est identique.

Le signal qu'on appelle "crayon appuyé sur l'écran" est animé typiquement par un interrupteur présent dans le bec du crayon optique. Remarquez que cet interrupteur est relié à l'une des entrées du potentiomètre et qu'on doit le lire comme on lit le bouton droit ou celui du milieu de la souris.

Les principes de fonctionnement du crayon optique sont les suivants :

1. Dès que le système sort de la suppression verticale, l'ensemble des circuits de saisie du crayon optique sont validés automatiquement.

2. Le faisceau vidéo commence à créer l'image, en balayant de gauche à droite, à chaque ligne horizontale, au fur et à mesure qu'il dessine l'image de haut en bas de l'écran.

3. Les capteurs du crayon optique détectent une impulsion de lumière dès que le faisceau vidéo passe. Le crayon convertit l'impulsion de la lumière en une impulsion électrique sur la ligne de "Déclenchement du Faisceau" ['Beam Trigger'] (broche 6).

4. Le signal du déclencheur informe l'ensemble des circuits internes qu'ils doivent saisir et sauvegarder le contenu courant du registre de faisceau, VPOSR. Ceci permet de déterminer l'endroit où on a appuyé le crayon sur l'écran par la lecture des valeurs verticale et horizontale exactes du compteur de faisceau au moment où le faisceau passe au-dessous du crayon.

LECTURE DES REGISTRES DU CRAYON OPTIQUE

Le registre du crayon optique se trouve à la même adresse que les compteurs de faisceau. Voici la signification des bits :

VPOSR :	Bit 15	Trame paire/trame impaire. 0=trame impaire
	Bits 14-1	Code d'ID de la puce. Ne dépend pas de la valeur !
	Bit 0	V8 (bit de position verticale le plus significatif)
VHPOSR :	Bits 15-8	V7-V0 (position verticale)
	Bits 7-0	H8-H1 (position horizontale)

Le 'software' a la possibilité de se référer au registre comme à un mot long dont l'adresse serait VPOSR.

Pour ce qui est de la position, voici la résolution des registres :

Verticale	1 ligne de balayage en mode non entrelacé 2 lignes de balayage en mode entrelacé (par conséquent, si l'on connaît la trame qui se trouve sous l'affichage, on peut déterminer la bonne position)
Horizontale	2 pixels basse résolution en mode haute ou basse définition

La qualité du crayon optique déterminera l'importance du jeu des termes de valeur paire. Dans la plupart des applications, il faudra faire la moyenne de plusieurs lectures.

Pour valider l'entrée du crayon optique, il faut écrire un 1 dans le bit 3 de BPLCON0. Dès qu'on a validé l'entrée du crayon optique et que celui-ci délivre un signal en provenance du déclencheur, la valeur de VPOSR est gelée. Si le déclencheur ne se manifeste pas, les compteurs mémorisent cette valeur à la fin du champ d'affichage. Il n'est pas possible de lire l'emplacement courant du faisceau pendant que le registre VPOSR est verrouillé. Le système débloque le registre à la fin d'une suppression verticale interne (position verticale 20). Dans le système, il n'existe aucun bit pour indiquer qu'il y a eu un déclenchement du crayon optique. Pour déterminer qu'un déclenchement s'est produit, utiliser une des méthodes suivantes :

1. Lire (mot long) deux fois VPOSR.

2. Si les deux valeurs ne sont pas identiques, le crayon optique ne s'est pas déclenché depuis la dernière fois que le faisceau a atteint la partie supérieure de l'écran (V = 20).

3. Si les deux valeurs sont identiques, masquer les 15 bits de poids fort du mot 32 bits et les comparer avec la valeur hexa $10500 (V = 261).

4. Si la valeur de VPOSR est supérieure à $10500, le crayon optique ne s'est pas déclenché depuis la dernière fois que le faisceau a atteint la partie supérieure de l'écran (V = 20) et la position qu'on lit est la position du crayon optique à l'écran.

Une méthode quelque peu simplifiée pour déterminer la valeur véritable du crayon optique implique qu'on informe le 'software' du système pour que celui-ci lise le registre seulement pendant la période de suppression verticale interne 0 < V20 :

1. Lire (mot long) deux fois VPOSR, pendant la période 0 < V20.

2. Masquer les 15 bits de poids fort du mot 32 bits et les comparer avec la valeur hexa $10500 (V = 261).

3. Si la valeur de VPOSR est supérieure à $10500, le crayon optique ne s'est pas déclenché depuis la dernière fois que le faisceau a atteint la partie supérieure de l'écran (V = 20) et la position qu'on lit est la position du crayon optique à l'écran.

Remarquez que, si le verrou du crayon optique est validé, le registre VPOSR peut être verrouillé sans arrêt et qu'on ne pourra lui faire confiance en tant que compteur. Ce type de fonctionnement peut être à l'origine de problèmes avec le 'software' qui essaie de puiser une synchro en s'appuyant sur les impulsions d'horloge de VPOSR.

ENTRE/SORTIE NUMERIQUE SUR LE PORT DU CONTROLEUR

L'Amiga a la possibilité de lire plusieurs contrôleurs non standard de type différent. Les lignes de contrôle présentes dans le registre POTGO (adresse $DFF034) ont la possibilité de redéfinir les fonctions de certaines broches du port du contrôleur.

Le Tableau 8-4 est une description des bits du registre POTGO. L'adresse POTGO ($DFF034) est destinée seulement à l'écriture du registre de contrôle du potentiomètre. L'adresse POTINP ($DFF016) est destinée seulement à la lecture du registre de contrôle du potentiomètre. Ce registre contrôle un port d'E/S bi-directionnel 4 bits qui partage les quatre mêmes broches avec les quatre potentiomètres d'entrée.

Tableau 8-4 : Registres POTGO ($DFF034) et POTINP ($DFF016)

Numéro du Bit	*Nom*	*Fonction*
15	OUTRY	Validation de sortie du bit 14 (1=sortie)
14	DATRY	données du port 2, broche 9
13	OUTRX	Validation de sortie du bit 12
12	DATRX	données du port 2, broche 5
11	OUTLY	Validation de sortie du bit 10
10	DATLY	données du port 1, broche 9 (bouton droit de la souris)
09	OUTLX	Validation de sortie du bit 8
08	DATLX	données du port 1, broche 5 (bouton milieu de la souris)
07-01	X	numéro d'identification de la révision de la puce
00	START	Démarre les potentiomètres (décharger les condensateurs, lancer les compteurs)

Au lieu de se servir des broches du potentiomètre pour avoir des entrées résistives variables, on peut les utiliser comme port d'entrée/sortie quatre bits. Ceci permet d'avoir deux broches supplémentaires sur chacun des deux ports du contrôleur pour des E/S polyvalentes.

Si on valide la sortie en mettant à 1 toutes les broches, l'Amiga déconnecte du port l'ensemble des circuits de contrôle des potentiomètres et configure la broche en sortie. L'état du bit de données contrôle le niveau logique sur la broche de sortie. On doit écrire dans ce registre à l'adresse POTGO et le lire à l'adresse POTINP. Sur ces lignes, il y a des condensateurs de grande capacité et il va falloir jusqu'à 330 micro-secondes pour que l'état de la ligne change.

Pour se servir de tout le registre en tant qu'entrée, en testant l'état courant des broches des potentiomètres, il faut mettre à 0 tout le registre POTGO. A partir de ce moment, on pourra lire l'état courant en utilisant l'adresse POTINP valable seulement en lecture. Remarquez que les bits mis à 1 comme entrées seront connectés aux compteurs proportionnels (cf. la description du bit START de POTGO).

On peut aussi employer ces lignes pour en faire des entrées de boutons. Un bouton est un interrupteur normalement ouvert qui va à la masse.
L'Amiga doit mettre une résistance sur la broche de test. Pour ce faire, mettre la bonne broche en sortie et charger la ligne (mettre à 1 à la fois OUT.. et DAT...).
La lecture de POTINP donnera un 0, si on appuie sur le bouton, un 1, si on n'appuie pas.

Il est possible aussi de configurer en sortie les boutons de mise à feu des manettes de commande. CIAADDRA ($BFE201) contient un masque qui correspond bit par bit au registre de lecture de données, CIAAPRA ($BFE001). Le fait de mettre un 1 dans la position de direction crée une sortie dans le bit correspondant. Pour plus de détails, cf. l'Appendice qui traite du 8520.

CONTROLEUR DU LECTEUR DE DISQUETTES

Le contrôleur du lecteur de disquettes incorporé du système a la possibilité de gérer au plus quatre périphériques de type MFM. Typiquement, ceux-ci sont des lecteurs de disquettes double face, double densité, 3 pouces et demi (90 mm) ou 5 pouces et un quart. Un lecteur de disquettes de 3 pouces et demi est présent dans l'unité de base.

Le contrôleur est extrêmement souple. Il peut avoir accès à la mémoire par les DMA à toute une piste de données brutes au format MFM en un seul tour de disquette. Des registres spéciaux permettent au CPU de se synchroniser avec des données spécifiques ou de lire en entrée un seul octet à la fois. Virtuellement, le contrôleur a la possibilité de lire et d'écrire n'importe quelle disquette double densité codée au format MFM, y compris le format des disquettes Amiga V1.0, les disquettes 5 pouces et un quart des IBM PC (MS-DOS), les disquettes 3 pouces et demi des IBM PC (MS-DOS) et plusieurs autres disquettes au format CP/M (marque enregistrée). Le contrôleur est prévu pour lire et écrire plusieurs types de disquettes utilisant la méthode d'Enregistrement Codé par Groupes ['Group Coded Recording'] (GCR), y compris les disquettes des Apple II. En employant des astuces, pour ce qui est de la vitesse du moteur, le contrôleur peut lire et écrire des disquettes au format Commodore 1541/1571.

REGISTRES UTILISES PAR LE SOUS-SYSTEME DU LECTEUR DE DISQUETTES

Le sous-système du lecteur de disquettes utilise deux ports qui se trouvent dans les puces CIA 8520 du système et différents registres qui se trouvent dans la puce Paula.

CIAAPRA	($BFE001)	quatre bits d'entrée pour test du lecteur de disquettes
CIABPRB	($BFD100)	huit bits de sortie pour sélection, contrôle et pas-à-pas du lecteur de disquettes
ADKCON	($DFF09E)	bits de contrôle (registre destiné seulement à l'écriture)
ADKCONR	($DFF010)	bits de contrôle (registre destiné seulement à la lecture)
DSKPTH	($DFF020)	pointeur DMA (32 bits)
DSKLEN	($DFF024)	longueur pour le DMA
DSKBYTR	($DFFOlA)	octet de données du lecteur de disquette et lecture d'état
DSKSYNC	($DFF07E)	Synchro du lect. de disk; renferme un mot pour le test de comparaison.

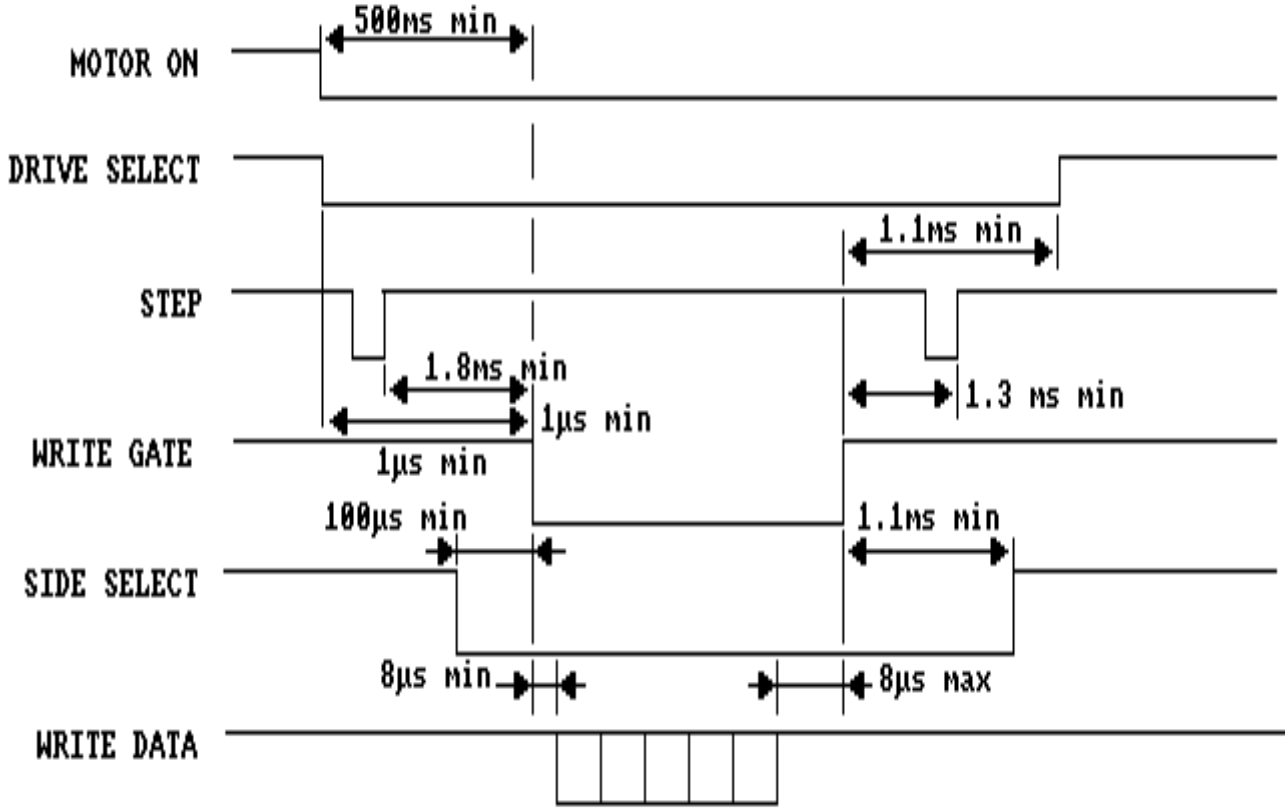


FIGURE 8-7: Chinon Timing Diagram

PLANCHE 8-7 : Diagramme de Synchronisation de Chinon

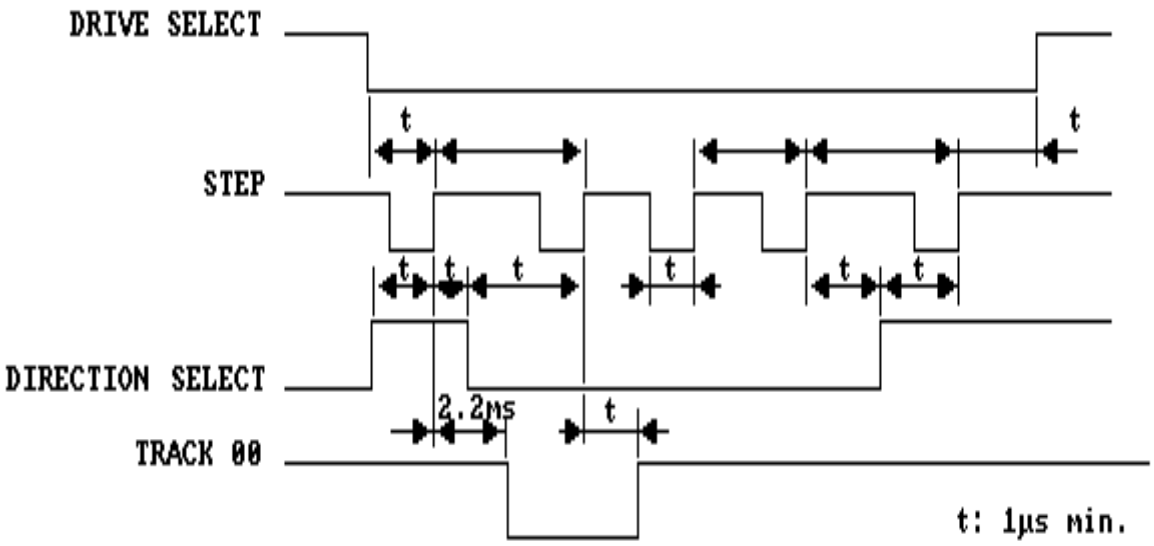


FIGURE 8-8 PART 1: Chinon Timing Diagram (continued)

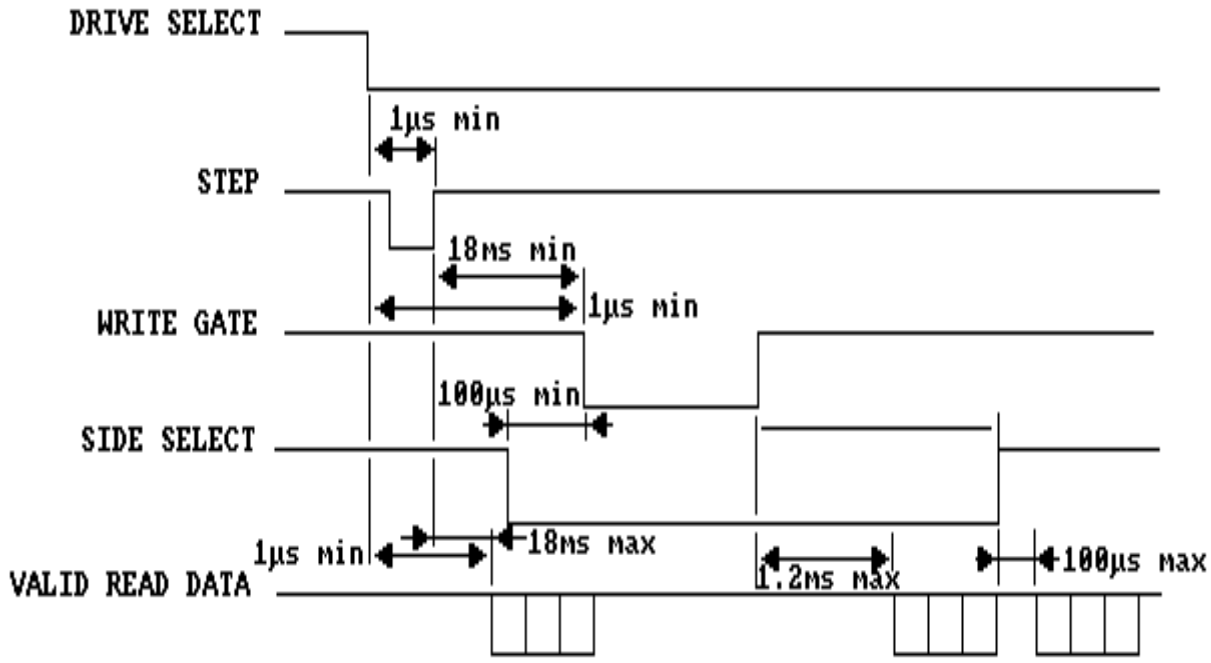


FIGURE 8-8 PART 2: Chinon Timing Diagram (continued)

PLANCHE 8-8 : Diagramme de Synchronisation de Chinon (suite)

Le tableau ci-dessous liste les bits des puces 8520 utilisés par le sous-système du lecteur de disquettes.
Les bits avec l'étiquette "PA" sont des bits d'entrée dans CIAAPRA ($BFE001).
Les bits avec l'étiquette "PB" sont des bits de sortie qui se trouvent dans CIABPRB ($BFD100).
On peut trouver plus de renseignements sur le fonctionnement des puces 8520 dans l'Appendice F.

Tableau 8-5 : Sous-système du Lecteur de Disquettes

Bit	Nom	Fonction
PA5	DSKRDY*	Disquette prête (actif à 'low'). Le lecteur de disquettes mettra cette ligne à 'high' quand il saura que le moteur tourne à pleine vitesse. Ce signal n'est valide que si le moteur est en marche; en d'autres circonstances, l'information concernant la configuration peut masquer la signification de cette entrée.
PA4	DSKTRACK0*	Détection de la piste zéro. Le lecteur de disquettes mettra cette ligne à 'low' quand ses têtes seront positionnées sur la piste zéro. Le 'software' ne doit pas essayer de faire progresser le lecteur vers l'extérieur quand ce signal est actif. Certains lecteurs refuseront d'avancer, d'autres essaieront de le faire, ce qui peut parfois endommager l'alignement. Ceci étant, tous les nouveaux lecteurs devront refuser de progresser vers l'extérieur.
PA3	DSKPROT*	La disquette est protégée en écriture (actif à 'low').
PA2	DSKCHANGE*	On a retiré la disquette du lecteur. Le signal passe à 'low' à chaque fois qu'on retire une disquette. Il reste à 'low' tant qu'on n'a pas inséré une disquette ET l'on reçoit une impulsion d'un pas.
PB7	DSKMOTOR*	Contrôle le moteur de l'unité disquette (actif à 'low'). Dans le système de l'Amiga, ce signal n'est pas standard. Chaque lecteur mémorise le signal du moteur à chaque fois qu'on met en service le signal de sélection. Le moteur du lecteur de disquettes reste dans cet état jusqu'à ce qu'on mette à nouveau en service la sélection. DSKMOTOR* contrôle aussi l'activité de la diode luminescente sur la face avant du lecteur. Tout 'software' qui sélectionne un lecteur doit mettre en marche le signal du moteur avant de le sélectionner. Un lecteur "se souvient" de l'état de son moteur quand il n'est pas sélectionné. Tous les moteurs des lecteurs sont mis hors service après réinitialisation du système. Après avoir mis le moteur en service, le 'software' doit en outre attendre une demi-seconde (500 ms) ou attendre que la ligne DSKRDY* passe à 'low'.

PB6	DSKSEL3*	Sélectionner le lecteur de disquettes 3 (actif à 'low').
PB5	DSKSEL2*	Sélectionner le lecteur de disquettes 2 (actif à 'low').
PB4	DSKSEL1*	Sélectionner le lecteur de disquettes 1 (actif à 'low').
PB3	DSKSEL0*	Sélectionner le lecteur de disquettes 0 (lecteur interne) (actif à 'low').
PB2	DSKSIDE	Indiquer la tête de lecture de l'unité de disquettes qu'il faut employer. Zéro indique la tête supérieure. DSKSIDE doit rester stable pendant 100 micro-secondes avant de commencer à écrire. Après écriture, on devra laisser passer au moins 1,3 milli-secondes avant de commuter DSKSIDE.
PB1	DSKDIREC	Indiquer la direction pour chercher les têtes de lecture. Un zéro laisse supposer que l'on cherche à l'extérieur de l'axe central. La piste zéro se trouve dans la partie extérieure de la disquette. Cette ligne doit être mise en place, avant la véritable impulsion de pas, en écrivant séparément dans le registre.
PB0	DSKSTEP*	Fait avancer les têtes de lecture de l'unité de disquettes. Ce signal doit toujours être utilisé en tant qu'impulsion rapide (haut, momentanément bas, puis haut). On garantit que les lecteurs employés sur l'Amiga sont à même de passer à la piste suivante en 3 milli-secondes. Certains lecteurs accepteront une cadence plus rapide, d'autres échoueront. Les boucles de décrémentation du compteur qui se charge de ce délai ne sont pas adaptées. Cf. l'Appendice F pour trouver une meilleure solution. Lors d'un changement de direction, il faut compter un délai minimum de 18 milli-secondes depuis la dernière impulsion du pas. Le temps nécessaire à la stabilisation des lecteurs de l'Amiga est de 15 milli-secondes.
FLAG	DSKINDEX*	Impulsion d'indice de l'unité de disquettes ($BFDD00, bit 4). On peut s'en servir pour générer une interruption de niveau 6. Pour des détails, voir l'Appendice F.

CONTROLE DES CANAUX DMA DE L'UNITE DE DISQUETTES

Normalement, le système transfère les données vers l'unité de disquettes par un accès direct en mémoire (DMA). Quatre facteurs contrôlent les DMA de l'unité de disquettes :

- o Pointeur sur une zone vers laquelle ou de laquelle on doit déplacer les données.
- o Longueur des données que les DMA doivent déplacer.
- o Direction du transfert des données (lecture/écriture)
- o Validation des DMA.

DSKPTH - POINTEUR SUR LES DONNEES

On indique une adresse d'une taille de 32 bits de laquelle ou vers laquelle on doit transférer les données. Le bit le moins significatif de l'adresse doit être zéro et le tampon doit se trouver en mémoire 'chip'. On doit écrire la valeur de l'adresse en tant que mot long dans le registre DSKPTH ($DFF020).

DSKLEN - LONGUEUR, DIRECTION, VALIDATION DU DMA

Tous les bits de contrôle en rapport avec ce sujet sont renfermés dans un registre destiné seulement à l'écriture, qui s'appelle DSKLEN :

Tableau 8-6 : Registre DSKLEN ($DFF024)

Numéro de Bit	Nom	Emploi
15	DMAEN	Validation secondaire des DMA de l'unité de disquettes
14	WRITE	Ecrire sur la disquette (RAM → Disk si égal à 1)
13-0	LENGTH	Nombre de mots à transférer

Le 'hardware' a besoin d'une séquence particulière pour pouvoir démarrer les DMA en direction de l'unité de disquettes. Cette séquence empêche d'écrire sur la disquette accidentellement. En bref, on devra mettre en service deux fois le bit DMAEN du registre DSKLEN pour pouvoir réellement valider le 'hardware' des DMA de l'unité de disquettes. Voici ce qu'il faudra faire :

1. Valider les DMA de l'unité de disquette du registre DMACON (pour plus de renseignement, voir le Chapitre 7).

2. Mettre DSKLEN à $4000, par ce moyen on force les DMA de l'unite de disquettes à se mettre hors service.

3. Mettre la valeur voulue dans le registre DSKLEN.

4. Ecrire à nouveau cette valeur dans le registre DSKLEN. Ceci fait démarrer les DMA pou de bon.

5. A la fin du DMA, remettre le registre DSKLEN à $4000, pour éviter d'écrire accidentellement sur la disquette.

Au fur et à mesure qu'on transfère des mots de données, la valeur longueur est diminuée d'une unité. Après chaque transfert, la valeur du pointeur est augmentée d'une unité. Le pointeur pointe sur le prochain mot de données à lire ou à écrire. Lorsque la valeur longueur arrive à zéro, le transfert se termine.

La méthode recommandée pour lire à partir d'une disquette est celle qui consiste à lire toute une piste dans le tampon et ensuite à chercher le(s) secteur(s) voulu(s). L'emploi du registre DSKSYNC (décrit ci-dessous) garantit l'alignement des mots de données. Avec ce procédé, on ne devra lire qu'une seule fois toute une piste de disquette. Dans un chargeur à grande vitesse, la progression en direction de la tête de lecture suivante pourra intervenir pendant le traitement et la vérification de la somme de contrôle de la piste précédente. Avec cette méthode, il n'y aura pas de moments critiques pour ce qui est de la durée, lors d'une lecture de données, étant donné que tous les autres sous-systèmes (tels que la partie graphique ou audio) peuvent continuer à fonctionner.

Si l'on n'a pas assez de mémoire pour créer des tampons de pistes (ou que pour une toute autre raison l'on décide de ne pas vouloir lire une piste en entier), le 'hardware' de l'unité de disquettes accepte un ensemble limité d'opportunités pour rechercher des secteurs. Il existe un registre qu'on a la possibilité d'interroger pour examiner le flux d'entrée de l'unité de disquettes.

Il y a un bug 'hardware' qui provoque la perte des trois derniers bits qu'on envoie à l'unité de disquettes. En outre, lors d'une opération des DMA en lecture, le dernier mot peut ne pas être enregistré (ceci parce qu'on ne peut lire qu'un mot de moins par rapport au nombre de mots demandés).

DSKBYTR - OCTET DE DONNEES DU LECTEUR DE DISQUETTE ET LECTURE D'ETAT (SEULEMENT EN LECTURE)

Ce registre est le tampon de données du microprocesseur de l'unité de disquette. En mode lecture, les données en provenance de la disquette sont placées dans ce registre un octet à la fois. Dès que chaque octet arrive dans le registre, le bit DSKBYT est mis à vrai. Ce bit est remis à zéro, si on lit le registre DSKBYTR.

On peut employer DSKBYTR pour synchroniser le processeur à la rotation de la disquette avant d'envoyer un ordre de lecture ou d'écriture sous contrôle des DMA.

Tableau 8-7 : Registre DSKBYTR

Numéro du Bit	Nom	Fonction
15	DSKBYT	S'il est mis à 1, il indique que ce registre renferme un octet de données valide (remis à zéro si on lit le registre).
14	DMAON	Indique si les DMA sont réellement validés. Tous les différents bits des DMA doivent être à vrai. C'est-à-dire le bit de DMAEN de DKSLEN et les bits DSKEN et DMAEN de DMACON.
13	DISKWRITE	Le bit d'écriture (dans DSKLEN) de l'unité de disquette est validé.
12	WORDEQUAL	Indique que le registre DISKSYNC est identique au flux d'entrée de l'unité de disquette. Ce qui est vrai tant que le flux d'entrée correspond au registre de synchronisation (d'une courte durée égale à deux micro-secondes).
11-8		Actuellement inutilisés; ils ne dépendent pas de la valeur de lecture.
7-0	DATA	Données en octets de l'unité de disquette.

ADKCON ET ADKCONR - REGISTRE DE CONTROLE DISQUETTE ET AUDIO

Pour ce registre, ADKCON est l'adresse destinée seulement à l'écriture et ADKCONR est l'adresse destinée seulement à la lecture. Tous les bits ne sont pas réservés exclusivement à l'unité de disquettes. Le bit 15 du registre permet de mettre à 1 ou à 0 n'importe quel bit ou n'importe quel groupe de bits séparément. Si le bit 15 a une valeur de un, lors d'une écriture, tous les bits 0-14 seront mis dans le bit correspondant. Si le bit 15 a une valeur égale à zéro, tous les bits mettrons à zéro le bit correspondant.

Tableau 8-8 : Registres ADKCON et ADKCONR

Numéro du Bit	Nom	Fonction
15	SET/CLR	Bit de contrôle qui permet de mettre à 1 ou à 0 chaque bit individuellement, sans affecter le reste du registre. Si le bit 15 a une valeur de 1, les bits indiqués sont à 1. Si le bit 15 a une valeur de 0, les bits indiqués sont à 0.
14	PRECOMP1	MSB [le bit le plus significatif] de l'indicateur de Pré-compensation
13	PRECOMP0	LSB [le bit le moins significatif] de l'indicateur de Pré-compensation Une valeur de 00 ne sélectionne rien. Une valeur de 01 sélectionne 140 ns. Une valeur de 10 sélectionne 280 ns. Une valeur de 11 sélectionne 560 ns.
12	MFMPREC	Une valeur de 0 sélectionne la Pré-compensation CGR. Une valeur de 1 sélectionne la Pré-compensation MFM.
10	WORDSYNC	Une valeur de 1 valide la synchronisation et le démarrage des DMA lors de la lecture d'un mot sur la disquette. On doit écrire le mot sur lequel on va se synchroniser à l'adresse DSKSYNC ($DFF07E). Cette possibilité est très utile.
9	MSBSYNC	Une valeur de 1 valide la synchronisation sur le bit le plus significatif de l'entrée (utilisé d'habitude pour GCR).
8	FAST	Une valeur de 1 sélectionne deux micro-secondes par élément de bit (d'habitude MFM). Les données doivent être des données brutes valides codées MFM. 0 sélectionne quatre micro-secondes par bit (d'habitude GCR).

Les données brutes codées MFM qu'on doit fournir au contrôleur de l'unité de disquettes auront une taille deux fois plus grande que celle des données non codées. Le tableau qui suit montre cette relation :

```
1 ---> 01
0 ---> 10   ; si données suivante est un 0
0 ---> 00   ; si données suivante est un 1
```

Par une manipulation intelligente, on peut se servir du 'blitter' pour coder et décoder les données MFM.

Dans une forme ordinaire d'enregistrement GCR, le bit le plus significatif de chaque octet de données est toujours égal à 1. MSBSYNC, s'il est égal à 1, informe le contrôleur de l'unité de disquettes qu'il doit chercher ce bit de synchronisation à chaque octet de la disquette. Si on lit une disquette au format CGR, le 'software' doit utiliser une table de conversion qui s'appelle 'nybble-izer' [générateur de demi-octets] pour garantir que les données qu'on écrit sur la disquette n'ont pas trop de 1 ou de 0 qui se suivent.

DSKSYNC - SYNCHRONISATEUR D'ENTREE DE L'UNITE DE DISQUETTES

On utilise le registre DSKSYNC pour synchroniser le flux d'entrée. Celui-ci est très utile lors de la lecture d'une disquette. Si le bit WORDSYNC est activé, dans ADRCON, il n'y aura pas de transfert de données tant qu'on ne rencontrera pas, dans le flux d'entrée, un mot qui corresponde au mot qui se trouve dans le registre DSKSYNC. Lors d'une lecture, le DMA démarre, sur la disquette, sur le mot qui suit. Lors d'un DMA de lecture de disquette, le contrôleur se re-synchronise à chaque fois qu'il trouve une correspondance de mots. Typiquement, on mettra DSKSYNC à la valeur magique de marque de synchronisation MFM, c'est-à-dire à $4489.

En outre, le bit DSKSYNC de INTREQ est mis à 1 quand le flux d'entrée est en correspondance avec le registre DSKSYNC. Le bit DSKSYNC de INTREQ est indépendant de la validation de WORDSYNC.

LES INTERRUPTIONS DE L'UNITE DE DISQUETTES

Le contrôleur de l'unité de disquettes peut générer trois sortes d'interruptions :

- DSKSYNC (niveau 5, bit 12 d'INTREQ) - le flux d'entrée correspond au registre DSKSYNC.
- DSKBLK (niveau 1, bit 1 d'INTREQ) - le DMA de la disquette est totalement fini.
- INDEX (niveau 6, broche drapeau du 8520) - le capteur d'indice a été déclenché.

Les interruptions sont expliquées de façon plus approfondie dans la section "Longueur, Direction, Validation du DMA". Pour plus de renseignements sur les interruptions, voir le Chapitre 7, "Le 'Hardware' de Contrôle du Système". Pour plus de renseignements sur le 8520, cf. l'Appendice F.

LE CLAVIER

Le clavier est interfacé au système par le biais du registre série à décalage de l'une des deux puces CIA 8520. La ligne de données du clavier est reliée à la broche SP, l'horloge du clavier est reliée à la broche CNT. L'Appendice H contient une description détaillée de cette interface.

COMMENT ON RECEPTIONNE LES DONNEES DE CLAVIER

LA ligne CNT est employée en tant qu'horloge du clavier. A chaque transition de cette ligne, un bit de données, en provenance du clavier, sera pointé. Celui-ci envoie une impulsion quand chaque bit de donnée est stable sur la ligne SP. L'impulsion d'horloge est active à 'low'. La partie montante de l'impulsion pointe les données.

Après avoir réceptionné un octet de données en provenance du clavier, le 8520 envoie une interruption en direction du processeur. Le clavier attend un signal 'handshake' [poignée de main] en provenance du système avant de transmettre d'autres frappes de touche. Ce signal est envoyé par le processeur qui rythme la ligne SP alternativement par des signaux 'low' et 'high'. Bien que certains claviers puissent détecter une impulsion de type 'handshake' d'une micro-seconde, cette impulsion devra durer au moins 85 micro-secondes pour pouvoir fonctionner avec tous les modèles de clavier de l'Amiga.

Si l'on réceptionne une autre frappe avant que le processeur n'ait accepté la précédente, le micro-processeur du clavier conserve les frappes dans un tampon de type "frappe continue" pour un total de 10 codes de touches.

LE TYPE DE DONNEES QU'ON RECOIT

Les données de clavier ne sont pas réceptionnées sous forme de caractères ASCII. Au contraire, pour plus de souplesse, elles sont reçues sous forme de codes de touche. Ces codes de touche comprennent les deux transitions touche appuyée/touche relâchée. Ce qui permet au 'software' d'utiliser les deux types d'information pour déterminer exactement ce qui se passe au clavier.

Nous donnons une liste de valeurs hexadécimales qui sont assignées au clavier. Le fait d'appuyer sur une touche transmet cette valeur plus $80. Le croquis du clavier, à la fin de cette section, indique les positions qui correspondent à la description donnée dans le paragraphe ci-dessous.

Remarquez que les codes de touche ne fournissent que des renseignements sur la position de la touche dans le clavier, la légende gravée sur chaque touche changeant d'un pays à l'autre.

CODES DE TOUCHE BRUTS --> 00-3F HEXA

Voici les codes de touche assignés à des positions spécifiques pour la partie principale du clavier. Les lettres qui se trouvent sur les touches changent selon les pays; tous les pays n'emploient pas la disposition de touches QWERTY. Pour ce qui est de la position, ces codes de touche sont mieux décrits dans la Planche 8-9 et dans la Planche 8-10 à la fin de la section qui traite du clavier. Les claviers internationaux possèdent deux touches supplémentaires qu'on a "découpé" à partir des touches plus grandes de la version américaine. Il s'agit de la touche $30, découpée dans la touche 'Shift' gauche et de la touche $2B découpée dans la touche 'Return'.

CODES DE TOUCHE BRUTS --> 40-5F HEXA (CODES COMMUNS A TOUS LES CLAVIERS)

40	Espace
41	Espace arrière
42	Tabulation
43	Touche 'ENTER' du pavé numérique
44	Touche 'RETURN'
45	Echappement [Escape]
46	Effacement [Delete]
4C	Touche curseur fléchée vers le haut
4D	Touche curseur fléchée vers le bas
4E	Touche curseur fléchée à droite
4F	Touche curseur fléchée à gauche
50-59	Touches de fonction F1-F10
5F	Touche d'aide [Help]

CODES DE TOUCHE BRUTS --> 60-67 HEXA (CODES DES TOUCHES QUALIFICATIVES)

60	Touche majuscule droite
61	Touche majuscule gauche
62	Touche verrou pour les majuscules [Caps Lock]
63	Touche de "Contrôle" [Ctrl]
64	Touche ALT gauche
65	Touche ALT droite
66	Touche Amiga gauche (oo touche Commodore)
67	Touche Amiga droite

F0-FF HEXA

Ces codes de touche sont employés par le clavier pour communiquer avec le 68000 et ne sont pas associées à une frappe. Ces touches ne possèdent pas de drapeau de transition et, par conséquent, un code de 8 bits est suffisant à les décrire totalement :

78 Avertissement pou une réinitialisation. On a appuyé sur les touches CTRL-AMIGA-AMIGA. Le clavier attendra 10 secondes au maximum avant de réinitialiser la machine. (N'est pas disponible sur tous les modèles de clavier.)

F9 Dernier code de touche incorrect, on transmettra le même code pour la touche suivante.

FA Dépassement du tampon de touches du clavier.

FC Le test que le clavier fait sur lui-même a échoué. En outre, la diode de la touche 'Caps Lock' clignote pour indiquer la source de l'erreur. La diode clignote une fois pour une défaillance de la ROM, deux fois pour une défaillance de la RAM et trois fois si la minuterie de garde n'a pas fonctionné.

FD Faire démarrer le fonctionnement du flux de touches (pour des touches sur lesquelles on a appuyé ou qu'on a gardé enfoncées à la mise sous tension).

FE Mettre fin au fonctionnement du flux.

D'ordinaire, ces codes de touche sont filtrés par des gestionnaires de clavier.

LIMITATIONS DU CLAVIER

Le clavier de l'Amiga est une matrice de lignes et de colonnes avec un interrupteur de touche à chaque intersection (voir l'Appendice H pour un diagramme de la matrice). A cause de ceci, le clavier est sujet à un phénomène qu'on appelle "les frappes fantômes". Bien qu'en général on n'ait aucun problème lors d'une frappe, les jeux peuvent exiger qu'on appuie indépendamment sur plusieurs touches à la fois. En analysant la matrice, on peut déterminer quels touches peuvent interférer avec d'autres ainsi que les touches qui n'interfèrent jamais.

Les frappes fantôme se produisent si on appuie simultanément sur certaines suites de touches. Par exemple, si l'on appuie simultanément sur les touches "A" et "S". Remarquez que le "A" et le "S" sont transmis à l'écran. Tout en continuant à appuyer sur "A" et "S", appuyez la touche "Z". Avec le clavier original de l'Amiga 1000, on verra s'afficher un "Z" et un "X". Lors de la mise en chantier de l'Amiga 500, on avait amélioré le contrôleur pour qu'il s'aperçoive de situations de frappe fantôme simples comme celle ci-dessus; au lieu de donner naissance à une frappe fantôme, le contrôleur s'empêchait d'envoyer tout caractère tant que le système n'avait pas remis à zéro la matrice (le fait de relâcher la touche "A" ou la touche "S" met à zéro la matrice). Certains claviers Amiga récents implémentent comme vraie "la frappe simultanée de N touches", par laquelle toute combinaison de touches peut être détectée simultanément.

On a conçu tous les claviers pour qu'une frappe fantôme ne se produise pas pendant une frappe normale, mais seulement au cas où l'on appuierait sur des combinaisons de touches inhabituelles, comme celle décrite ci-dessus. Normalement, le clavier semblera pouvoir accepter une frappe simultanée de N touches, c'est-à-dire qu'on n'aura plus de doigts à sa disposition avant de pouvoir donner naissance à une frappe fantôme.

NOTE

Sept touches ne trouvent pas de place dans la matrice et ne contribueront jamais à générer des frappes fantôme. Il s'agit de la touche CIRL, des deux touches SHIFT, des deux touches Amiga et des deux touches ALT.

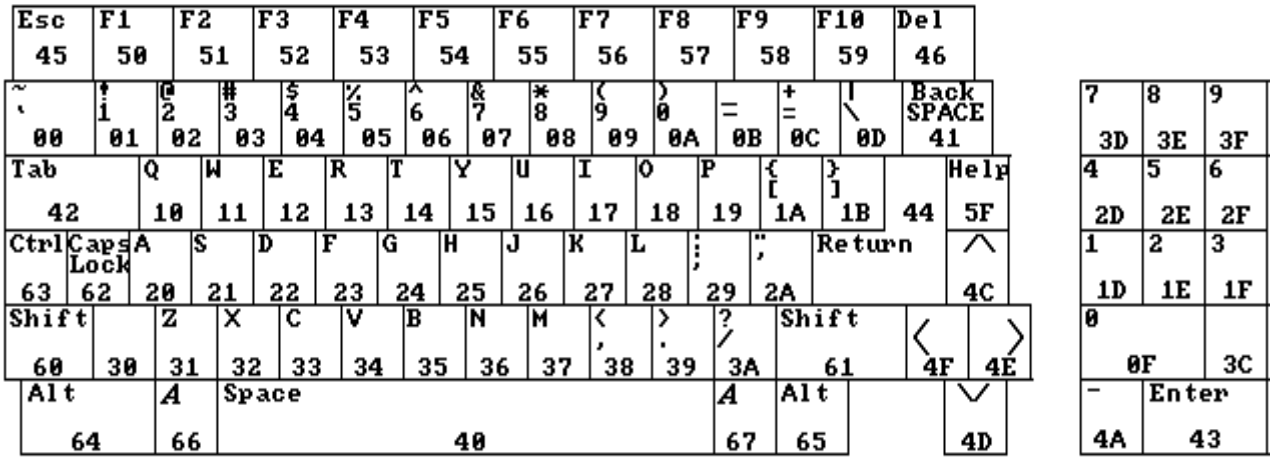


FIGURE 8-9: Amiga 1000 Keyboard, Showing Keycodes in Hex

PLANCHE 8-9 : Le Clavier de l'A1000, où on peut voir le Code des Touches en Hexa

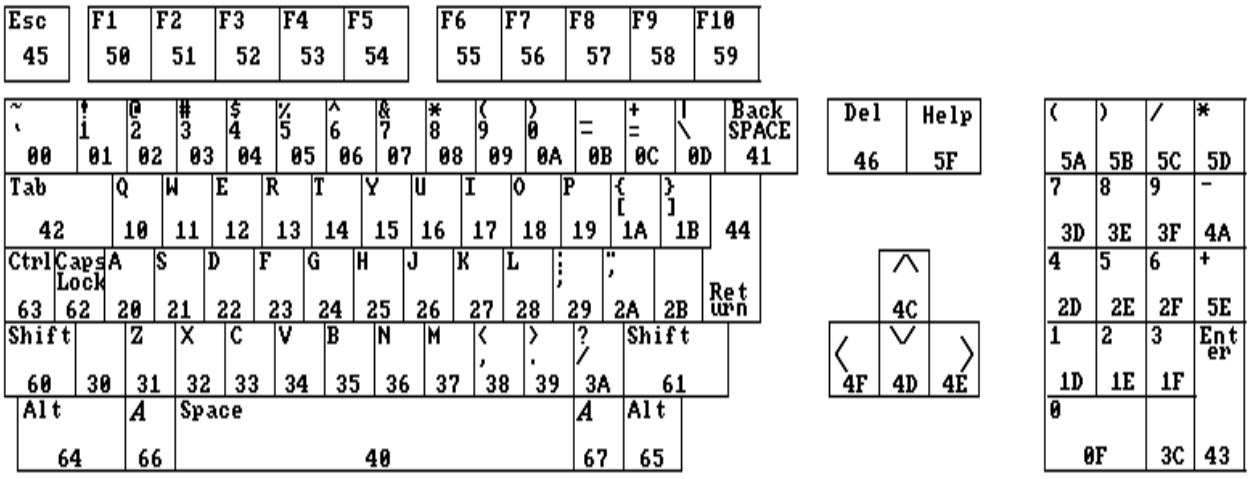


FIGURE 8-10: Amiga 500/2000 Keyboard, Showing Keycodes in Hex

PLANCHE 8-10 : Le Clavier de l'Amiga 500/2000, où on peut voir le Code des Touches en Hexa

INTERFACE D'ENTREES/SORTIE PARALLELE

L'interface parallèle bi-directionnelles polyvalente est un connecteur 25 broches qui se trouve dans la partie arrière de l'ordinateur.
D'habitude, on se sert de ce connecteur avec une imprimante parallèle.

A chaque octet de données écrit dans le registre du port parallèle, le 'hardware' génère automatiquement une impulsion sur la broche de lecture de données. L'impulsion de réception en provenance du périphérique parallèle est relayée par une interruption. Pour la connexion des broches et pour la synchronisation, cf. les Appendices E et F.

INTERFACE SERIE

Un connecteur 25 broches sur le panneau arrière de l'ordinateur sert d'interface série polyvalente. Par le biais du connecteur on a la possibilité de piloter un vaste assortiment de périphériques divers, y compris un modem externe ou une imprimante série.

Pour la connexion des broches, cf. l'Appendice E.

INTRODUCTION A L'ENSEMBLE DES CIRCUITS SERIE

La puce spécialisée Paula renferme un Récepteur/Transmetteur Universel Asynchrone, c'est-à-dire un UART. Ce UART est programmable pour un débit variable pouvant aller de 110 bits à un million de bits par seconde. Il a la possibilité de réceptionner ou d'envoyer des données d'une longueur de huit ou neuf bits programmable.

L'implémentation du UART est pourvue d'un 'software' de contrôle de haut niveau. Le UART a la possibilité de détecter les erreurs d'engorgement qui se produisent lorsqu'un autre système transmet des données plus vite qu'on ne peut les extraire du registre de réception de données. En outre, il existe des bits d'état, ainsi que des interruptions, en cas de tampon de réception plein ou en cas de tampon de transmission vide. Il existe aussi un bit d'état supplémentaire indiquant que "tous les bits ont été décalés". Nous parlerons de tous ces sujets ci-dessous.

MISE EN PLACE DE LA VITESSE DE TRANSMISSION DE L'INFORMATION

La vitesse de transmission ['baud rate'] est contrôlée par le contenu du registre SERPER. Les bits 14-0 de SERPER sont les bits diviseurs de la vitesse de transmission de l'information.

Toute la synchronisation se fait sur la base d'une "impulsion d'horloge couleur", qui est égale à 279,36 nano-secondes sur les machines en NTSC et égale à 281,94 nano-secondes sur les machines en PAL. Si on met le diviseur de SERPER à un chiffre N, se produisent alors N+1 impulsions d'horloge couleur entre des échantillons de l'état de la broche d'entrée (pour une réception) ou entre des envois de bits de sortie (pour une transmission).
Ainsi, on aura SERPER = (3579545/taux)-1 *[taux = taux de transmission ou de réception]*.
Et, sur une machine PAL, on aura SERPER = (3546895/taux)-1 *[taux = taux de transmission ou de réception]*.
Par exemple, la valeur correcte de SERPER, pour 9600 bauds sur une machine NTSC est égale à (3579545/9600)-1 = 371.

Avec un câble d'une longueur raisonnable, la vitesse de transmission maximale de l'information digne de confiance sera de l'ordre de 150000-250000 bits par seconde. La vitesse maximale variera selon les machines. A cette vitesse élevée, il n'est pas possible de gérer le niveau des interruptions. Il faudra que le bout récepteur se trouve dans une boucle de lecture serrée. Par l'emploi d'une information de contrôle lente et par une frappe très rapide, on peut construire un réseau de communication bon marché.

MISE EN PLACE DU MODE RECEPTION

Pour le nombre de bits qu'on doit réceptionner avant que le système nous informe que le registre de réception est plein, on peut indiquer soit huit soit neuf (ce qui permet une transmission 8 bits avec bit de parité). Dans le cas opposé, l'ensemble des circuits de réception s'attendent à voir un bit de démarrage, huit ou neuf bits de données et, pour finir, un bit d'arrêt.

On met en place le mode réception par le bit 15 du registre SERPER, destiné exclusivement à l'écriture. Le bit 15 est égal à un, si l'on choisit des données de huit bits. Dans la plupart des applications, l'état normal du bit est égal à 0.

CONTENU DU REGISTRE DE RECEPTION DES DONNEES

Le registre série de réception de données d'entrée a une taille de 16 bits. Il renferme les données 8 ou 9 bits d'entrée et des bits d'état.

Les données sont réceptionnées, un bit à la fois, dans un registre à décalage interne série/parallèle. Quand le nombre correct de bits de temps se sera écoulé, le contenu du registre sera transféré dans le registre série destiné à la lecture des données (SERDATR), que nous montrons dans le Tableau 8-9, et l'on signalera qu'on trouve ici des données qui sont prêtes.

Dès que le transfert de données s'est mis en place, le registre à décalage de réception devient lui aussi apte à accepter des nouvelles données. Après avoir reçu l'interruption de tampon de réception plein, on aura un maximum d'un caractère de réception de temps entier (de 8 à 10 bits de temps) pour accepter les données et réinitialiser l'interruption. Si l'on ne remet pas à zéro l'interruption à temps, le bit OVERRUN est mis à 1.

Le Tableau 8-9 montre la définition des différentes positions des bits dans SERDATR.

Tableau 8-9 : Registres SERDATR / ADKCON

SERDATR

Numero du Bit	Nom	Fonction
15	OVRUN	bit OVERRUN (Image - se manifeste aussi dans le registre de demande d'interruption.) Indique qu'on a reçu un autre octet de données avant que le processeur n'ait pu prélever l'octet précédent. Pour éviter cette situation, il est nécessaire de réinitialiser INTF_RBF (bit 11, tampon de réception plein) dans INTREQ.
14	RBF	TAMPON DE LECTURE PLEIN (Image - se manifeste aussi dans le registre de demande d'interruption.) Quand ce bit est à 1, il y a des données qui sont prêtes à être prélevées par le processeur. Après avoir lu le contenu de ce registre de données, on doit réinitialiser le bit INTF_RBF de INTREQ pour éviter un engorgement.
13	TBE	TAMPON DE TRANSMISSION VIDE (Ce n'est pas une image - une interruption se produit si le tampon est vide.) Quand le bit 14 est égal à 1, les données du registre de données de sortie (SERDAT) ont été transférées dans le registre à décalage série de sortie, par conséquent, SERDAT est prêt à accepter un autre mot en sortie. Ceci est vrai aussi, si le tampon est vide. D'habitude, on utilise ce bit pour une opération de modem duplex intégral.
12	TSRE	REGISTRE A DECALAGE DE TRANSMISSION VIDE Si ce bit est égal à 1, le registre à décalage de sortie a terminé sa tâche, toutes les données ont été transmises et le registre est à nouveau disponible. Si l'on arrête l'écriture de données dans le registre de sortie (SERDAT), alors ce bit passe à 1, après que le mot actuellement dans le registre à décalage et le mot placé dans SERDAT, aient été tous les deux transmis. D'habitude, on utilise ce bit pour une opération de modem duplex intégral.
11	RXD	Lecture directe de la broche RXD de la puce Paula.
10		Inutilisé pour l'instant.
9	STP	Bit d'arrêt, si l'on a indiqué 9 bits de données pour la réception.

8	STP	Bit d'arrêt, si l'on a indiqué 8 bits de données pour la réception.
		OU
	DB 8	9ème bit de données, si l'on a indiqué 9 bits pour la réception.
7-0	DB7-DB0	Les 8 bits de données de poids faible des données réceptionnées. Les données sont VRAIES (les données qu'on est en train de lire ont la même polarité que les données qu'on attendait).

ADKCON

15	SET/CLR	Permet de mettre à 1 ou à 0 les bits individuellement. Si le bit 15 est égal à 1, les bits indiqués sont mis à 1. Si le bit 15 est égal à 0, les bits indiqués sont mis à 0.
11	UARTBRK	Force la broche de transmission à zéro.

COMMENT SONT TRANSMISES LES DONNEES EN SORTIE

On envoie les données sur les lignes de transmission en écrivant dans le registre série de sortie de données (SERDAT). Ce registre est réservé exclusivement à l'écriture.

Les données seront transmises à la vitesse qu'on a fixée pour leur lecture. Juste après avoir écrit les données dans ce registre, le système commencera la transmission à la vitesse de transmission sélectionnée.

Au début de l'opération, les données de SERDAT sont transférées dans un registre série à décalage interne. Dès que le transfert vers ce registre aura pris fin, SERDAT pourra accepter d'autres données; l'interruption TBE signale cet état.

Les données seront prélevées dans le registre à décalage, à raison d'un bit pour chaque intervalle de temps, en commençant par le bit le moins significatif. Le décalage continue jusqu'à ce que tous les bits égaux à 1 aient été décalés. On peut indiquer de cette manière, n'importe quel chiffre ou combinaison de données, ainsi que les bits d'arrêt.

SERDAT est un registre 16 bits qui permet de contrôler le format (l'aspect) des données transmises. Pour construire une séquence de données typique, telle que celle constituée d'un bit de démarrage, de huit bits de données et d'un bit d'arrêt, il faudra écrire dans SERDAT un contenu comme celui que nous montrons dans les Planches 8-11 et 8-12.

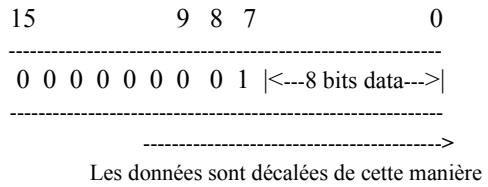


PLANCHE 8-11 : Aspect Initial du Registre SERDAT et du Registre à Décalage

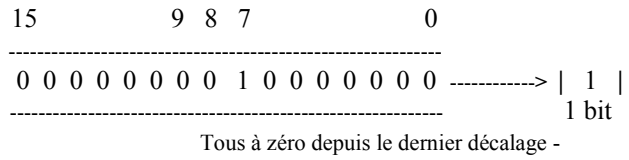


PLANCHE 8-12 : Aspect Final du Registre à Décalage

Le registre arrête le décalage et signale "registre à décalage vide" (TSRE), quand il y a un bit égal à 1 présent à la position du bit de décalage et que le reste du contenu du registre est composé de 0. Dès qu'on charge dans ce registre un nouveau contenu non nul, le décalage recommence.

SPECIFICATION DU CONTENU DU REGISTRE

Les données à transmettre sont placées dans le registre de sortie (SERDAT). En plus des bits de données, on devra ajouter des bits 1 en tant que bits d'arrêt. Normalement, on envoie un ou deux bits d'arrêt.

La transmission du bit de démarrage est indépendante du contenu du registre. Un bit de démarrage est généré automatiquement avant l'envoi du premier bit de données (le bit 0).

Le fait d'écrire dans ce registre fait démarrer la transmission des données. Si on n'écrit que des zéros dans le registre, la transmission des données n'aura pas lieu.

CONNEXIONS DE SORTIE POUR L'AFFICHAGE

Tous les Amiga possèdent à l'arrière un connecteur 23 broches. Celui-ci contient les sorties et les entrées vidéo pour les périphériques externes destinés au 'genlock'. Sur le connecteur, on dispose de deux sorties vidéo séparées:

o Moniteurs RVB ("RVB analogique"). Prévoit quatre sorties; Rouge (R), Vert (V), Bleu (B) et Synchro (S). Celles-ci peuvent générer simultanément à l'écran un maximum de 4096 couleurs différentes en utilisant l'ensemble des circuits actuellement disponibles sur l'Amiga.

o Moniteurs RVB numériques. Prévoit quatre sorties distinctes de celles vues ci-dessus, Rouge (R), Vert (V), Bleu (B), Half-Intensity (I) et Synchro (S). Tous les niveaux de sortie sont des niveaux logiques (0 ou 1). Sur certains moniteurs, ces sorties permettent un maximum de 15 combinaisons de couleurs possibles, où les valeurs 0000 et 0001 sont reliées à la même valeur de sortie (l'intensité 'half' sans couleurs est identique à l'intensité 'full' sans couleurs). D'autres moniteurs relient arbitrairement 16 combinaisons à 16 couleurs choisies de façon arbitraire.

Remarquez que les signaux de synchro en provenance de l'Amiga ne sont pas tamponnés. Pour une utilisation avec un périphérique qui dispose d'une forte charge sur les sorties de synchro, il sera nécessaires de mettre en place des tampons externes.

Les Amiga 500 et 2000 possèdent un jack vidéo monochrome pleine bande pour une utilisation de l'ordinateur avec des moniteurs monochromes meilleur marché. Les couleurs de l'Amiga se combinent en intensité selon le tableau suivant :

Rouge	Vert	Bleu
30%	60%	10%

L'Amiga 1000 possède un jack modulateur RF [radiofréquence]. Un adaptateur est disponible qui permet à l'Amiga d'employer un poste de télévision pour l'affichage. Le son stéréo est disponible sur le jack, mais, en général, dans un poste de télévision, il sera mélangé pour donner à l'écoute un son monophonique.

L'Amiga 1000 possède un jack vidéo pour la couleur composite. Il est adapté à l'enregistrement direct sur magnétoscope, mais la sortie n'est pas de qualité professionnelle. Pour une utilisation avec un moniteur monochrome, l'information couleur produit des effets non souhaités; une sélection soignée des couleurs ou une modification de l'ensemble des circuits internes permet d'améliorer le résultat. Il existe des adaptateurs pour une sortie vidéo composite pour l'A500, l'A1000 et l' A2000 qui se connectent dans le port RVB 23 broches.

L'Amiga 2000 possède un "emplacement vidéo" qui contient plus de signaux qu'on ne pourrait en trouver ailleurs : tous les signaux du port RVB 23 broches, la vidéo numérique non encodée, le crayon optique, l'alimentation, l'audio, les impulsions de synchronisateur couleur, la commutation de pixels, la synchro, les signaux d'horloge, etc..

APPENDICE A

RECAPITULATION DES REGISTRES - ORDRE ALPHABETIQUE

Cette appendice contient la récapitulation définitive en ordre alphabétique
de l'ensemble des registres, ainsi que l'emploi des bits individuels.

Les adresses indiquées ici sont celles qu'utilisent les puces spécialisées (les circuits "Agnus", "Denise" et "Paula") entr'elles pour le transfert des données. Le Copper aussi utilise ces adresses pour écrire dans des registres particuliers de ces puces. Pour écrire dans ces registres avec le 68000, calculer l'adresse pour le 68000 avec la formule suivante :

adresse pour le 68000 = (adresse de la puce) + $DFF000

Par exemple, pour écrire avec le 68000 à l'adresse ADKCON (adresse = $09E), on devra écrire à l'adresse $DFF09E. Aucune autre adresse d'accès ne sera valide. On ne doit pas accéder aux registres inutilisés.

Tous les bits indiqués "inutilisés" devront être mis à zéro. On ne doit pas faire confiance, en lecture, à la valeur d'un bit inutilisé.
Les registres sont destinés seulement à la lecture ou seulement à l'écriture.
Si on lit un registre destiné à l'écriture, celui-ci devient inutilisable.
Le fait d'écrire dans un registre destiné à la lecture provoquera des résultats inattendus.

Tous les registres de type "pointeur" sont organisés sur 32 bits sur la limite d'un mot long. On peut écrire dans ces registres avec une instruction MOVE.L. On doit écrire le bit le moins significatif égal à zéro. Les puces spécialisées ne peuvent avoir accès qu'à la mémoire 'CHIP'; l'emploi d'une adresse qui ne se trouve pas en mémoire 'CHIP' fera échouer l'opération (voir la documentation sur AllocMem() ou le manuel du compilateur pour plus de renseignements sur la mémoire 'CHIP'). Les données pour la disquette, celles des sprites, des plans de bits, les données audio, les listes pour le Copper et toute données sur qui on peut faire un 'blit' ou à qui on peut avoir accès par le biais des canaux DMA devra se trouver en mémoire 'CHIP'.

Si on applique une opération 'strobe' à un registre, qui répondra à une instruction de lecture ou d'écriture, il faut faire attention à utiliser un MOVE.W, et non un CLR.W. Sur un 68000, l'instruction CLR provoque la lecture et la mise à zéro (deux accès), tandis qu'elle ne provoque qu'un simple accès, sur un 68020. Ce qui donnera des résultats différents avec des processeurs différents.

Registre	Adresse	Lecture Ecriture	Agnus/ Denise/ Paula	Fonction
ADKCON	09E	W	P	Contrôle audio/disquette, écriture
ADKCONR	010	R	P	Contrôle audio/disquette, lecture

BIT#	EMPLOI	
15	SET/CLR	Bit de contrôle de mise à 1/mise à 0. Détermine si les bits mis à 1 restent à 1 ou s'ils passent à 0. Les bits à 0 restent inchangés.
14-13	PRECOMP 1-0	

CODE	VALEUR DE PRE-COMP
00	aucune
01	140 nano-sec.
10	280 nano-sec.
11	560 nano-sec.

12	MFMPREC	(1 = pré-compensation MFM; 0 = pré-compensation GCR)
11	UARTBRK	Force un arrêt de l'UART (initialiser TXD), si vrai.
10	WORDSYNC	Valide la synchronisation pour la lecture de la disquette sur un mot égal à DISK SYNC CODE qui se trouve à l'adresse (3F)*2.
09	MSBSYNC	Valide la synchronisation pour la lecture de la disquette sur le MSB (bit le plus significatif). Pour application de type GCR.
08	FAST	Contrôle de la vitesse d'écriture des données sur la disquette : 1-rapide(2µs) 0=lent(4µs). (rapide-->format MFM, lent-->format MFM ou GCR).
07	USE3PN	Emploi du canal audio 3 pour ne rien moduler.
06	USE2P3	Emploi du canal audio 2 pour moduler la période du canal 3.
05	USE1P2	Emploi du canal audio 1 pour moduler la période du canal 2.
04	USE0P1	Emploi du canal audio 0 pour moduler la période du canal 1.
03	USE3VN	Emploi du canal audio 3 pour ne rien moduler.
02	USE2V3	Emploi du canal audio 2 pour moduler le volume du canal 3.
01	USE1V2	Emploi du canal audio 1 pour moduler le volume du canal 2.
00	USE0V1	Emploi du canal audio 0 pour moduler le volume du canal 1.

NOTE : Si on module à la fois la période et le volume sur le même canal, ceux-ci seront alternés. Premier mot xxxxxxxx V6-V0, deuxième mot P15-P0 (et ainsi de suite)

AUDxDAT	0AA	W	P	Canal audio x de données

Ce registre est le tampon de données pour le canal audio x (x=0,1,2,3). Il contient 2 octets de données complémentés à 2 qui sont envoyées en sortie séquentiellement (par conversion numérique/analogique) sur les broches de sortie audio. Le contrôleur DMA transfère automatiquement les données dans ce registre à partir de la mémoire vive. Même le processeur a la possibilité d'écrire directement dans ce registre. Quand il n'y a plus de données (mots sortis= longueur) et que les données du registre ont été utilisées, on envoie une demande d'interruption au canal audio.

AUDxLCH	0A0	W	A	Adresse canal audio x (3 bits de poids fort)
AUDxLCL	0A2	W	A	Adresse canal audio x (15 bits de poids faible)

Cette paire de registres contient l'adresse de départ (emplacement) des données DMA du canal audio x (x=0,1,2, 3). Il ne s'agit pas d'un registre de pointeur et, par conséquent, il faudra le recharger seulement si on doit envoyer en sortie un autre emplacement en mémoire.

AUDxLEN	0A4	W	P	Longueur canal audio x

Ce registre renferme la longueur (nombre de mots) de données DMA d'un canal audio x.

AUDxPER	0A6	W	P	Période canal audio x

Ce registre renferme la période (débit) de transfert des données DMA d'un canal audio x. La valeur de période minimale est égale à 12 signaux d'horloge couleur. Ce qui signifie que le plus petit nombre qu'on peut placer dans ce registre est la valeur décimale 124. Ce qui correspond à la fréquence d'échantillon maximale de 28,86 KHz.

AUDxVOL	0A8	W	P	Volume canal audio x

Ce registre renferme le réglage du volume d'un canal audio x. Les bits 6, 5, 4, 3, 2, 1, 0 indiquent 65 niveaux de volume linéaires comme nous l'indiquons ci-dessous.

BIT#	EMPLOI
15-07	Inutilisés
06	Force le volume au maximum (64 un, aucun zéro)
05-00	Met en place l'un des 64 niveaux (000000=pas de sortie; 111111=63, 63 un, un seul 0)

BLTAFWM	044	W	A	Masque du premier mot de contrôle du 'blitter' pour la source A
BLTALWM	046	W	A	Masque du dernier mot de contrôle du 'blitter' pour la source A

On applique sur les modèles de ces deux registres une opération logique AND avec le premier et le dernier mot de chaque ligne de données de la source A dans le 'blitter'. Pour le mode remplissage ou pour le mode tracé de ligne, on ne devrait mettre que des 1 dans ces registres.

BLTCON0	040	W	A	Registre contrôle 'blitter' 0
BLTCON1	042	W	A	Registre contrôle 'blitter' 1

Ces deux registres de contrôle sont utilisés ensemble pour
contrôler les opérations du 'blitter'.
Il existe deux modes principaux (surface et ligne) qu'on sélectionne
par l'intermédiaire du bit 0 de BLTCON1, comme indiqué ci-dessous.

MODE SURFACE ("normal")

BIT	BLTCON0	BLTCON1
15	ASH3	BSH3
14	ASH2	BSH2
13	ASH1	BSH1
12	ASA0	BSH0
11	USEA	X
10	USEB	X
09	USEC	X
08	USED	X
07	LF7	X
06	LF6	X
05	LF5	X
04	LF4	EFE
03	LF3	IFE
02	LF2	FCI
01	LF1	DESC
00	LF0	LINE(0)

ASH 3-0	Valeur de décalage de la source A
BSH 3-0	Valeur de décalage de la source B
USEA	Bit de contrôle de mode pour utiliser la source A
USEB	Bit de contrôle de mode pour utiliser la source B
USEC	Bit de contrôle de mode pour utiliser la source C
USED	Bit de contrôle de mode pour utiliser la cible D
LF 7-0	Lignes de sélection du 'minterm' de fonction logique
EFE	Validation remplissage exclusif
IFE	Validation remplissage inclusif
FCI	Entrée de retenue pour le remplissage
DESC	Bit de contrôle pour le mode décrémentation (adresse décroissante)
LINE	Bit de contrôle pour le mode ligne (mis à 0)

BLTCON0 (suite) TRACE LIGNE
BLTCON1 (suite) TRACE LIGNE

TRACE LIGNE MODE LIGNE (tracé de ligne)

BIT#	BLTCON0	BLTCON1
15	START3	TEXTURE3
14	START2	TEXTURE2
13	START1	TEXTURE1
12	START0	TEXTURE0
11	1	0
10	0	0
09	1	0
08	1	0
07	LF7	0
06	LF6	SIGN
05	LF5	0 (Inutilisé)
04	LF4	SUD
03	LF3	SUL
02	LF2	AUL
01	LF1	SING
00	LF0	LINE(=1)

TRACE LIGNE

START 3-0 Point de départ de ligne (de 0 à 15, en hexa)
LF7-0 Pour sélectionner l'équation D=(AC+ABC),
on devrait pré-charger les lignes de sélection de
fonction logique du 'minterm' avec 4A.
Comme A contient un seul bit vrai (8000),
la plupart des bits passeront dans le champ C
sans être modifiés (not A and C), mais un
bit inversera le champ de C et se combinera
avec la texture (A and B and not C).
Le bit de A sera déplacé automatiquement
dans le mot par le 'hardware'.

TRACE LIGNE

LINE	Bit de contrôle du mode ligne (mis à 1)
SIGN	Drapeau pour le signe
0	Réservé pour un mode ultérieur
SING	Un seul bit par ligne horizontale à utiliser pour le remplissage de la surface suivante
SUD	Parfois en haut ou en bas (=AUD*)
SUL	Parfois en haut ou à gauche
AUL	Toujours en haut ou à gauche

Les trois bits ci-dessous sélectionnent
l'octant pour dessiner une ligne :

OCT	SUD	SUL	AUL
0	1	1	0
1	0	0	1
2	0	1	1
3	1	1	1
4	1	0	1
5	0	1	0
6	0	0	0
7	1	0	0

On emploie la source "B" pour la texture des
lignes dessinées.

BLTDDAT -- -- -- Registre de données cible du 'blitter'

Ce registre renferme les données résultantes en provenance de chaque mot d'opération du 'blitter' jusqu'au moment où ces données sont envoyées à la zone cible en mémoire vive. Il s'agit d'une adresse bidon que le processeur n'a pas la possibilité de lire. Le transfert se fait automatiquement pendant le traitement du 'blitter'.

BLTSIZE 058 W A Démarrage du 'blitter' et taille (largeur, hauteur de la fenêtre)

Ce registre contient la largeur et la hauteur de l'opération du 'blitter' (en mode ligne, la largeur devra être = 2, la hauteur = longueur de la ligne). Le fait d'écrire dans ce registre fera démarrer le 'blitter', par conséquent, on devra y écrire en dernier, après avoir initialisé tous les pointeurs et les registres de contrôle.

BIT 15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00
---
h9 h8 h7 h6 h5 h4 h3 h2 h1 h0, l5 l4 l3 l2 l1 l0

h=hauteur=lignes verticales (10 bits=1024 lignes au maxi)
l=largeur=pixels horizontaux (6 bits=64 mots=1024 pixels maxi)

TRACE LIGNE BLTSIZE contrôle la longueur de la ligne et lance le tracé de ligne quand on y écrit. Le champ h contrôle la longueur de la ligne (10 bits donnent des lignes d'une longueur maxi de 1024 pixels). Le champ l doit être mis à 02 pour tous les tracés de ligne.

BLTxDAT 074 W A Registre de données du 'blitter' de la source x

Ce registre renferme les données de la zone source x (x=A,B,C), données qui sont utilisées par le 'blitter'. Normalement, ce registre est chargé par le canal DMA du 'blitter'; par conséquent, il peut être aussi pré-chargé par le micro-processeur.

TRACE LIGNE BLTADAT est utilisé comme registre d'indice et on devra le pré-charger avec 8000. BLTBDAT est utilisé pour la texture; on devra le charger avec FF, si l'on ne souhaite pas avoir de texture (pleine ligne).

BLTxMOD	064	W	A	Modulo x du 'blitter'

Ce registre renferme le modulo pour la source (x=A,B,C) ou la cible (x=D) du 'blitter'. Un modulo est un chiffre qu'on ajoute automatiquement à l'adresse, à chaque fin de ligne, afin que l'adresse pointe au début de la ligne suivante. Chaque source et chaque cible ont leur propre modulo, qui permet à chacune d'avoir une taille différente, bien que chacune utilise une surface identique lors de son traitement par le 'blitter'.

TRACE LIGNE	BLTAMOD et BLTBMOD sont utilisés comme registres de stockage pour la pente de la ligne et doivent être pré-chargés, respectivement, avec les valeurs suivantes : (4Y-4X) et (4Y). Y/X = pente de la ligne. BLTCMOD et BLTDMOD doivent être, tous les deux, pré-chargés avec la largeur (en octets) de l'image à l'intérieur de laquelle sera dessinée la ligne (normalement, deux fois la largeur de l'écran en mots).

BLTxPTH	050	W	A	Pointeur sur x du 'blitter' (les 3 bits de poids fort)
BLTxPTL	052	W	A	Pointeur sur x du 'blitter' (les 15 bits de poids faible).

Cette paire de registres contient une adresse 18 bits de la zone source (x=A,B,C) ou de la zone cible (x=D) des données DMA du 'blitter'. On doit pré-charger ce pointeur avec l'adresse initiale des données que le 'blitter' va traiter. Dès que le 'blitter' aura terminé, la paire de registres contiendra la dernière adresse des données (plus l'incrémentation et le modulo).

TRACE LIGNE	BLTAPTL est employé comme registre accumulateur et devra être pré-chargé avec la valeur initiale de (2Y-X), où Y/X est la pente de la ligne. BLTCPT et BLTDPT (pour H et pour L) devront être pré-chargés avec l'adresse où commence la ligne.

BPLlMOD	108	W	A	Modulo du plan de bits (plans impairs)
BPL2MOD	10A	W	A	Modulo du plan de bits (plans pairs)

Ces registres renferment les modulo des plans de bits pairs et impairs. Un modulo est un chiffre qu'on ajoute automatiquement à l'adresse à la fin de chaque ligne, pour que l'adresse puisse ensuite pointer sur le début de la ligne suivante.
Comme les plans de bits pairs et impairs possèdent des modulo séparés, leurs dimensions pourront être différentes et différentes des dimensions de la fenêtre d'affichage.

BPLCON0	100	W	A D	Registre de contrôle du plan de bits (bits de contrôle divers)
BPLCON1	102	W	D	Registre de contrôle du plan de bits (contrôle du déroulement horizontal)
BPLCON2	104	W	D	Registre de contrôle du plan de bits (contrôle de la priorité vidéo)

Ces registres contrôlent le traitement des plans de bits et différents aspects de l'affichage.

BIT#	BPLCON0	BPLCON1	BPLCON2
15	HIRES	X	X
14	BPU2	X	X
13	BPU1	X	X
12	BPU0	X	X
11	HOMOD	X	X
10	DBLPF	X	X
09	COLOR	X	X
08	GAUD	X	X
07	X	PF2H3	X
06	X	PF2H2	PF2PRI
05	X	PF2H1	PF2P2
04	X	PF2H0	PF2P1
03	LPEN	PF1H3	PF2P0
02	LACE	PF1H2	PF1P2
01	ERSY	PF1H1	PF1P1
00	X	PF1H0	PF1lP0

HIRES	= Mode haute résolution (640)
BPU	= Code d'utilisation des plans de bits 000-110 (d'AUCUN à 6 compris)
HOMOD	= Mode 'hold-and-modify'
DBLPF	= Double aire de jeu (PF1=plans de bits impairs PF2=pairs)
COLOR	= Validation COULEUR vidéo composite
GAUD	= Validation 'genlock' audio (multiplexée sur la broche BKGND pendant le vertical blank)
LPEN	= Validation du crayon optique (réinitialisé à la mise sous tension)
LACE	= Validation mode entrelacé (réinitialisé à la mise sous tension)
ERSY	= Re-synchronisation externe (les 'pads' HSYNC, VSYNC deviennent des entrées) (réinitialisé à la mise sous tension)
PF2PRI	= Aire de jeu 2 (plans de bits pairs) a la priorité sur l'aire de jeu 1 (plans de bits impairs)
PF2P	= Code de priorité de l'aire de jeu 2 (par rapport aux sprites)
PF1P	= Code de priorité de l'aire de jeu 1 (par rapport aux sprites)
PF2H	= Code de déroulement horizontal de l'aire de jeu 2
PF1H	= Code de déroulement horizontal de l'aire de jeu 1

BPLxDAT	110	W	D	Données du plan de bits x (conversion parallèle/série)

Ces registres reçoivent les données DMA prélevées en mémoire vive par les pointeurs de plans de bits décrits ci-dessus. D'autres micro-processeurs peuvent aussi y écrire. Ils agissent comme un tampon parallèle/série de six mots pour un maximum de six plans de bits en mémoire. La conversion parallèle/série se déclenche quand on écrit le plan de bits #1, en indiquant l'achèvement de tous les plans de bits pour ce mot (16 pixels). Le MSB (bit le plus significatif) sort en premier et, par conséquent, se trouve toujours à gauche.

BPLxPTH	0E0	W	A	Pointeur de plan de bits x (les 3 bits de poids fort)
BPLxPTL	0E2	W	A	Pointeur de plan de bits x (les 15 bits de poids faible)

Cette paire de registres contient un pointeur 18 bits sur l'adresse des données DMA du plan de bits x (x = 1, 2, 3, 4, 5, 6). Ce pointeur doit être réinitialisé par le processeur ou le Copper pour pointer au début des données d'un plan de bits à chaque période de suppression verticale.

CLXCON	098	W	D	Contrôle de collision

Ce registre contrôle les plans de bits compris (validés) dans la détection de collision et l'état exigé, si celui-ci est inclus. Il contrôle aussi l'inclusion individuelle des sprites de numéro impair dans la détection de collision en faisant une opération logique 'OR' entre ces derniers et les sprites correspondants de numéro pair.

BIT#	FONCTION	DESCRIPTION
15	ENSP7	Validation sprite 7 (opération logique OR avec sprite 6)
14	ENSP5	Validation sprite 5 (opération logique OR avec sprite 4)
13	ENSP3	Validation sprite 3 (opération logique OR avec sprite 2)
12	ENSP1	Validation sprite 1 (opération logique OR avec sprite 0)
11	ENBP6	Validation plan de bits 6 (concordance requise pour collision)
10	ENBP5	Validation plan de bits 5 (concordance requise pour collision)
09	ENBP4	Validation plan de bits 4 (concordance requise pour collision)
08	ENBP3	Validation plan de bits 3 (concordance requise pour collision)
07	ENBP2	Validation plan de bits 2 (concordance requise pour collision)
06	ENBP1	Validation plan de bits 1 (concordance requise pour collision)
05	NVBP6	Valeur de concordance pour collision avec plan de bits 6
04	MVBP5	Valeur de concordance pour collision avec plan de bits 5
03	MVBP4	Valeur de concordance pour collision avec plan de bits 4
02	MVBP3	Valeur de concordance pour collision avec plan de bits 3
01	MVBP2	Valeur de concordance pour collision avec plan de bits 2
00	MVBP1	Valeur de concordance pour collision avec plan de bits 1

NOTE : Les plans de bits invalidés n'ont pas la possibilité d'éviter des collisions. Par conséquent, si tous les plans de bits sont invalidés, il y aura des collisions perpétuelles, sans tenir compte des valeurs de concordance.

CLXDAT	00E	R	D	Registre de données pour collision (lecture et mise à zéro)

Cette adresse lit et (met à zéro) le registre de détection de collision. Ci-dessous les bits d'assignation.

NOTE : Tous les plans de bits de numéro impair de l'aire de jeu 1 sont validés. Tous les plans de bits de numéro pair de l'aire de jeu 2 sont validés.

BIT#	COLLISIONS REGISTERED
15	Inutilisé
14	Sprite 4 (ou 5) avec sprite 6 (ou 7)
13	Sprite 2 (ou 3) avec sprite 6 (ou 7)
12	Sprite 2 (ou 3) avec sprite 4 (ou 5)
11	Sprite 0 (ou 1) avec sprite 6 (ou 7)
10	Sprite 0 (ou 1) avec sprite 4 (ou 5)
09	Sprite 0 (ou 1) avec sprite 2 (ou 3)
08	Aire de jeu 2 avec sprite 6 (ou 7)
07	Aire de jeu 2 avec sprite 4 (ou 5)
06	Aire de jeu 2 avec sprite 2 (ou 3)
05	Aire de jeu 2 avec sprite 0 (ou 1)
04	Aire de jeu 1 avec sprite 6 (ou 7)
03	Aire de jeu 1 avec sprite 4 (ou 5)
02	Aire de jeu 1 avec sprite 2 (ou 3)
01	Aire de jeu 1 avec sprite 0 (ou 1)
00	Aire de jeu 1 avec aire de jeu 2

COLORxx	180	W	D	Table de couleur xx

Il y a 32 registres de couleur (xx=00-31) qu'on appelle parfois collectivement "palette de couleurs".
Ils renferment des codes 12 bits qui représentent les couleurs rouge, verte et bleu des systèmes RVB. On sélectionne un de ces registres à la fois (par le code vidéo série de BPLxDAT) pour l'envoyer sur les broches de sortie vidéo RVB.
Le tableau ci-dessous indique l'emploi des bits du registre de couleurs.

```
BIT#  15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00
-----------------------------------------------------------------------------------------------
RVB   X   X   X   X  R3  R2 B1 R0 V3 V2  V1 V0 B3 B2  B1 B0
```

B=bleu, V=vert, R=rouge

COP1LCH	080	W	A	1er registre d'adresse du Copper (les 3 bits de poids fort)
COP1LCL	082	W	A	1er registre d'adresse du Copper (les 15 bits de poids faible)
COP2LCH	084	W	A	2ème registre d'adresse du Copper (les 3 bits de poids fort)
COP2LCL	086	W	A	2ème registre d'adresse du Copper (les 15 bits de poids faible)

Ces registres contiennent les adresses de saut décrites ci-dessus.

COPCON 02E W A Registre de contrôle du Copper

Il s'agit d'un registre 1 bit, qui, lorsqu'il se trouve à vrai, permet au Copper d'avoir accès au 'hardware du 'blitter'. Ce bit est mis à zéro à la réinitialisation lors de la mise sous tension, pour que le Copper n'ait pas la possibilité d'avoir accès au 'hardware' du 'blitter'.

BIT	NOM	FONCTION
01	CDANG	Mode danger du Copper. Permet au Copper d'avoir accès au 'blitter', si vrai.

COPINS 08C W A Identification de prélèvement d'instruction Copper

C'est une adresse bidon générée par le Copper chaque fois qu'il est en train de charger des instructions dans son propre registre d'instruction. Ce qui se produit concrètement à chaque cycle du Copper, si l'on excepte le deuxième (IR2) cycle de l'instruction MOVE. Nous donnons ci-dessous les trois types d'instructions.

MOVE - Transférer immédiatement vers la cible.
WAIT - Attendre jusqu'à ce que le compteur de faisceau soit égal ou supérieur à (interdit au Copper le bus tant qu'on n'aura pas atteint la position du faisceau).
SKIP - Saute si le compteur de faisceau est égal ou supérieur à (sautera l'instruction MOVE suivante tant qu'on n'aura pas atteint la position du faisceau).

COPINS (suite)

BIT#	MOVE IR1	MOVE IR2	WAIT UNTIL IR1	WAIT UNTIL IR2	SKIP IF IR1	SKIP IF IR2
15	X	RD15	VP7	BFD *	VP7	BFD *
14	X	RD14	VP6	VE6	VP6	VE6
13	X	RD13	VP5	VE5	VP5	VE5
12	X	RD12	VP4	VE4	VP4	VE4
11	X	RD11	VP3	VE3	VP3	VE3
10	X	RD10	VP2	VE2	VP2	VE2
09	X	RD09	VP1	VE1	VP1	VE1
08	DA8	RD08	VP0	VE0	VP0	VE0
07	DA7	RD07	HP8	HE8	HP8	HE8
06	DA6	RD06	HP7	HE7	HP7	HE7
05	DA5	RD05	HP6	HE6	HP6	HE6
04	DA4	RD04	HP5	HE5	HP5	HE5
03	DA3	RD03	HP4	HE4	HP4	HE4
02	DA2	RD02	HP3	HE3	HP3	HE3
01	DA1	RD01	HP2	HE2	HP2	HE2
00	0	RD00	1	0	1	1

IR1 =Premier registre d'instruction
IR2 =Deuxième registre d'instruction
DA =Adresse cible pour l'instruction MOVE. Prélevée pendant l'intervalle IR1, employée pendant l'intervalle IR2 sur le bus RGA.
RD =Données en mémoire vive prélevées par l'instruction MOVE, pendant l'intervalle IR2, directement de la mémoire vive à l'adresse qu'on indique au champ DA.
VP =Bit de comparaison pour la position verticale du faisceau.
HP =Bit de comparaison pour la position horizontale du faisceau.
VE =Valide la comparaison (bit de masque).
HE =Valide la comparaison (bit de masque).

* NOTE BFD = Invalide "Blitter Terminé". Lorsque ce bit est vrai, le drapeau "Blitter Terminé" n'a aucun effet sur le Copper. Quand ce bit est égal à zéro, le drapeau "Blitter Terminé" doit être vrai (en plus du reste de la comparaison de bits) avant que le Copper ne puisse sortir de son état d'attente ou sauter une instruction. Remarquez qu'il n'est pas possible de masquer la comparaison V7.

Fondamentalement, le Copper est un circuit deux cycles qui demande l'accès au bus seulement pendant les cycles de mémoire impairs (quatre cycles mémoire par instruction). Ceci évite des collisions avec l'affichage, l'audio, l'unité de disquettes, le rafraîchissement et les sprites, qui, tous, utilisent les cycles pairs. Le Copper nécessite (et a) la priorité seulement sur le 'blitter' et le micro-processeur.

Le Copper ne possède que trois types d'instructions : MOVE [TRANSFERE] immédiatement, WAIT [ATTENDRE] jusqu'à et SKIP [SAUTER] si. Toutes ces instructions (à l'exception de WAIT) ont besoin de deux cycles de bus (et de deux mots pour l'instruction). Etant donné qu'il ne demande l'accès que pendant les cycles impairs, il lui faudra quatre fois quatre cycles mémoire par instruction (les cycles mémoire ont une durée de 200 nano-secondes).

Il existe deux registres de saut indirect, COP1LC et COP2LC.
Il s'agit de registres pointeurs 18 bits dont on utilise le contenu
pour modifier le compteur du programme pour l'initialisation ou des sauts.
Ces registres sont transférés au programme, chaque fois qu'on écrit dans
les adresses 'strobe' COPJMP1 ou COPJMP2.
En outre, COP1LC est utilisé automatiquement au début de chaque
période de suppression verticale.

Il est important que l'un des deux registres de saut soit
initialisé et que ses adresses 'strobe' de saut soient
actives après la mise sous tension mais avant que le DMA
du Copper soit à son tour initialisé. Ceci garantit une
adresse de démarrage et un état déterminés.

COPJMP1	088	S	A	Re-démarrage du Copper à partir de la première adresse
COPJMP2	08A	S	A	Re-démarrage du Copper à partir de la deuxième adresse

Il s'agit des adresses 'strobe'. Quand on y écrit, elles
provoquent un saut indirect du Copper, en utilisant
l'adresse qui se trouve dans le premier ou le deuxième
registre d'adresse, registres décrits ci-dessous.
Le Copper lui-même a la possibilité d'écrire dans ces
adresses et provoque ainsi son propre saut indirect.

DDFSTOP	094	W	A	Arrêt prélèvement données d'affichage (position horiz.)
DDFSTRT	092	W	A	Début prélèvement données d'affichage (position horiz.)

Ces registres contrôlent la synchronisation horizontale du début et de l'arrêt du prélèvement des données d'affichage des DMA des plans de bits. La synchronisation verticale des DMA des plans de bits est identique à celle des fenêtres d'affichage décrite ci-dessus. Les modulo des plans de bits dépendent de la taille horizontale du plan de bits et des dimensions de la fenêtre de prélèvement des données.

Assignations des bits du registre

BIT#	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USAGE	X	X	X	X	X	X	X	X	H8	H7	H6	H5	H4	H3	X	X

(les bits marqués d'un X devraient toujours se trouver à zéro pour garder une compatibilité ascendante)

Les tableaux ci-dessous indiquent la synchro pour le début et l'arrêt des différents contenus des registres.

DDFSTRT (bord gauche prélèvement données affichage)

USAGE	H8	H7	H6	H5	H4
Extra large (maxi) *	0	0	1	0	1
Large	0	0	1	1	0
Normale	0	0	1	1	1
Etroite	0	1	0	0	0

DDFSTOP (bord droit prélèvement données affichage)

USAGE	H8	H7	H6	H5	H4
Etroite	1	1	0	0	1
Normale	1	1	0	1	0
Large (maxi)	1	1	0	1	1

DIWSTOP	090	W	A	Point d'arrivée fenêtre d'affichage (position horizontale-verticale en bas à droite)
DIWSTRT	08E	W	A	Début fenêtre d'affichage (position horizontale-verticale en haut à gauche)

Ces registres contrôlent les dimensions de la fenêtre d'affichage, ainsi que sa position en fixant le coin supérieur gauche et le coin inférieur droit.

BIT#	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USAGE	V7	V6	V5	V4	V3	V2	V1	V0	H7	H6	H5	H4	H3	H2	H1	H0

DIWSTRT est réservé verticalement aux 2/3 de la partie supérieure de l'affichage (V8=0) et, horizontalement, aux 3/4 de la partie gauche de l'affichage (H8=0).

DIWSTOP est réservé verticalement à 1/2 de la partie supérieure de l'affichage (V8=/=V7) et, horizontalement, à 1/4 de la partie droite de l'affichage (H8=1).

DMACON	096	W	A D P	Ecriture de contrôle DMA (mise à 0 ou à 1)
DMACONR	002	R	A P	Lecture de contrôle DMA (et de l'état du 'blitter')

Ce registre contrôle tous les canaux DMA et renferme les bits d'état des DMA du 'blitter'.

BIT#	FONCTION	DESCRIPTION
15	SET/CLR	Bit de contrôle de mise à 0 et à 1. Détermine si les bits où on a écrit un 1 devront être mis à 1 ou à 0. Les bits où on a écrit un 0, restent inchangés.
14	BBUSY	Bit d'état de disponibilité du 'blitter' (destiné seulement à la lecture).
13	BZERO	Bit d'état "zéro logique" du 'blitter' (destiné seulement à la lecture).
12	X	
11	X	
10	BLTPRI	Priorité des DMA du 'blitter' (sur le micro-processeur) (appelé aussi "blitter nasty" ["bit de danger"]) (invalide la broche /BLS, afin d'éviter que le processeur vole des cycles de bus pendant le fonctionnement des DMA du 'blitter').
09	DMAEN	Validation de tous les DMA ci-dessous
08	BPLEN	Validation DMA plans de bits
07	COPEN	Validation DMA Copper
06	BLTEN	Validation DMA Blitter
05	SPREN	Validation DMA sprites
04	DSKEN	Validation DMA unité de disquette
03	AUD3EN	Validation DMA canal audio 3
02	AUD2EN	Validation DMA canal audio 2
01	AUD1EN	Validation DMA canal audio 1
00	AUD0EN	Validation DMA canal audio 0

DSKBYTR	01A	R	P	Lecture d'état et des octets de données de la disquette

Ce registre est le tampon de données du processeur de l'unité de disquettes. Les données en provenance de la disquette (en mode lecture) sont chargées dans ce registre un bit à la fois et le bit 15 (DSKBYT) est mis à vrai.

BIT#	FONCTION	DESCRIPTION
15	DSKBYT	Octet qui indique que la disquette est prête (mis à 0 en lecture)
14	DMAON	Image du bit 15 (DMAEN) dans DSKLEN, sur qui on fait une opération logique AND avec le bit 09 (DMAEN) de DMACON
13	DISKWRITE	Image du bit 14 (WRITE) de DSKLEN
12	WORDEQUAL	Ce bit est vrai à condition que le registre DSKSYNC soit égal aux données en provenance de la disquette.
11-08	X	Inutilisés
07-00	DATA	Données en octets de la disquette

DSKDAT	026	W	P	Ecriture des données DMA sur la disquette
DSKDATR	008	ER	P	Lecture des données DMA à partir de la disquette (adresse bidon pour la lecture précédente)

Ce registre est le tampon des données DMA de l'unité de disquettes. Il renferme deux octets de données qui sont soit envoyées (écrites) sur la disquette, soit reçues (lues) à partir de la disquette. Le mode écriture est validé par le bit 14 du registre LENGTH. Le contrôleur des DMA transfère des données à ou de ce registre et de la mémoire vive ou en mémoire vive automatiquement et, quand il n'y a plus de données DMA (length=0), il envoie une interruption de type bloc de disquette.
Pour les interruptions, voir ci-dessous.

DSKLEN	024	W	P	Taille des données de la disquette

Ce registre contient la taille (nombre de mots) des données DMA de la disquette. Il renferme aussi deux bits de contrôle, un bit de validation DMA et un bit de direction (lecture/écriture) des DMA.

BIT#	FONCTION	DESCRIPTION
15	DMAEN	Validation des DMA disquette
14	WRITE	Ecriture sur la disquette, si égal à 1 (de la mémoire vive à la disquette)
13-0	LENGTH	Taille (# de mots) des données DMA.

DSKPTH	020	W	A	Pointeur disquette (3 bits de poids fort)
DSKPTL	022	W	A	Pointeur disquette (15 bits de poids faible)

Cette paire de registres contient l'adresse 18 bits des données DMA de la disquette. Ces registres d'adresse doivent être validés par le processeur ou par le Copper avant que les DMA disquette ne soient validés.

DSKSYNC	07E	W	P	Registre de synchro de la disquette

Contient le code de concordance pour la synchronisation de la disquette en lecture. Cf. le bit 10 d'ADKCON.

INTENA	09A	W	P	Bits de validation d'interruption (met les bits à 1 ou à 0)
INTENAR	01C	R	P	Bits de validation d'interruption (en lecture)

Ce registre contient les bits de validation d'interruption. Nous donnons ci-dessous l'assignation des bits pour demander et pour valider les interruptions.

BIT#	FONCTION	NIVEAU	DESCRIPTION
15	SET/CLR		Bit de contrôle de mise à 0 et à 1. Détermine si les bits où on a écrit un 1 devront être mis à 1 ou à 0. Les bits où on a écrit un 0, restent inchangés.
14	INTEN		Interrupteur général d'interruption (seulement pour une validation et non pour une demande)
13	EXTER	6	Interruption externe
12	DSKSYN	5	Le registre de synchro de la disquette (DSKSYNC) correspond aux données de la disquette
11	RBF	5	Tampon de réception du port série plein
10	AUD3	4	Bloc du canal audio 3 terminé
09	AUD2	4	Bloc du canal audio 2 terminé
08	AUDl	4	Bloc du canal audio 1 terminé
07	AUD0	4	Bloc du canal audio 0 terminé
06	BLIT	3	'Blitter' terminé
05	VERTB	3	Début de la suppression verticale
04	COPER	3	Copper
03	PORTS	2	Ports d'E/S et minuteries
02	SOFT	1	Réservé aux interruptions faites par le 'software'
01	DSKBLK	1	Bloc disquette terminé
00	TBE	1	Tampon de transmission du port série vide

INTREQ	09C	W	P	Bits de demande d'interruption (mis à 0 ou à 1)
INTREQR	01E	R	P	Bits de demande d'interruption (en lecture)

Ce registre contient les bits de demande d'interruption (ou drapeaux). Ces bits peuvent être interrogés par le processeur; s'ils sont validés par les bits listés dans le registre suivant, ils pourront provoquer des interruptions du processeur. Une mise à 0 et une mise à 1 sont nécessaires pour charger des données arbitraires dans ce registre. Ces bits d'état ne sont pas réinitialisés automatiquement, après une interruption, et il faudra les réinitialiser, si nécessaire, en écrivant dans cette adresse. L'assignation des bits est identique à celle du registre de validation ci-dessous.

JOY0DAT	00A	R	D	Données joystick/souris 0(verticale à gauche, horizontale)
JOY1DAT	00C	R	D	Données joystick/souris 1(verticale à droite, horizontale)

Ces adresses lisent chacune une paire de compteurs 8 bits pour la souris. 0=paire gauche du contrôleur, 1=paire droite du contrôleur (en tout, quatre compteurs). Nous indiquons ci-dessous l'emploi des bits pour les adresses gauche et droite. Chaque compteur est rythmé par des signaux provenant de deux broches du contrôleur. On peut lire les bits 1 et 0 de chaque compteur pour déterminer l'état de ces deux broches. Ceci permet aux broches de se dédoubler en tant qu'entrées de commutation pour les manettes de commande.

Utilisation du compteur pou la souris :
(broches 1,3=signal d'horloge Y, broches 2,4=signal d'horloge X)

BIT#	15,	14,	13,	12,	11,	10,	09,	08,	07,	06,	05,	04,	03,	02,	01,	00
0DAT	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0
1DAT	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0

Le tableau suivant montre l'utilisation des broches du connecteur souris/manette de commande. Les broches (et leurs fonctions) sont échantillonnées (et multiplexées) dans la puce Denise dans les intervalles des signaux d'horloge indiqués dans le tableau. Celui-ci n'est qu'un tableau de consultation et le programmeur ne devrait pas en avoir besoin. (Remarquez que les fonctions de la manette de commande sont toutes "zéro actives" sur les broches du connecteur).

			Échantillonnées par DENISE		
Connex.	Fonction	Fonction			
Broches	Joystick	Souris	Broche	Nom	Horloge
L1	FORW*	Y	38	M0V at	CCK
L3	LEFT*	YQ	38	M0V at	CCK*
L2	BACK*	X	9	M0H at	CCK
L4	RIGH*	XQ	9	M0H at	CCK*
R1	FORW*	Y	39	M1V at	CCK
R3	LEFT*	YQ	39	M1V at	CCK*
R2	BACK*	X	8	M1H at	CCK
R4	RIGH*	XQ	8	M1H at	CCK*

Après avoir été échantillonnés, les signaux des broches du connecteur sont employés pour la quadrature pour rythmer les compteurs de la souris. Les fonctions de la manette de commande LEFT et RIGHT (actives à 1) sont disponibles directement sur les bits Y1 et X1 de chaque compteur. Par conséquent, pour recréer les fonctions FORWARD et BACK de la manette de commande, il sera nécessaire de les combiner logiquement (OR exclusif) avec les deux bits de poids faible de chaque compteur. C'est ce qui est illustré dans le tableau suivant :

Pour détecter	lire ces bits de compteur
Forward (avant)	Y1 xor Y0 (BIT#09 xor BIT#08)
Left (gauche)	Y1
Back (arrière)	X1 xor X0 (BIT#01 xor BIT#00)
Right (droite)	X1

JOYTEST	036	W	D	Ecriture dans les quatre compteurs manette de jeu/souris en même temps.

Données de test à écrire dans le compteur de la souris :

BIT#	15,	14,	13,	12,	11,	10,	09,	08,	07,	06,	05,	04,	03,	02,	01,	00
0DAT	Y7	Y6	Y5	Y4	Y3	Y2	xx	xx	X7	X6	X5	X4	X3	X2	xx	xx
1DAT	Y7	Y6	Y5	Y4	Y3	Y2	xx	xx	X7	X6	X5	X4	X3	X2	xx	xx

POT0DAT	012	R	P	Paire gauche de données des compteurs du potentiomètre (vert.,horiz.)
POT1DAT	014	R	P	Paire droite de données des compteurs du potentiomètre (vert.,horiz.)

Chacune de ces deux adresses lit une paire de compteurs de potentiomètre 8 bits (quatre compteurs en tout).
Nous indiquons ci-dessous l'assignation des bits pour ces deux adresses.
Les compteurs sont arrêtés par les signaux en provenance de deux connecteurs du contrôleur (gauche-droite), avec deux broches chacun.

BIT#	15,	14,	13,	12,	11,	10,	09,	08,	07,	06,	05,	04,	03,	02,	01,	00
RIGHT	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0
LEFT	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	X7	X6	X5	X4	X3	X2	X1	X0

CONNECTEURS				PAULA	
Loc.	Dir.	Sym	Broche	Broche#	Nom Broche
RIGHT	Y	RY	9	36	(POT1Y)
RIGHT	X	RX	5	35	(POT1X)
LEFT	Y	LY	9	33	(POT0Y)
LEFT	X	LX	5	32	(POT0X)

POTGO	034	W	P	Ecriture de données et démarrage du port de potentiomètre
POTGOR	016	R	P	Lecture de données du port de potentiomètre (autrefois appelé POTINP)

Ce registre contrôle un port d'E/S bi-directionnel 4 bits qui partage les mêmes quatre broches, comme les quatre compteurs de potentiomètre ci-dessus.

BIT#	FONCTION	DESCRIPTION
15	OUTRY	Validation de sortie pour la broche 36 de Paula
14	DATRY	E/S données sur broche 36 de Paula
13	OUTRX	Validation de sortie pour la broche 35 de Paula
12	DATRX	E/S données sur broche 35 de Paula
11	OUTLY	Validation de sortie pour la broche 33 de Paula
10	DATLY	E/S données sur broche 33 de Paula
09	OUTLX	Validation de sortie pour la broche 32 de Paula
08	DATLX	E/S données sur broche 32 de Paula
07-01	0	Réservé pour code d'ID de la puce (actuellement 0)
00	START	Démarrage potentiomètres (condensateurs de décharge, démarrage des compteurs)

REFPTR	028	W	A	Pointeur de rafraîchissement

Ce registre est employé comme générateur d'adresse pour un rafraîchissement dynamique de la mémoire vive. On peut y écrire seulement pour faire des essais; le processeur ne devrait jamais y écrire.

SERDAT 030 W P Ecriture des données et des bits d'arrêt du port série (tampon de transmission de données)

Cette adresse écrit des données dans un tampon de transmission de données. A partir de ce tampon, les données sont transférées dans un registre à décalage pour une transmission, si le registre est vide. Ceci met en place la demande d'interruption TBE (tampon de transmission vide). On doit fournir un bit d'arrêt en tant que partie du mot de données. La taille du mot de données est mise en place selon la position du bit d'arrêt.

BIT#	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
USE	0	0	0	0	0	0	S	D8	D7	D6	D5	D4	D3	D2	D1	D0

Note: S = bit d'arrêt = 1, D = bits de données.

SERDATR 018 R P Lecture de données et d'état du port série (tampon de réception de données)

Cette adresse lit les données en provenance d'un tampon de réception de données. Dans ce tampon, les données sont chargées à partir d'un registre à décalage. Plusieurs demandes d'interruptions sont aussi lues à cette adresse, en même temps que des données, comme nous le montrons ci-dessous :

BIT#	SYM	FONCTION
15	OVRUN	Surcharge du port série récepteur. Mis à 0 par le bit de réinitialisation 11 de INTREQ.
14	RBF	Tampon de réception du port série plein (image).
13	TBE	Tampon de transmission du port série vide (image).
12	TSRE	Registre à décalage de transmission du port série vide. Réinitialisé, quand on charge le tampon.
11	RXD	La broche RXD réceptionne les données série de l'UART pour un test direct du bit par le micro-processeur.
10	0	Inutilisé
09	STP	Bit d'arrêt
08	STP-DB8	Bit d'arrêt si LONG, autrement bit de données.
07	DB7	Bit de données
06	DB6	Bit de données
05	DB5	Bit de données
04	DB4	Bit de données
03	DB3	Bit de données
02	DB2	Bit de données
01	DB1	Bit de données
00	DB0	Bit de données

SERPER	032	W	P	Contrôle et période du port série

Ce registre contient le bit de contrôle LONG auquel nous faisions référence ci-dessus et un chiffre 15 bits qui définit le taux d'émission/réception du port série. Si ce chiffre est N alors le taux sera égal à 1 bit toutes les (N+1) * 0.2794 micro-secondes.

BIT#	SYM	FONCTION
15	LONG	Définit la réception série en tant que mot de 9 bits.
14-00	RATE	Définit le taux d'émission/réception taux=1 / ((N+1) * 0.2794 micro-sec.)

SPRxCTL	142	W	A D	Contrôle et position d'arrêt verticale du sprite x
SPRxPOS	140	W	A D	Données de la position initiale vert./horiz. du sprite x

Ces deux registres travaillent ensemble, comme registres de contrôle des dimensions et des caractéristiques des sprites. D'ordinaire, ils sont chargés par le canal DMA de sprite lors de la suppression verticale; cependant, on peut les charger à tout moment par l'intermédiaire d'un autre processeur.

Registre SPRxPOS :

BIT#	SYM	FONCTION
15-08	SV7-SV0	Valeur verticale initiale. Le bit de poids fort (SV8) se trouve dans le registre SPRxCTL ci-dessous.
07-00	SH8-SH1	Valeur horizontale initiale. Le bit de poids faible (SH0) se trouve dans le registre SPRxCTL ci-dessous.

Registre SPRxCTL (le fait d'écrire à cette adresse invalide le circuit comparateur de sprites vertical) :

BIT#	SYM	FONCTION
15-08	EV7-EV0	Les 8 bits de poids faible de la valeur verticale finale (d'arrêt)
07	ATT	Bit de contrôle 'attach' du sprite (pour les sprites impairs)
06-04	X	Inutilisés
02	SV8	Bit de poids fort de la valeur verticale initiale
01	EV8	Bit de poids fort de la valeur verticale finale (d'arrêt)
00	SH0	Bit de poids faible de la valeur horizontale initiale

SPRxDATA	144	W	D	Registre A de données de l'image du sprite x
SPRxDATB	146	W	D	Registre B de données de l'image du sprite x

Ces registres mettent dans un tampon les données d'image du sprite. Habituellement, ils sont chargés par le canal DMA de sprite, mais peuvent être chargés à n'importe quel moment par un autre processeur. Quand intervient une comparaison horizontale, les tampons sont vidés dans des registres à décalage et envoyés à l'affichage de manière séquentielle, le MSB (bit plus significatif), à gauche, en premier.

NOTE : Le fait d'écrire dans un tampon valide (arme) le sprite. Le fait d'écrire dans le registre SPRxCTL invalide le sprite. Si les sprites sont validés, les données des tampons A et B seront envoyées en sortie lorsque le compteur de faisceau sera égal à la valeur de position horizontale du sprite dans le registre SPRxPOS.

SPRxPOS cf. SPRxCTL

SPRxPTH	120	W	A	Pointeur du sprite x (3 bits de poids fort)
SPRxPTL	122	W	A	Pointeur de sprite x (15 bits de poids faible)

Cette paire de registres contient les adresses 18 bits des données DMA du sprite x (x=0,1,2,3,4,5,6,7). Ces adresses devront être initialisées par le processeur ou le Copper à chaque intervalle de suppression verticale.

STREQU	038	S	D	'Strobe' pour la synchro horizontale avec VB et EQU
STRHOR	03C	S	D P	'Strobe' pour la synchro horizontale
STRLONG	03E	S	D	'Strobe' pour identification d'une ligne paire horizontale.

La première des trois adresses 'strobe' ci-dessus est placée dans le bus d'adresse de destination pendant la première plage de temps consacrée au rafraîchissement. Le quatrième 'strobe', indiqué ci-dessus, est utilisé pendant la deuxième plage de temps consacrée au rafraîchissement de chacune des autres lignes impaires à identifier (228). Il y a quatre plages de temps pour le rafraîchissement et toute plage non utilisée par les 'strobes' enverra une adresse nulle (FF) sur le bus d'adresse de destination.

STRVBL	03A	S	D	'Strobe' pour la synchro horizontale avec VB (suppression verticale)
VHPOSR	006	R	A	Lecture de la position verticale et horizontale du faisceau ou du crayon optique
VHPOSW	02C	W	A	Ecriture de la position verticale et horizontale du faisceau ou du crayon optique

BIT#	15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00
USE	V7 V6 V5 V4 V3 V2 V1 V0, H8 H7 H6 H5 H4 H3 H2 H1

RESOLUTION = 1/160 de la largeur de l'écran (280 nano-sec.)

VPOSR	004	R	A	Lecture du bit vertical le plus significatif (et chute de trame)
VPOSW	02A	W	A	Ecriture du bit vertical le plus significatif (et chute de trame)

BIT#	15, 14, 13, 12, 11, 10, 09, 08, 07, 06, 05, 04, 03, 02, 01, 00
USE	LOF -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- V8

LOF=Trame paire (bit d'auto-contrôle du 'toggle' [interrupteur à bascule] dans BPLCON0).

APPENDICE B

RECAPITULATION DES REGISTRES PAR ORDRE D'ADRESSE

Cette appendice contient des renseignements sur l'ensemble des registres par ordre d'adresse.

Voici les codes et les abréviations employés dans cette appendice :

& Registre employé seulement par un canal DMA.

% Registre employé d'ordinaire par un canal DMA, parfois par les processeurs.

+ Paire de registres d'adresse. Il devra s'agir d'une adresse paire qui pointe en mémoire 'chip'.

* Le Copper ne peut écrire à cette adresse.

- Le Copper ne peut écrire à cette adresse tant que COPCON, le "bit de danger du Copper" est vrai.

A,D,P
A=puce Agnus, D=puce Denise, P=puce Paula.

W,R
W=destiné exclusivement à l'écriture; R=destiné exclusivement à la lecture.

ER Lecture anticipée ['Early read']. Il s'agit d'un transfert de données DMA en mémoire vive, en provenance de la disquette ou du 'blitter'. La synchronisation pour la mémoire vive demande que les données se trouvent sur le bus avant les cycles de lecture du micro-processeur.
Par conséquent, c'est la synchronisation d'Agnus qui déclenche ces transferts et non une adresse de lecture sur le bus d'adresse de destination.

S 'Strobe' (adresse d'écriture sans bits de registre). Le fait d'écrire dans le registre produit un résultat.

PTL,PTH
Pointeur en mémoire 'chip' qui adresse les données DMA. Il devra être rechargé par un processeur avant d'être utilisé (suppression verticale pour les pointeurs de sprites et de plans de bits et, pour les pointeurs du 'blitter', avant de démarrer ce dernier).

LCLLCH
Emplacement en mémoire 'chip' (adresse initiale) des données DMA. Employé pour remettre en marche les pointeurs automatiquement, des pointeurs comme le compteur de programme du Copper (pendant la suppression verticale) et le compteur d'échantillons audio (lorsque prend fin la totalité de la longueur audio).

MOD
Modulo 15 bits. Un chiffre qu'on ajoute automatiquement à l'adresse en mémoire, à la fin de chaque ligne, pour générer l'adresse de départ de la ligne suivante. Ce qui permet au 'blitter' (ou à la fenêtre d'affichage) de travailler sur (ou d'afficher) une fenêtre de données plus petite que l'image qui se trouve réellement en mémoire (mappe de mémoire). Utilise 15 bits, plus l'extension de signe.

NOM	ADR	R/W	PUCE	FONCTION
BLTDDAT &	*000	ER	A	Lecture "anticipée" de la cible du 'blitter' (adresse bidon)
DMACONR	*002	R	A P	Lecture de contrôle (et d'état du 'blitter') DMA
VPOSR	*004	R	A	Lecture du bit le plus sign. (et chute de trame)
VHPOSR	*006	R	A	Lecture de position vert. et horiz. du faisceau
DSKDATR &	*008	ER	P	Lecture "anticipée" des données disquette (adresse bidon)
JOY0DAT	*00A	R	D	Données joystick-souris 0 (vert.,horiz.)
JOY1DAT	*00C	R	D	Données joystick-souris 1 (vert.,horiz.)
CLXDAT	*00E	R	D	Registre de données pour collision (lecture et initialisation)
ADKCONR	*010	R	P	Lecture du registre de contrôle disquette, audio
POT0DAT	*012	R	P	Données paire de compteurs de potentiomètre 0 (vert.,horiz.)
POT1DAT	*014	R	P	Données paire de compteurs de potentiomètre 1 (vert.,horiz.)
POTGOR	*016	R	P	Lecture de données du port de potentiomètre (anciennement appelé POTINP)
SERDATR	*018	R	P	Lecture d'état et de données du port série
DSKBYTR	*01A	R	P	Lecture d'état et des octets de données de la disquette
INTENAR	*01C	R	P	Lecture des bits de validation d'interruption
INTREQR	*01E	R	P	Lecture des bits de demande d'interruption
DSKPTH +	*020	W	A	Pointeur disquette (3 bits de poids fort)
DSKPTL +	*022	W	A	Pointeur disquette (15 bits de poids faible)
DSKLEN	*024	W	P	Taille des données de la disquette
DSKDAT &	*026	W	P	Ecriture de données DMA disquette
REFPTR &	*028	W	A	Pointeur pour le rafraîchissement
VPOSW	*02A	W	A	Ecriture du bit vertical le plus sign. (et chute de trame)
VHPOSW	*02C	W	A	Ecriture des positions vert. et horiz. du faisceau
COPCON	*02E	W	A	Registre de contrôle du co-processeur (CDANG)
SERDAT	*030	W	P	Ecriture des données et des bits d'arrêt du port série
SERPER	*032	W	P	Contrôle et période du port série
POTGO	*034	W	P	Ecriture de données et démarrage du port de potentiomètre
JOYTEST	*036	W	D	Ecriture dans les quatre compteurs joystick/souris en même temps
STREQU &	*038	S	D	'Strobe' pour synchro horiz. avec VB et EQU
STRVBL &	*03A	S	D	'Strobe' pour synchro horiz. avec VB (vertical blank)
STRHOR &	*03C	S	D P	'Strobe' pour synchro horiz.
STRLONG &	*03E	S	D	'Strobe' pour identification d'une ligne paire horizontale
BLTCON0	-040	W	A	Registre de contrôle du 'blitter' 0
BLTCON1	-042	W	A	Registre de contrôle du 'blitter' 1
BLTAFWM	-044	W	A	Masque pour le premier mot de la source A du 'blitter'
BLTALWM	-046	W	A	Masque pour le dernier mot de la source A du 'blitter'
BLTCPTH +	-048	W	A	Pointeur du 'blitter' sur la source C (les 3 bits de poids fort)
BLTCPTL +	-04A	W	A	Pointeur du 'blitter' sur la source C (les 15 bits de poids faible)
BLTBPTH +	-04C	W	A	Pointeur du 'blitter' sur la source B (les 3 bits de poids fort)
BLTBPTL +	-04E	W	A	Pointeur du 'blitter' sur la source B (les 15 bits de poids faible)
BLTAPTH +	-050	W	A	Pointeur du 'blitter' sur la source A (les 3 bits de poids fort)
BLTAPTL +	-052	W	A	Pointeur du 'blitter' sur la source A (les 15 bits de poids faible)
BLTDPTH +	-054	W	A	Pointeur du 'blitter' sur la cible D (les 3 bits de poids fort)
BLTDPTL +	-056	W	A	Pointeur du 'blitter' sur la cible D (les 15 bits de poids faible)
BLTSIZE	-058	W	A	Démarrage du 'blitter' et dimensions (hauteur, largeur de la fenêtre)
	-05A			
	-05C			
	-05E			
BLTCMOD	-060	W	A	Modulo du 'blitter' pour la source C
BLTBMOD	-062	W	A	Modulo du 'blitter' pour la source B
BLTAMOD	-064	W	A	Modulo du 'blitter' pour la source A
BLTDMOD	-066	W	A	Modulo du 'blitter' pour la cible D
	-068			
	-06A			
	-06C			
	-06E			
BLTCDAT %	-070	W	A	Registre de données de la source C du 'blitter'
BLTBDAT %	-072	W	A	Registre de données de la source B du 'blitter'
BLTADAT %	-074	W	A	Registre de données de la source A du 'blitter'

NOM	ADR	R/W	PUCE	FONCTION
	-076			
	-078			
	-07A			
	-07C			
DSKSYNC	-07E	W	P	Registre du modèle de synchro de la disquette pour lire la disquette
COP1LCH +	080	W	A	Premier registre d'adresse du co-processeur (les 3 bits de poids fort)
COP1LCL +	082	W	A	Premier registre d'adresse du co-processeur (les 15 bits de poids faible)
COP2LCH +	084	W	A	Deuxième registre d'adresse du co-processeur (les 3 bits de poids fort)
COP2LCL +	086	W	A	Deuxième registre d'adresse du co-processeur (les 15 bits de poids faible)
COPJMP1	088	S	A	Relance du co-processeur à la première adresse
COPJMP2	08A	S	A	Relance du co-processeur à la deuxième adresse
COPINS	08C	W	A	Identification de prélèvement d'instruction du co-processeur
DIWSTRT	08E	W	A	Début de la fenêtre d'affichage (positions vert./horiz. du coin supérieur gauche)
DIWSTOP	090	W	A	Fin de la fenêtre d'affichage (positions vert./horiz. du coin inférieur droit)
DDFSTRT	092	W	A	Début de prélèvement de données d'affichage pour un plan de bits (position horiz.)
DDFSTOP	094	W	A	Fin de prélèvement de données d'affichage pour un plan de bits (position horiz.)
DMACON	096	W	A D P	Ecriture de contrôle DMA (mettre à 0 ou à 1)
CLXCON	098	W	D	Contrôle de collision
INTENA	09A	W	P	Bits de validation d'interruption (mettre les bits à 0 ou à 1)
INTREQ	09C	W	P	Bits de demande d'interruption (mettre les bits à 0 ou à 1)
ADKCON	09E	W	P	Contrôle de l'UART disquette, audio
AUD0LCH +	0AO	W	A	Adresse du canal audio 0 (les 3 bits de poids fort)
AUD0LCL +	0A2	W	A	Adresse du canal audio 0 (les 15 bits de poids faible)
AUD0LEN	0A4	W	P	Longueur du canal audio 0
AUD0PER	0A6	W	P	Période du canal audio 0
AUD0VOL	0A8	W	P	Volume du canal audio 0
AUD0DAT &	0AA	W	P	Données du canal audio 0
	0AC			
	0AE			
AUD1LCH +	0B0	W	A	Adresse du canal audio 1 (les 3 bits de poids fort)
AUD1LCL +	0B2	W	A	Adresse du canal audio 1 (les 15 bits de poids faible)
AUD1LEN	0B4	W	P	Longueur du canal audio 1
AUD1PER	0B6	W	P	Période du canal audio 1
AUD1VOL	0B8	W	P	Volume du canal audio 1
AUD1DAT &	0BA	W	P	Données du canal audio 1
	0BC			
	0BE			
AUD2LCH +	0C0	W	A	Adresse du canal audio 2 (les 3 bits de poids fort)
AUD2LCL +	0C2	W	A	Adresse du canal audio 2 (les 15 bits de poids faible)
AUD2LEN	0C4	W	P	Longueur du canal audio 2
AUD2PER	0C6	W	P	Période du canal audio 2
AUD2VOL	0C8	W	P	Volume du canal audio 2
AUD2DAT &	0CA	W	P	Données du canal audio 2
	0CC			
	0CE			
AUD3LCH +	0D0	W	A	Adresse du canal audio 3 (les 3 bits de poids fort)
AUD3LCL +	0D2	W	A	Adresse du canal audio 3 (les 15 bits de poids faible)
AUD3LEN	0D4	W	P	Longueur du canal audio 3
AUD3PER	0D6	W	P	Période du canal audio 3
AUD3VOL	0D8	W	P	Volume du canal audio 3
AUD3DAT &	0DA	W	P	Données du canal audio 3

NOM	ADR	R/W	PUCE	FONCTION
	0DC			
	0DE			
BPL1PTH +	0E0	W	A	Pointeur du plan du bits 1 (les 3 bits de poids fort)
BPL1PTL +	0E2	W	A	Pointeur du plan de bits 1 (les 15 bits de poids faible)
BPL2PTH +	0E4	W	A	Pointeur du plan de bits 2 (les 3 bits de poids fort)
BPL2PTL +	0E6	W	A	Pointeur du plan de bits 2 (les 15 bits de poids faible)
BPL3PTH +	0E8	W	A	Pointeur du plan de bits 3 (les 3 bits de poids fort)
BPL3PTL +	0EA	W	A	Pointeur du plan de bits 3 (les 15 bits de poids faible)
BPL4PTH +	0EC	W	A	Pointeur du plan de bits 4 (les 3 bits de poids fort)
BPL4PTL +	0EE	W	A	Pointeur du plan de bits 4 (les 15 bits de poids faible)
BPL5PTH +	0F0	W	A	Pointeur du plan de bits 5 (les 3 bits de poids fort)
BPL5PTL +	0F2	W	A	Pointeur du plan de bits 5 (les 15 bits de poids faible)
BPL6PTH +	0F4	W	A	Pointeur du plan de bits 6 (les 3 bits de poids fort)
BPL6PTL +	0F6	W	A	Pointeur du plan de bits 6 (les 15 bits de poids faible)
	0F8			
	0FA			
	0FC			
	0FE			
BPLCON0	100	W	A D	Registre de contrôle des plans de bits (divers bits de contrôle)
BPLCON1	102	W	D	Registre de contrôle des plans de bits (valeur de déroulement PF1, PF2)
BPLCON2	104	W	D	Registre de contrôle des plans de bits (contrôle de priorité)
	106			
BPL1MOD	108	W	A	Modulo plans de bits (plans impairs)
BPL2MOD	10A	W	A	Modulo plans de bits (plans pairs)
	10C			
	10E			
BPL1DAT &	110	W	D	Données du plan de bits 1 (conversion parallèle-série)
BPL2DAT &	112	W	D	Données du plan de bits 2 (conversion parallèle-série)
BPL3DAT &	114	W	D	Données du plan de bits 3 (conversion parallèle-série)
BPL4DAT &	116	W	D	Données du plan de bits 4 (conversion parallèle-série)
BPL5DAT &	118	W	D	Données du plan de bits 5 (conversion parallèle-série)
BPL6DAT &	11A	W	D	Données du plan de bits 6 (conversion parallèle-série)
	11C			
	11E			

NOM	ADR	R/W	PUCE	FONCTION
SPR0PTH +	120	W	A	Pointeur du sprite 0 (les 3 bits de poids fort)
SPR0PTL +	122	W	A	Pointeur du sprite 0 (les 15 bits de poids faible)
SPR1PTH +	124	W	A	Pointeur du sprite 1 (les 3 bits de poids fort)
SPR1PTL +	126	W	A	Pointeur du sprite 1 (les 15 bits de poids faible)
SPR2PTH +	128	W	A	Pointeur du sprite 2 (les 3 bits de poids fort)
SPR2PTL +	12A	W	A	Pointeur du sprite 2 (les 15 bits de poids faible)
SPR3PTH +	12C	W	A	Pointeur du sprite 3 (les 3 bits de poids fort)
SPR3PTL +	12E	W	A	Pointeur du sprite 3 (les 15 bits de poids faible)
SPR4PTH +	130	W	A	Pointeur du sprite 4 (les 3 bits de poids fort)
SPR4PTL +	132	W	A	Pointeur du sprite 4 (les 15 bits de poids faible)
SPR5PTH +	134	W	A	Pointeur du sprite 5 (les 3 bits de poids fort)
SPR5PTL +	136	W	A	Pointeur du sprite 5 (les 15 bits de poids faible)
SPR6PTH +	138	W	A	Pointeur du sprite 6 (les 3 bits de poids fort)
SPR6PTL +	13A	W	A	Pointeur du sprite 6 (les 15 bits de poids faible)
SPR7PTH +	13C	W	A	Pointeur du sprite 7 (les 3 bits de poids fort)
SPR7PTL +	13E	W	A	Pointeur du sprite 7 (les 15 bits de poids faible)
SPR0POS %	140	W	A D	Données de la position initiale vert.-horiz. du sprite 0
SPR0CTL %	142	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 0
SPR0DATA %	144	W	D	Registre de données de l'image A du sprite 0
SPR0DATB %	146	W	D	Registre de données de l'image B du sprite 0
SPR1POS %	148	W	A D	Données de la position initiale vert.-horiz. du sprite 1
SPR1CTL %	14A	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 1
SPR1DATA %	14C	W	D	Registre de données de l'image A du sprite 1
SPR1DATB %	14E	W	D	Registre de données de l'image B du sprite 1
SPR2POS %	150	W	A D	Données de la position initiale vert.-horiz. du sprite 2
SPR2CTL %	152	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 2
SPR2DATA %	154	W	D	Registre de données de l'image A du sprite 2
SPR2DATB %	156	W	D	Registre de données de l'image B du sprite 2
SPR3POS %	158	W	A D	Données de la position initiale vert.-horiz. du sprite 3
SPR3CTL %	15A	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 3
SPR3DATA %	15C	W	D	Registre de données de l'image A du sprite 3
SPR3DATB %	15E	W	D	Registre de données de l'image B du sprite 3
SPR4POS %	160	W	A D	Données de la position initiale vert.-horiz. du sprite 4
SPR4CTL %	162	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 4
SPR4DATA %	164	W	D	Registre de données de l'image A du sprite 4
SPR4DATB %	166	W	D	Registre de données de l'image B du sprite 4
SPR5POS %	168	W	A D	Données de la position initiale vert.-horiz. Du sprite 5
SPR5CTL %	16A	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 5
SPR5DATA %	16C	W	D	Registre de données de l'image A du sprite 5
SPR5DATB %	16E	W	D	Registre de données de l'image B du sprite 5

NOM	ADR	R/W	PUCE	FONCTION
SPR6POS %	170	W	A D	Données de la position initiale vert.-horiz. du sprite 6
SPR6CTL %	172	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 6
SPR6DATA %	174	W	D	Registre de données de l'image A du sprite 6
SPR6DATB %	176	W	D	Registre de données de l'image B du sprite 6
SPR7POS %	178	W	A D	Données de la position initiale vert.-horiz. du sprite 7
SPR7CTL %	17A	W	A D	Données de contrôle et de position d'arrêt verticale du sprite 7
SPR7DATA %	17C	W	D	Registre de données de l'image A du sprite 7
SPR7DATB %	17E	W	D	Registre de données de l'image B du sprite 7
COLOR00	180	W	D	Table de couleur 00
COLOR01	182	W	D	Table de couleur 01
COLOR02	184	W	D	Table de couleur 02
COLOR03	186	W	D	Table de couleur 03
COLOR04	188	W	D	Table de couleur 04
COLOR05	18A	W	D	Table de couleur 05
COLOR06	18C	W	D	Table de couleur 06
COLOR07	18E	W	D	Table de couleur 07
COLOR08	190	W	D	Table de couleur 08
COLOR09	192	W	D	Table de couleur 09
COLOR10	194	W	D	Table de couleur 10
COLOR11	196	W	D	Table de couleur 11
COLOR12	198	W	D	Table de couleur 12
COLOR13	19A	W	D	Table de couleur 13
COLOR14	19C	W	D	Table de couleur 14
COLOR15	19E	W	D	Table de couleur 15
COLOR16	1A0	W	D	Table de couleur 16
COLOR17	1A2	W	D	Table de couleur 17
COLOR18	1A4	W	D	Table de couleur 18
COLORl9	1A6	W	D	Table de couleur 19
COLOR20	1A8	W	D	Table de couleur 20
COLOR21	1AC	W	D	Table de couleur 21
COLOR22	1AC	W	D	Table de couleur 22
COLOR23	1AE	W	D	Table de couleur 23
COLOR24	1B0	W	D	Table de couleur 24
COLOR25	1B2	W	D	Table de couleur 25
COLOR26	1B4	W	D	Table de couleur 26
COLOR26	1B6	W	D	Table de couleur 27
COLOR28	1B8	W	D	Table de couleur 28
COLOR29	1BA	W	D	Table de couleur 29
COLOR30	1BC	W	D	Table de couleur 30
COLOR31	1BE	W	D	Table de couleur 31
RESERVED	1110X			
RESERVED	111lX			
NO-OP(NULL)	1FE			

APPENDICE C

LISTE DE L'IMPLANTATION DES BROCHES DES PUCES SPECIALISEES

NOTE : * Veut dire signal zéro actif ['low signal'].

ASSIGNATION DES BROCHES D'AGNUS

BROCHE #	DESIGNATION	FONCTION	DEFINITION
01-09	D8-D0	Lignes du bus de données 8-0	E/S
10	VCC	+5 Volts	E
11	RES*	Réinitialisation du système	E
12	INT3*	Interruption de niveau 3	S
13	DMAL	Ligne pour demande DMA	E
14	BLS*	Ralentir 'blitter'	E
15	DBR*	Demande du bus de données	S
16	ARW*	Ecriture d'Agnus en RAM	S
17-24	RGA8-RGA1	Bus d'adresse registre 8-1	E/S
25	CCK	Signal d'horloge couleur	E
26	CCKQ	Délai signal d'horl. coul.	E
27	VSS	Masse	E
28-36	DRA0-DRA8	Bus d'adr. Mém. dynamique 0-8	S
37	LP*	Entrée crayon optique	E
38	VSY*	Synchro verticale	E/S
39	CSY*	Synchro composite	S
40	HSY*	Sunchro horizontale	E/S
41	VSS	Masse	E
42-48	D15-D9	Lignes du bus de données 15-9	E/S

ASSIGNATION DES BROCHES DE DENISE

BROCHE #	DESIGNATION	FONCTION	DEFINITION
01-07	D6-D0	Lignes du bus de données 6-0	E/S
08	M1H	Signal horizontal souris 1	E
09	M0H	Signal horizontal souris 2	E
10-17	RGA8-RGA1	Bus d'adresse registre 8-1	E
18	BURST*	Impulsion synchronisateur coul.	S
19	VCC	+5 Volts	E
20-23	R0-R3	Bits vidéo pour le rouge 0-3	S
24-27	B0-B3	Bits vidéo pour le bleu 0-3	S
28-31	G0-G3	Bits vidéo pour le vert 0-3	S
32	N/C	Non connectée	N/C
33	ZD*	Indicateur vidéo externe	S
34	N/C	Non connecté	N/C
35	7M	7,15909 MHz	E
36	CCK	Signal d'horloge couleur	E
37	VSS	Masse	E
38	M0V	Signal vertical souris 0	E
39	M1V	Signal vertical souris 1	E
40-48	D15-D7	Lignes bus de données 15-7	E/S

ASSIGNATION DES BROCHES DE PAULA

BROCHE #	DESIGNATION	FONCTION	DEFINITION
01-07	D8-D2	Lignes du bus de données 8-2	E/S
08	VSS	Masse	E
09-10	D1-D0	Lignes du bus de données 1-0	E/S
11	RES*	Réinitialisation du système	E
12	DMAL	Ligne pour demande DMA	S
13-15	IPL0*-IPL2	Lignes d'interruption 0-2	S
16	INT2*	Interruption de niveau 2	E
17	INT3*	Interruption de niveau 3	E
18	INT6*	Interruption de niveau 6	E
19-26	RGA8-RGA1	Bus d'adresse registre 8-1	E
27	VCC	+5 Volts	E
28	CCK	Signal d'horloge couleur	E
29	CCKQ	Délai signal d'horl. coul.	E
30	AUDB	Signal audio droit	S
31	AUDA	Signal audio gauche	S
32	POT0X	Potentiomètre 0X	E/S
33	POT0Y	Potentiomètre 0Y	E/S
34	VSSANA	Masse analogique	E
35	POT1X	Potentiomètre 1X	E/S
36	POT1Y	Potentiomètre 1Y	E/S
37	DKRD*	Lecture de données de la disquette	E
38	DKWD*	Ecriture de données sur disquette	S
39	DKWE	Validation écriture disquette	S
40	TXD	Données transmission série	S
41	RXD	Données réception série	E
42-48	D15-D9	Lignes du bus de données 15-9	E/S

ASSIGNATION DES BROCHES DE 'FAT AGNUS'

BROCHE #	DESIGNATION	FONCTION	DEFINITION
01-14	RD15-RD2	Lignes bus registre 15-2	E/S
17	INT3*	Interruption 'blitter' prêt	S
18	DMAL	Demande DMA audio/disquette	E
18	RD1	Ligne bus registre 1	E/S
18	RST*	Réinitialisation	E
19	BLS*	Ralentir 'blitter'	E
20	DBR*	Demande bus de données	S
21	RRW	Ecriture/lecture en mém.dynam.	S
22	PRW	Ecriture/lecture processeur	E
23	RGEN*	Validation RG	E
24	AS*	Adresse 'strobe'	E
25	RAMEN*	Validation mémoire vive	E
26-33	RGA8-RGA1	Bus d'adresse registre 8-1	S
34	28MHZ	Horloge principale	E
35	XCLR	Horloge alternative	E
36	XCLKEN*	Validation horloge princ.	E
37	CDAC*	Horloge 7 MHz décalée inversée	S
38	7MHZ	Horloge 28 MHz divisée par 4	S
39	CCRQ	Délai signal d'horl. coul.	S
40	CCR	Signal d'horloge	S
41	TEST	Test - accès registres	E
43-51	MA0-MA8	Lignes de bus sortie 0-8	S
52	LDS*	'Strobe' données de poids faible	E
53	UDS*	'Strobe' données de poids fort	E
54	CASL*	'Strobe' adresse colonne inférieure	S
55	CASU*	'Strobe' adresse colonne supérieure	S
56	RAS1*	'Strobe' adresse ligne un	S
57	RAS0*	'Strobe' adresse ligne zéro	S
59-77	A19-A1	Lignes bus d'adresse 19-1	E
78	LP*	Crayon optique	S
79	VSY*	Synchro verticale	E/S
80	CSY*	Synchro vidéo composite	S
81	HSY*	Synchro horizontale	E/S
84	RD0	Ligne bus registre 0	E/S

APPENDICE D

MAPPE MEMOIRE DU SYSTEME

Une mappe mémoire 'software' véritable, qui montrerait l'utilisation par le système des différentes portions de mémoire vive et de l'espace libre, n'est pas fournie et n'est pas possible à fournir avec l'Amiga. Toute la mémoire est réservée de manière dynamique par le gestionnaire de mémoire et les adresses actuelles peuvent changer d'une version à l'autre, d'une machine à l'autre ou d'une initialisation à l'autre (pour des détails, cf. la fonction AllocMem() d'Exec. Pour trouver les adresses des structures du système, le 'software' doit employer des procédures d'accès déterminées, en commençant par prélever l'adresse de la bibliothèque Exec ['exec.library'] à l'adresse de base 4; la seule adresse mémoire absolue du système. Tout le 'software' a été écrit pour pouvoir être chargé et rechargé n'importe où en mémoire par le chargeur. Ce qui suit n'est qu'une mise en page générale des zones de mémoire à l'intérieur de la présente génération d'ordinateurs Amiga.

PLAGE D'ADRESSES	NOTES
000000-03FFFF	256 Ko de mémoire vive 'chip'
040000-07FFFF	256 Ko de mémoire vive 'chip' (carte en option)
080000-0FFFFF	512 Ko de mémoire vive 'chip' étendue (jusqu'à Mo)
100000-1FFFFF	Réservé. Ne pas s'en servir.
200000-9FFFFF	Premier espace auto-config 8 Mo.
A00000-BEFFFF	Réservé. Ne pas s'en servir.
BFD000-BFDF00	8520-B (accès seulement aux adresses paires)
BFE001-BFEF01	8520-A (accès seulement aux adresses impaires)
	Les chiffres soulignés choisissent le registre auquel on peut avoir accès, parmi les 16 registres internes. Cf. Appendice F.
C00000-DFEFFF	Réservé. Ne pas s'en servir.
C00000-D7FFFF	Mémoire d'extension interne.
D80000-DBFFFF	Réservé. Ne pas s'en servir.
DC0000-DCFFFF	Horloge temps réel.
DFF000-DFFFFF	Registres des puces. Cf. Appendice A et Appendice B.
E00000-E7FFFF	Réservé. Ne pas s'en servir.
E80000-E8FFFF	Espace auto-config. C'est ici qu'apparaissent les cartes avant que le système ne les reloge à leur adresse définitive.
E90000-EFFFFF	Deuxième espace auto-config (d'ordinaire pour des cartes d'E/S de 64 Ko).
F00000-FBFFFF	Réservé. Ne pas s'en servir.
FC0000-FFFFFF	ROM du Système d'exploitation 256 Ko.

APPENDICE E

LES INTERFACES

Cette appendice se compose de quatre parties distinctes, qui sont toutes en rapport avec la façon dont l'Amiga communique avec le monde extérieur.

La première partie précise les broches de sortie des connecteurs accessibles de l'extérieur et la puissance disponible de chaque connecteur. Cette partie, par conséquent, ne fournit pas d'indications sur le chargement ou la synchronisation.

La deuxième partie décrit brièvement la fonction de ces broches dont l'usage peut ne pas paraître évident.

La troisième partie contient la liste des connexions d'un certain nombre de connecteurs internes, principalement celui de l'unité de disquette.

La quatrième partie précise combien sont variés les signaux se rapportant aux ports disponibles du 8520. Ces informations donnent la possibilité au programmeur d'établir un lien entre les adresses des ports et des éléments du monde extérieur (ou des signaux de contrôle internes) qui pourraient être concernés.

La troisième et la quatrième partie sont de premier abord destinées aux programmeurs du système et, d'une manière générale, ne devraient pas être utilisées par des programmeurs d'applications.

D'ordinaire, le 'software' des systèmes est configuré pour gérer la mise en place de signaux bien précis et ne s'occupe aucunement de la manière dont pourraient changer, physiquement parlant, les connexions. En d'autres termes, si vous possédez une version du 'software' du système qui concorde avec le niveau de révision de la machine (normalement, il s'agit d'une condition qui est vraie), lorsque vous demandez qu'un bit particulier soit mis à 1, peu vous importe de savoir à quel port est connecté le bit en question. Ainsi, les programmeurs d'applications se fieront plus à la documentation du système qu'ils ne s'adresseront directement aux ports.

NOTE

Dans un système d'exploitation multitâche, plusieurs tâches différentes ont la possibilité de se mettre sur les rangs pour utiliser les ressources du système. Les programmeurs d'applications se devraient de suivre les règles établies pour être à même de garantir la compatibilité de leurs programmes avec le système.

********** 1ère PARTIE - CONNECTEURS SUR LE MONDE EXTERIEUR **********

Voici une liste des connexions de l'Amiga avec le monde extérieur.

RS232 et port MIDI

BROCHE	RS232	A1000	A500/ A2000	CBM PCs	HAYES	DESCRIPTION
1	GND	GND	GND	GND	GND	MASSE DE PROTECTION
2	TXD	TXD	TXD	TXD	TXD	DONNEE TRANSMISE
3	RXD	RXD	RXD	RXD	RXD	DONNEE RECEPTIONNEE
4	RTS	RTS	RTS	RTS	-	DEMANDE D'EMISSION
5	CTS	CTS	CTS	CTS	CTS	PRET A EMETTRE
6	DSR	DSR	DSR	DSR	DSR	DONNEE PRETE A ENVOYER
7	GND	GND	GND	GND	GND	MASSE DE REFERENCE
8	CD	CD	CD	DCD	DCD	MODEM DETECTE
9	-	-	+12v	+12v	-	ALIM. +12 VOLTS
10	-	-	-12v	-12v	-	ALIM. -12 VOLTS
11	-	-	AUDO	-	-	SORTIE AUDIO
12	S.SD	-	-	-	SI	INDICATEUR DE VITESSE
13	S.CTS	-	-	-	-	
14	S.TXD	-5Vdc	-	-	-	ALIM. -5 VOLTS
15	TXC	AUDO	-	-	-	SORTIE AUDIO DE L'AMIGA
16	S.RXD	AUDI	-	-	-	ENTREE AUDIO DE L'AMIGA
17	RXC	EB	-	-	-	HORL. TAMPON 716 KHZ POUR PORT
18	-	INT2*	AUDI	-	-	LIGNE D'INTERRUPTION DE L'AMIGA
19	S.RTS	-	-	-	-	
20	DTR	DTR	DTR	DTR	DTR	TERMINAL PRET A RECEVOIR
21	SQD	+5	-	-	-	ALIM. +5 VOLTS
22	RI	-	RI	RI	RI	INDICATEUR D'APPEL
23	SS	+12Vdc	-	-	-	ALIM. +12 VOLTS
24	TXC1	C2*	-	-	-	HORLOGE 3.58 MHZ
25	-	RESB*	-	-	-	REINIT. SYSTEME TAMPONNEE

PORT PARALLELE (CENTRONICS)

BROCHE	1000	500/2000	Commodore PC
1	DRDY*	STROBE*	STROBE*
2	Data 0	Data 0	Data 0
3	Data 1	Data 1	Data 1
4	Data 2	Data 2	Data 2
5	Data 3	Data 3	Data 3
6	Data 4	Data 4	Data 4
7	Data 5	Data 5	Data 5
8	Data 6	Data 6	Data 6
9	Data 7	Data 7	Data 7
10	ACK*	ACK*	ACK*
11	BUSY (data)	BUSY	BUSY
12	POUT (cl)	POUT	POUT
13	SEL	SEL	SEL
14	GND +5v	pullup	AUTODXT
15	GND	NC	ERROR*
16	GND	RESET*	INIT*
17	GND	GND	SLCT IN*
18-22	GND	GND	GND
23	+5	GND	GND
24	NC	GND	GND
25	Reset*	GND	GND

CLAVIER ... RJ11

	A1000	A2000
1	+5 Volts	KCLK
2	HORLOGE	KDAT
3	DONNEE	NON CONNECTEE
4	MASSE	MASSE
5	-	+5 Volts

Ne s'applique pas à l'A500.

Video ... MALE TYPE D-SUB 23 BROCHES

Pour A500, A1000, A2000 jusqu'à un changement éventuel.

1	XCLK*	13	GNDRTN (Return for XCLKEN*)
2	XCLKEN*	14	ZD*
3	RED	15	C1*
4	GREEN	16	GND
5	BLUE	17	GND
6	DI	18	GND
7	DB	19	GND
8	DG	20	GND
9	DR	21	A1000/2000 -5 VOLT POWER
10	CSYNC*		A500 -12 VOLT POWER
11	HSYNC*	22	+12 VOLT POWER
12	VSYNC*	23	+5 VOLT POWER

Moniteur RF ... 8 BROCHES TYPE DIN (J2) seulement pour l'A1000

1	NON CONNECTEE
2	MASSE
3	SIGNAL AUDIO GAUCHE
4	COMPARATEUR VIDEO
5	MASSE
6	NON CONNECTEE
7	ALIM. +12 VOLTS
8	SIGNAL AUDIO DROIT

UNITE DISQUETTES EXTERNE ... FEMELLE TYPE D-SUB 23 BROCHES

Pour A500, A1000 et A2000 avec des différences remarquées sur l'A2000

1	DISQUETTE PRETE*	13	SELECT. FACE*
2	LECTURE DONNEES*	14	PROTECT. EN ECRITURE*
3	MASSE	15	PISTE0*
4	MASSE	16	AUTORISAT. ECRITURE*
5	MASSE	17	ECRITURE DONNEES*
6	MASSE	18	DEPLAC. TETE*
7	MASSE	19	DIRECT. DEPL. TETE
8	MOTEUR E.S./H.S.*	20	SELECT. DISQ. 3*
9	SELECT. DISQ. 2* (1)		A2000 inutilisé (1)
	A2000 SELECT. 3* (1)	21	SELECT. DISQ. 1*
10	ARRET MOTEUR*		A2000 select. disq. 2* (1)
11	CHANG. DISQUETTE*	22	DEBUT CYLINDRE*
12	ALIM. +5 VOLTS	23	ALIM. +12 VOLTS
12	ALIM. +5 VOLTS		

(1) SEL1B* n'est pas le lecteur de disquettes 1, mais plutôt le premier lecteur externe. Toutes les lignes de sélection ne pourront être implémentées.

RAMEX...PLAT 60 BROCHES ECARTEMENT (0,156 POUCES) TYPE (P1) A1000 seulement

---

1	masse	A	masse
2	D15	B	D14
3	+5 Volts	C	+5 Volts
4	D12	D	D13
5	masse	E	masse
6	D11	F	D10
7	+5 Volts	H	+5 Volts
8	D8	J	D9
9	masse	K	masse
10	D7	L	D6
11	+5 Volts	M	+5 Volts
12	D4	N	D5
13	masse	P	masse
14	D3	R	D2
15	+5 Volts	S	+5 Volts
16	D0	T	D1
17	masse	U	masse
18	DRA4	V	DRA3
19	DRA5	W	DRA2
20	DRA6	X	DRA1
21	DRA7	Y	DRA0
22	masse	Z	masse
23	RAS*	AA	RRW*
24	masse	BB	masse
25	masse	CC	masse
26	CASU0*	DD	CASU1*
27	masse	EE	masse
28	CASL0*	FF	CASL1*
29	+5 Volts	HH	+5 Volts
30	+5 Volts	JJ	+5 Volts

EXTENSION ... PLAT 86 BROCHES ECARTEMENT (0,1 POUCES) (P2)

PIN	A500	A1000	A2000	A2000b	FONCTION
1	x	x	x	x	masse
2	x	x	x	x	masse
3	x	x	x	x	masse
4	x	x	x	x	masse
5	x	x	x	x	Alim. 5 Volts cc
6	x	x	x	x	Alim. 5 Volts cc
7	x	x	x	x	Non Connectée
8	x	x	x	x	Alim. -5 Volts cc
9	x	x			Non Connectée
		x	x		Horloge 28 MHz
10	x	x	x	x	+12 Volts cc
11	x	x	x		Non Connectée
				x	/COPCFG (Configuration hors serv.)
12	x	x	x	x	CONFIG EN SERV., à la masse
13	x	x	x	x	masse
14	x	x	x	x	/C3 Horloge
15	x	x	x	x	CDAC Horloge
16	x	x	x	x	/C1 Horloge
17	x	x	x	x	/OVR
18	x	x	x	x	RDY
19	x	x	x	x	/INT2
20		x			/PALOPE
	x		x		Non Connectée
				x	/BOSS
21	x	x	x	x	A5
22	x	x	x	x	/INT6
23	x	x	x	x	A6
24	x	x	x	x	A4
25	x	x	x	x	masse
26	x	x	x	x	A3
27	x	x	x	x	A2
28	x	x	x	x	A7
29	x	x	x	x	A1
30	x	x	x	x	A8
31	x	x	x	x	FC0
32	x	x	x	x	A9
33	x	x	x	x	FC1
34	x	x	x	x	A10
35	x	x	x	x	FC2
36	x	x	x	x	A11
37	x	x	x	x	masse
38	x	x	x	x	A12
39	x	x	x	x	A13
40	x	x	x	x	/IPL0
41	x	x	x	x	A14
42	x	x	x	x	/IPL1
43	x	x	x	x	A15
44	x	x	x	x	/IPL2
45	x	x	x	x	A16
46	x	x	x	x	BEER*
47	x	x	x	x	A17
48	x	x	x	x	/VPA
49	x	x	x	x	masse
50	x	x	x	x	Horloge gestion [secteur] "E Clock"

EXTENSION ... PLAT 86 BROCHES (.1) (P2) (suite)

PIN	A500	A1000	A2000	A2000b	FONCTION
51	x	x	x	x	/VMA
52	x	x	x	x	A18
53	x	x	x	x	RST
54	x	x	x	x	Al9
55	x	x	x	x	/HLT
56	x	x	x	x	A20
57	x	x	x	x	A22
58	x	x	x	x	A21
59	x	x	x	x	A23
60	x	x	x		/BR
				x	/CBR
61	x	x	x	x	masse
62	x	x	x	x	/BGACK
63	x	x	x	x	D15
64	x	x	x		/BG
				x	/CBG
65	x	x	x	x	D14
66	x	x	x	x	/DTACK
67	x	x	x	x	D13
68	x	x	x	x	R/W
69	x	x	x	x	D12
70	x	x	x	x	/LDS
71	x	x	x	x	D11
72	x	x	x	x	/UDS
73	x	x	x	x	masse
74	x	x	x	x	/AS
75	x	x	x	x	D0
76	x	x	x	x	D10
77	x	x	x	x	D1
78	x	x	x	x	D9
79	x	x	x	x	D2
80	x	x	x	x	D8
81	x	x	x	x	D3
82	x	x	x	x	D7
83	x	x	x	x	D4
84	x	x	x	x	D6
85	x	x	x	x	masse
86	x	x	x	x	D5

JOYSTICK ... mâle type D-SUB 9 BROCHES

USAGE	JOYSTICK	SOURIS
1	AVANT*	(IMPULSION VERT.)
2	ARRIERE*	(IMPULSION HORIZ.)
3	GAUCHE*	(IMPULSION QUADRATURE VERT.)
4	DROITE*	(IMPULSION QUADRATURE HORIZ.)
5	POTENT. X	(ou bouton 3 ... si utilisé)
6	BOUTON FEU*	(ou bouton 1)
7	+5 VOLTS	
8	MASSE	
9	POTENT. Y	(ou bouton 2)

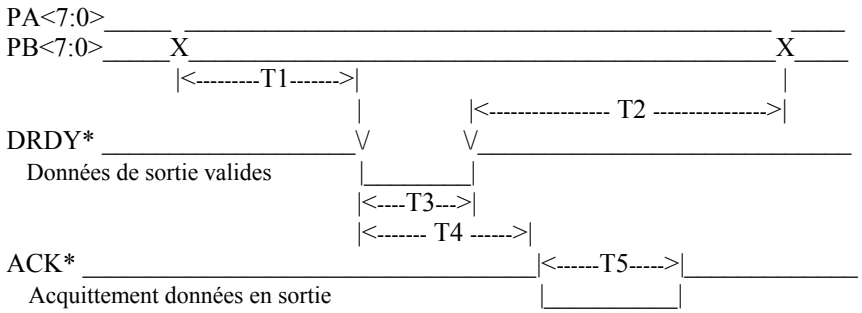
NORME POUR LE CONNECTEUR D'INTERFACE PARALLELE

Le connecteur 25 broches type D-Sub avec des broches (D-SUB 25 broches mâle pour l'A1000, femelle pour l'A500/A2000 et les compatibles IBM) sur la face arrière de l'Amiga, sert nominalement d'interface pour des imprimantes parallèle. Dans ce cas, les données circulent de l'Amiga vers l'imprimante. On peut employer cette surface en entrée ou pour des transferts de données bi-directionnels. Son implémentation est similaire à celle du connecteur Centronics, mais l'assignation des broches et les caractéristiques de pilotage diffèrent de manière importante de celles indiquées dans la norme (cf. Assignation des Broches). L'appellation des signaux correspond à celle utilisée ailleurs dans cette appendice, lorsque cela est possible.

ASSIGNATION DES BROCHES DU CONNECTEUR PARALLELE TYPE (J8)

NOM	DIR	NOTES
DRDY*	S	[DONNEE VALIDE]. Signal de données de sortie prêtes pour le Périphérique parallèle en mode sortie, utilisé conjointement à ACK* (acquittement, broche 10) pour une liaison asynchrone bi-directionnelle. Des fonctions telles que l'entrée de données à partir de l'Amiga sont acceptées en mode entrée (similaire à ACK* -acquittement, en mode sortie). Cf. diagrammes de synchronisation à la section suivante.
D0	E/S	+
Dl	E/S	|
D2	E/S	| [D0-D7 = DONNEES VALIDES].
D3	E/S	| D0-D7 englobent un bus bi-directionnel 8 bits pour
D4	E/S	| communiquer avec les périphériques parallèle,
D5	E/S	| nominalement, une imprimante.
D6	E/S	|
D7	E/S	+
ACK*	E	[ACQUITTEMENT]. Acquittement des données en sortie à partir d'un périphérique parallèle en mode sortie, employé conjointement à DRDY* (DONNE VALIDE, sur la broche 1) pour une liaison bi-directionnelle asynchrone. Des fonctions telles que l'entrée de données prêtes à partir d'un périphérique parallèle sont acceptées en mode entrée (similaire à DRDY*, donnée valide, en mode sortie). Le 8520 peut être programmé pour générer une interruption de niveau 2 en direction du 68000 lorsque l'entrée ACK* devient active.
BUSY	E/S	[IMPRIMANTE OCCUPEE]. Il s'agit d'une broche d'E/S multi-usage connectée aussi à la broche d'E/S de données série (horloge série sur la broche 12). Note : employée nominalement pour indiquer que le tampon d'imprimante est plein.
POUT	E/S	[PAPIER MANQUANT]. Il s'agit d'une broche d'E/S générique reliée à la broche d'E/S de l'horloge série (données série sur la broche 11). Note : employée nominalement pour indiquer qu'il n'y a plus de papier dans l'imprimante.
SEL	E/S	[SELECTION]. C'est une broche d'E/S multi-usage. Note : nominalement, une sortie destinée à une sélection à partir du périphérique parallèle vers l'Amiga. Sur l'A500/A2000, elle est aussi partagée avec le signal "indicateur d'appel" de l'RS232.
RESET*	S	[REINITIALISATION]. Réinitialisation du système Amiga.

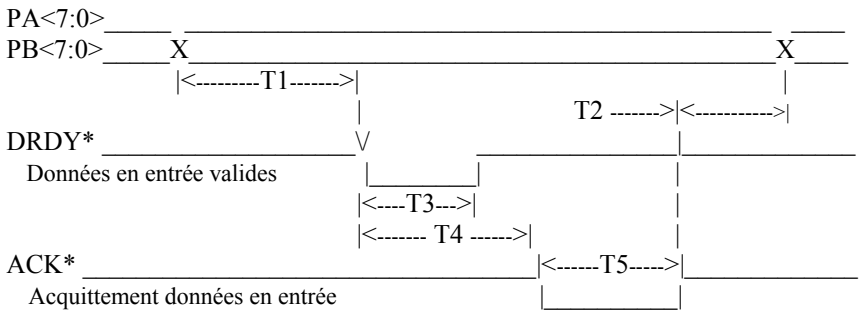
SYNCHRONISATION DU CONNECTEUR D'INTERFACE PARALLELE, CYCLE DE SORTIE



	Micro-secondes Min	Typ	Max	
T1:	4.3	-x-	5.3	Délai du temps de mise en route des données prêtes en sortie.
T2:	nsp	-x-	upc	Temps de conservation des données en sortie.
T3:	nsp	1.4	nsp	Largeur prête pour les données en sortie.
T4:	0	-x-	upc	Prêt à acquitter le délai.
T5:	nsp	-x-	upc	Largeur de l'acquittement.

nsp - non spécifié
ucp - sous contrôle du programme

SYNCHRONISATION DU CONNECTEUR D'INTERFACE PARALLELE, CYCLE D'ENTREE



	Micro-secondes Min	Typ	Max	
T1:	0	-x-	upc	Temps de mise en route des données en entrée.
T2:	nsp	-x-	upc	Temps de conservation des données en entrée.
T3:	nsp	-x-	ups	Largeur prête pour les données en entrée.
T4:	upc	-x-	upc	Données en entrée prêtes pour acquitter le délai de données.
T5:	nsp	1.4	nsp	Largeur de l'acquittement des données en entrée.

nsp - non spécifié
upc - sous contrôle du programme

NORME POUR LE CONNECTEUR D'INTERFACE SERIE

Ce connecteurs 25 broches de type Sub-D avec douilles (femelle) est utilisé pour interfacer les signaux RS-232-C standard. L'appellation des signaux correspond à celle employée à d'autres endroits de cette appendice.
Évitez les câbles de connecteurs "en nappe".

MISE EN GARDE : les broches du connecteur RS-232 autres que les broches standard, décrites ci-dessous, peuvent être connectées à une alimentation ou à d'autres signaux RS-232 non standard. Si vous confectionnez des câbles pour ce connecteur, câblez seulement les broches que vous utiliserez réellement dans une application précise.
Évitez les câbles de connecteurs "en nappe".

ASSIGNATION DES BROCHES DU CONNECTEUR D'INTERFACE SERIE TYPE (J6)

RS-232-C

NOM	DIR	STD	NOTES
FGND		o	Masse de protection -- ne pas relier à la masse du signal
TXD	S	o	Donnée transmise
RXD	E	o	Donnée réceptionnée
RTS	S	o	Demande de transmission
CTS	E	o	Prêt à transmettre
DSR	E	o	Donnée prête à envoyer
GND		o	Masse du signal -- ne pas relier à la masse de protection
CD	E	o	Modem détecté
-5V		n*	50 milli maximum *** ATTENTION -5 Volts ***
AUDO	S	n*	Sortie audio en provenance du port gauche (canaux 0, 3), prévue pour envoyer un signal audio au modem.
AUDI	E	n*	Entrée audio pour le port droit (canaux 1, 2), prévue pour recevoir le signal audio en provenance du modem; cette entrée se mélange avec la sortie analogique de droite (canaux 1, 2). Elle n'est pas numérisée et n'est aucunement utilisée par l'ordinateur.
DTR	S	o	Terminal prêt à réceptionner des données.
RI	E	o	Indicateur d'appel (seulement A500/A2000) partagé avec le signal "sélection" de l'imprimante.
RESB*	S	n*	Réinitialisation du sytème de l'Amiga.

NOTES :

n* : cf. la mise en garde ci-dessus.
Cf. 1ère PARTIE de cette appendice pour la numérotation des broches.

SYNCHRONISATION DU CONNECTEUR D'INTERFACE SERIE

La fréquence de travail maximale est de 19,2 KHz. Référez-vous au standard EIA pour les normes sur l'installation et le fonctionnement du RS-232-C. Un taux de 31,25 KHz sera accepté pour une utilisation avec l'adaptateur MIDI.

Les signaux de contrôle pour le modem (CTS, RTS, DTR, DSR, CD) sont totalement sous le contrôle du 'software'. Les lignes de contrôle du modem ne possèdent pas le 'hardware' pour qu'on puisse les affecter et sont totalement asynchrones avec TXD et RXD.

CARACTERISTIQUES ELECTRIQUES DU CONNECTEUR D'INTERFACE SERIE

SORTIES	MIN	TYP	MAX		
Vo(-):	-13.2	-x-	-2.5	V	Fourchette pour tension de sortie négative
Vo(+):	8.0	-x-	13.2	V	Fourchette pour tension de sortie positive
Io:	-x-	-x-	10.0	ma	Courant de sortie

ENTREES	MIN	TYP	MAX		
Vi(+):	3.0	-x-	25.0	V	Fourchette pour tension d'entrée positive
Vi(-):	-25.0	-x-	0.5	V	Fourchette pour tension d'entrée négative
Vhy:	-x-	1.0	-x-	V	Tension d'hysteresis d'entrée
Ii:	0.3	-x-	10.0	ma	Courant d'entrée

Les entrées non connectées sont interprétées également comme tensions d'entrée positives.

NORME POUR LE CONNECTEUR D'INTERFACE DU CONTROLEUR DES PERIPHERIQUES DE JEUX

Les deux connecteurs 9 broches de type Sub-D avec broches (mâles) sont utilisés pour interfacer quatre sortes de périphériques :

1. Souris ou boule roulante, 3 boutons maxi.

2. Manette de commande numérique, 2 bouton maxi.

3. Proportionnel (potentiomètre ou manette de commande proportionnelle), 2 boutons maxi.

4. Crayon optique, y compris le bouton "crayon appuyé à l'écran".

Nous parlerons de l'alignement des broches du connecteur dans les sections prévues pour les besoins similaires du traitement du 'software' et/ou du 'hardware'. L'appellation des signaux suit celle employée ailleurs dans cette appendice, quand cela a été possible.

J11 est le connecteur du port droit du contrôleur (JOY1DAT, POT1DAT).
J12 est le connecteur du port gauche du contrôleur (JOY0DAT, POT0DAT).

NOTE : Bien qu'on puisse avoir accès directement à la plupart du 'hardware' dont nous parlons ci-dessous, on devrait accéder au 'hardware' par le 'software' en ROM. Ceci permettra de garder au 'hardware' à venir toute sa transparence.

INTERFACE DU CONTROLEUR DE JEUX AVEC LES ENTREES DE QUADRATURE DE SOURIS/BOULE ROULANTE

Une souris ou une boule roulante sont des périphériques qui transforment un mouvement sur une surface plane en un train d'impulsions. Les techniques de la quadrature sont employées pour préserver la direction et la magnitude du déplacement. Les registres JOY0DAT et JOY1DAT deviennent des registres de compteur contenant le déplacement y, dans l'octet de poids fort, et le déplacement x, dans l'octet de poids faible. Un déplacement provoque les actions suivantes :

Haut: y est décrémenté
Bas: y est incrémenté
Droite: x est incrémenté
Gauche: x est décrémenté

Pour déterminer un déplacement, on lit JOYxDAT deux fois en soustrayant les valeurs x et y correspondantes (attention, modulo 128 arithmétique). Remarquez que, si le chiffre se modifie au-delà de 127, la valeur de la distance et celle de la direction deviennent, toutes les deux, douteuses. Il existe une relation entre l'intervalle d'échantillonnage et la vitesse maximale (qui est une modification de la distance) qui peut être résolue de la manière suivante :

Vitesse < Distance(maxi) / TempsD'Echantillonage

Vitesse < √(DeltaX**2 + DeltaY**2) / TempsD'Echantillonage

Pour un Amiga avec un échantillonnage de souris de 200 par pouce, la vitesse maximale dans les deux directions X et Y sera :

Vitesse < (128 Unités * 1 pouce/200 Unités) /0,017 sec = 38 pouces/seconde

qui devraient suffire à la majorité des utilisateurs.

NOTE : Le 'software' de l'Amiga est conçu pour exécuter des cycles de mise à jour pour la souris pendant la suppression verticale. Les compteurs, horizontal et vertical, sont toujours valides et on a la possibilité de les lire à tout moment.

UTILISATION DES BROCHES DU CONNECTEUR POUR LES ENTREES DE QUADRATURE DE SOURIS/BOULE ROULANTE

BR.	MNEMONIQUE	DESCRIPTION	REGISTRE 'HARDWARE'/NOTES
1	V	Impulsions verticales	JOY[0/1]DAT<15:8>
2	H	Impulsions horizontales	JOY[0/1]DAT(7:0>
3	VQ	Impuls. de quadrature vert.	JOY[0/1]DAT<15:8>
4	HQ	Impuls. de quadrature horiz.	JOY[0/1]DAT<7:0>
5	UBUT*	Bouton souris inutilisé	Cf. entrées proport..
6	LBUT*	Bouton gauche de la souris	Cf. Bouton de mise à feu
7	+5V	+5 Volts, courant contrôlé	
8	Masse		
9	RBUT*	Bouton droit de la souris	Cf. entrées proport.

L'INTERFACE DU PORT DE JEU AVEC DES MANETTES DE COMMANDE NUMERIQUES

Une manette de commande est un périphérique avec quatre interrupteurs normalement ouverts disposés à 90 degrés. Les registres JOY[0/1]DAT deviennent des ports d'entrée des interrupteurs, une fois codés, comme nous l'indiquons ci-dessous :

Avant: bit9 xor bit#8
Gauche: bit9
Arrière: bit1 xor bit0
Droite: bit1

Les données sont codées pour faciliter le traitement de la souris/boule roulante.

NOTE : Les entrées de direction gauche et droite sont aussi prévues pour être des entrées, respectivement, pour le bouton gauche ou le bouton droit. Dans ce cas, les entrées avant et arrière ne seront pas utilisées, tandis que le bouton gauche et le bouton droit deviennent des entrées pour les boutons au lieu de devenir des entrées pour les manettes de commande.

Les registres JOY[0/1]DAT sont toujours valides et on a la possibilité de les lire à tout moment.

UTILISATION DES BROCHES DU CONNECTEUR POUR LES ENTREES NUMERIQUES DES MANETTES DE COMMANDE

BR.	MNEMONIQUE	DESCRIPTION	REGISTRE 'HARDWARE'/NOTES
1	FORWARD*	Inter. avant de la manette	JOY[0/1]DAT<9 xor 8>
2	BACK*	Inter. arrière de la manette	JOY[0/1]DAT(1 xor 0>
3	LEFT*	Inter. gauche de la manette	JOY[0/1]DAT<9>
4	RIGHT*	Inter. droit de la manette	JOY[0/1]DAT<1>
5	Inutilisé		
6	FIRE*	Bouton gauche de la souris	Cf. Bouton de mise à feu
7	+5V	max 125 milli, 200 milli en pointe	Total pour les deux ports
8	Masse		
9	Inutilisé		

INTERFACE DU PORT DE JEUX AVEC LES BOUTONS DE MISE A FEU

Normalement, les boutons de mise à feu sont des interrupteurs ouverts acheminés sur le PRA0 de l'adaptateur 8520 de la manière suivante :

PRA0 bit 7 - Port gauche du contrôleur de mise à feu [FIRE*]
PRA0 bit 6 - Port droit du contrôleur de mise à feu [FIRE*]

Avant de lire ce registre, les bits correspondants du registre de direction de données doivent être mis à zéro pour pouvoir définir le mode d'entrée :

DDRA0<7:6> mis à zéro d'une manière appropriée

NOTE: Ne dérangez pas la mise en place d'autres bits de DDRA0 (nous recommandons l'utilisation des appels en ROM).

Cinq boutons sont toujours valides et on a la possibilité de pouvoir les lire à tout instant.

UTILISATION DES BROCHES DU CONNECTEUR POUR LES ENTREES DES BOUTONS DE MISE A FEU

BR.	MNEMONIQUE	DESCRIPTION
1	-x-	
2	-x-	
3	-x-	
4	-x-	
5	-x-	
6	FIRE*	Bouton gauche souris/bouton de mise à feu
7	-x-	
8	masse	
9	-x-	

INTERFACE DU PORT DE JEUX AVEC LES CONTROLEURS PROPORTIONNELS

Des contrôleurs proportionnels avec un élément résistif (potentiomètre) linéaire en papier sont acceptés d'une valeur maximale de 528 KOhms (nous recommandons 470 K +/- 10%). Les registres JOY[0/1]DAT contiennent des valeurs numériques de traduction pour y, dans l'octet de poids fort, et pour x, dans l'octet de poids faible. Une valeur très élevée indique une résistance externe très importante. L'Amiga exécute une conversion intégrée analogique/numérique de la façon suivante :

1. Pendant les 7 (en NTSC) ou 8 (en PAL) premières lignes horizontales de l'affichage, les condensateurs analogiques en entrée se déchargent et les compteurs de position renvoyés dans les registres POT[0/1]DAT sont gardés à zéro.

   Pendant le reste du champ d'affichage, on permet aux condensateurs de se recharger par l'intermédiaire de l'élément résistif du périphérique externe de contrôle.

2. La tension qui augmente graduellement est sans cesse comparée à un niveau de référence interne tandis que le compteur garde la trace du nombre de lignes depuis la fin de la période de remise à zéro.

3. Dès que la tension d'entrée dépasse enfin le seuil interne pour un canal d'entrée donné, la valeur du compteur courant est mémorisée dans le registre POT[0/1]DAT qui correspond au canal en question.

4. Pendant l'intervalle de suppression verticale, le 'software' examine les valeurs résultantes du registre POT[0/1]DAT et les interprète en termes de position de la manette de commande.

NOTE : Les entrées POTY et POTX sont désignées, respectivement, en tant que "bouton droit de la souris" et "bouton souris non utilisé". Un interrupteur ouvert correspond à une forte résistance, un interrupteur fermé à une résistance faible. Les boutons sont aussi disponibles dans les registres POTGO et POTINP. Il est recommandé d'utiliser des appels en ROM pour une future compatibilité 'hardware'.

Il est important de réaliser que le contrôleur proportionnel est plus un périphérique de "pointage" qu'une entrée de position absolue. C'est au 'software' qu'il incombe de fournir un calibrage, des fonctions qui fassent la moyenne des valeurs et qui en limitent l'étendue, pour supporter les besoins de contrôle nécessaires à une application.

Typiquement, on lit dans les registres POT[0/1]DAT pendant la période de suppression vidéo, mais les registres PEUVENT être disponibles avant cette période.

UTILISATION DES BROCHES DU CONNECTEUR POUR LES ENTREES PROPORTIONNELLES

BR.	MNEMONIQUE	DESCRIPTION	REGISTRE 'HARDWARE'/NOTES
1	XBUT	Bouton supplémentaire	
2	Inutilisé		
3	LBUT*	Bouton gauche	Cf. manette de comm. numérique
4	RBUT*	Bouton droit	Cf. manette de comm. numérique
5	POTX	Entrée X analogique	POT[0/1]DAT<7:0>, POTGO, POTINP
6	Inutilisé		
7	+5 Volts	125 milli maxi, 200 milli en pointe	
8	Masse		
9	POTY	entrée Y analogique	POT[0,1]DAT<15:8>, POTGO, POTINP

INTERFACE DU PORT DE JEU AVEC UN CRAYON OPTIQUE

Un crayon optique est un périphérique opto-électronique dont la partie sensible à la lumière est placée à proximité d'un tube cathodique. A mesure que le faisceau d'électrons balaie au-dessous du crayon optique, il génère une impulsion du déclencheur qui peut être habilité à mémoriser la position horizontale et verticale du faisceau. Il n'existe pas un bit 'hardware' qui indique le déclenchement, mais il est possible de le déterminer en utilisant l'une des deux manières indiquées au Chapitre 8, "Le 'Hardware' des Interfaces".

D'ordinaire, on lit la position du crayon optique pendant la période de suppression vidéo, mais on PEUT en disposer avant celle-ci.

UTILISATION DES BROCHES DU CONNECTEUR POUR LES ENTREES DU CRAYON OPTIQUE

BR.	MNEMONIQUE	DESCRIPTION	REGISTRE 'HARDWARE'/NOTES
1	Inutilisé		
2	Inutilisé		
3	Inutilisé		
4	Inutilisé		
5	LPENPR*	Crayon optique appuyé	Cf. entrées proportionnelles
6	LPENTG*	Déclencheur du crayon opt.	VPOSR, VHPOSR
7	+5V	125 milli max., 200 milli en pointe	Pour les deux ports
8	Masse		
9	Inutilisé		

Note : le crayon optique, selon les constructeurs, peut être différent.

NORME POUR LE CONNECTEUR D'INTERFACE DE L'UNITE DE DISQUETTE EXYTERNE

Le connecteur 23 broches type Sub-D avec douilles (DB23S) à l'arrière de l'Amiga est, nominalement, utilisé comme interface pour les périphériques au standard MFM.

ASSIGNATION DES BROCHES DU CONNECTEUR AVEC LE LECTEUR DE DISQUETTES EXTERNE (J7)

BR.	NOM	DIR	NOTES
1	RDY*	E/S	Si le moteur est en service, il indique que la disquette est installée et prête à tourner. Si le moteur n'est pas en service, c'est alors le stade d'identification de mode. Voir ci-dessous.
2	DKRD*	E	Entrée données MFM vers l'Amiga.
3	MASSE		
4	MASSE		
5	MASSE		
6	MASSE		
7	MASSE		
8	MTRXD*	OC	Données moteur en service, rythmées dans la bascule de mise en service du moteur des lecteurs par la transition à un état actif de SELxB*. Le temps de mise en route garanti est de 1,4 micro-secondes. Le temps d'arrêt garanti est de 1,4 micro-secondes.
9	SEL2B*	OC	Sélection du lecteur 2.*
10	DRESB*	OC	Réinitialisation du système de l'Amiga. Les lecteurs devront réinitialiser les bascules de mise en service des moteurs et mettre à 1 les bascules de protection contre l'écriture.
11	CHNG*	E/S	Note : Nominalement utilisé comme entrée collecteur ouvert. La bascule de changement de disquette est mise à 1 à la mise sous tension ou s'il n'y a pas de disquette dans le lecteur. La bascule est réinitialisée, si on sélectionne le lecteur, et la tête de lecture/écriture progresse, mais celle-ci n'avance que si une disquette se trouve dans le lecteur.
12	+5 VOLTS		Maxi 270 milli; 410 milli en pointe. Si la tension est inférieure à 3,75 Volts, on demande aux lecteurs de réinitialiser les bascules de mise en service des moteurs et de mettre à 1 les bascules de protection contre l'écriture.
13	SIDEB*	S	Face 1 si actif, face 0 si non actif.
14	WPRO*	E/S	Validé par la sélection, disquette protégee en écriture.
15	TK0*	E/S	Validé par le lecteur sélectionné quand la tête de lecture/écriture est positionnée sur la piste 0.
16	DKWEB*	OC	Circuit d'écriture (validation) du lecteur.
17	DKWDB*	OC	Données MFM en sortie de l'Amiga.
18	STEPB*	OC	Le lecteur sélectionné avance d'un cylindre dans la direction indiquée par DIRB.
19	DIRB	OC	Direction d'avancement de la tête. Inactif pour avancer vers le centre de la disquette (pistes avec un numéro d'ordre élévé).
20	SEL3B*	OC	Sélectionne le lecteur 3. *
21	SEL1B*	OC	Sélectionne le lecteur 1. *
22	INDEX*	I/O	L'indice est une impulsion qui est générée à chaque rotation de la disquette, entre la fin et le début d'un cylindre. Le 8520 peut être programmé, conditionnellement, pour générer une interruption de niveau 6 en direction du 68000 si l'entrée INDEX* devenait active.
23	+12 VOLTS		Maxi. 160 milli; 540 milli en pointe.

* Note : Les lignes de sélection du lecteur sont décalées au fur et à mesure qu'elles passent à travers une chaîne de périphériques en boucle chainée. Ainsi, le signal qui se manifeste comme une sélection du lecteur 2 au premier lecteur, s'affiche comme une sélection du lecteur 1 au deuxième lecteur et ainsi de suite...

MODE D'IDENTIFICATION DU CONNECTEUR DE DISQUETTES EXTERNE

Un mode identification est présent pour la lecture du flux de données d'identification 32 bits série à partir d'un périphérique externe. Pour initialiser ce mode, on doit d'abord mettre le moteur en service, puis l'arrêter. Voir la broche 8, MTRXD*, pour des explications sur la mise en service et hors service du moteur. La transition d'un état à l'autre réinitialise le registre à décalage série.
Après une initialisation, on devrait laisser le signal SELxB* dans un état inactif. Ici, nous entrons dans une boucle où SELxB* est conduit à l'état actif, lit les données série en entrée sur RDY* (broche 1) et conduit SELxB* à un état inactif. Répétons cette boucle pour un total de 32 fois pour lire 32 bits de données. Le bit le plus significatif est réceptionné en premier.

IDENTIFICATIONS PRE-DEFINIES DU CONNECTEUR DE DISQUETTE EXTERNE

$0000 0000 - il n'y a pas de lecteur de disquette.
$FFFF FFFF - disquette 3 pouces et demi standard de l'Amiga
$5555 5555 - 48 TPI double densité, double face.

Comme pour les autres numéros d'ID des autres périphériques, les utilisateurs devront entrer en contact avec Commodore pour l'assignation d'un numéro d'ID.
Les données d'entrée série sont zéro actives et, par conséquent, on devra les inverser pour qu'elles correspondent à celles données ci-dessus.

LIMITATIONS DU CONNECTEUR DE DISQUETTES EXTERNE

1. La longueur totale du câble, y compris le bouclage de chaîne, ne devra pas dépasser un mètre.

2. Un maximum de trois périphériques externes pourra se trouver sur cette interface, mais des implémentations spécifiques pourraient supporter moins de périphériques externes.

3. Chaque périphérique devra fournir une résistance élévatrice de 1000 Ohms sur ces sorties pilotées vers l'Amiga par un périphérique à collecteur ouvert (broches 8-10, 16-21).

4. Le système fournit l'alimentation seulement pour le premier périphérique externe des chaînes de bouclage.

********** 3ème PARTIE - CONNECTEURS INTERNES **********

UNITE DISQUETTES INTERNE … CABLE 34 BROCHES (J10)

1	MASSE	18	DIRB
2	CHNG*	19	MASSE
3	MASSE	20	STEPB*
4	MTROD* (led)	21	MASSE
5	MASSE	22	DKWDB*
6	N.C.	23	MASSE
7	MASSE	24	DKWEB*
8	INDEX*	25	MASSE
9	MASSE	26	TK0*
10	SELOB*	27	MASSE
11	MASSE	28	WPRO*
12	N.C.	29	MASSE
13	MASSE	30	DKRD*
14	N.C.	31	MASSE
15	MASSE	32	SIDEB*
16	MTROD*	33	MASSE
17	MASSE	34	RDY*

ALIMENTATION UNITE DISQUETTES INTERNE ...4 BROCHES EN LIGNE DIRECTE (J13)

1	+12 (certains lecteurs n'ont besoin que de +5 volts)
2	GND
3	GND
4	+5

********** 4ème PARTIE - ASSIGNATION DES SIGNAUX DU PORT POUR LE 8520 **********

Bits de données d'adresse BFFR01 7-0 (A12*) (int2)

PA7..port de jeu 1, broche 6 (bouton de mise à feu*)
PA6..port de jeu 0, broche 6 (bouton de mise à feu*)
PA5..RDY* disquette prête*
PA4..TK0* piste 00 de la disquette*
PA3..WPRO* protection contre l'écriture*
PA2..CHNG* changement de disquette*
PA1..LED* lumière diode (0=allumée) / contrôle du filtre audio (sur l'A500 et l'A2000)
PA0..OVL Bit de recouvrement mémoire [ROM/RAM]

SP...KDAT données clavier
CNT..KCLK horloge clavier
PB7..P7 donnée 7
PB6..P6 donnée 6
PB5..P5 donnée 5 Données de l'interface parallèle
PB4..P4 donnée 4 Centronics
PB3..P3 donnée 3
PB2..P2 donnée 2
PB1..P1 donnée 1
PB0..P0 donnée 0

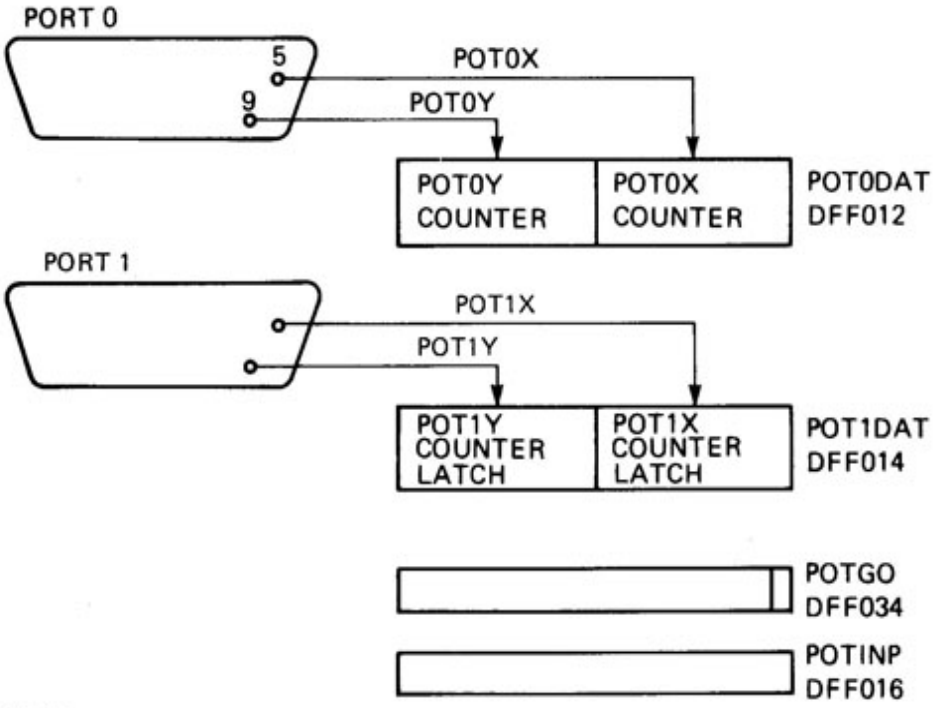
PC...drdy* Contrôle Centronics
F....ack*

Bits de données d'adresse BFDRFE 15-8 (A13*) (int6)

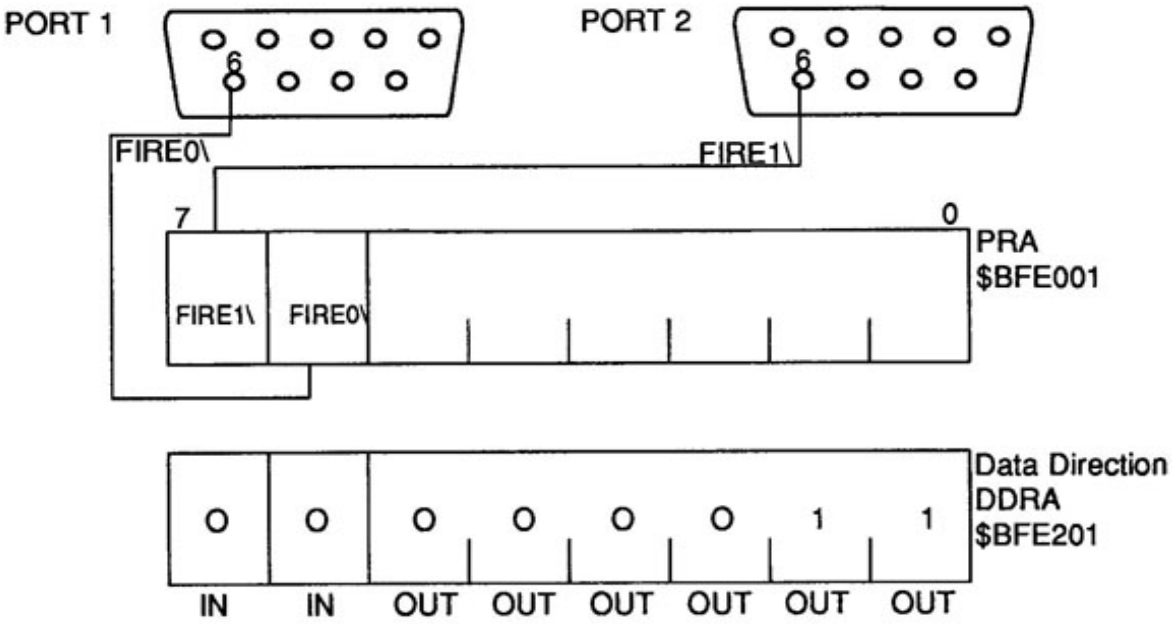
PA7..com ligne DTR*, sortie pilotée
PA6..com ligne RTS*, sortie pilotée
PA5..com ligne modem détecté*
PA4..com ligne CTS*
PA3..com ligne DSR*
PA2..SEL Contrôle Centronics
PA1..POUT +----- papier manquant-----------+
PA0..BUSY | +---imprim. Occupée----------+ |
| | | |
SPBUSY | +---bus série commodore ---+ |
CNT..POUT +---- bus série commodore------+

PB7..MTR* moteur
PB6..SEL3* sélection du 3ème lecteur externe
PB5..SEL2* sélection du 2ème lecteur externe
PB4..SEL1* sélection du 1er lecteur externe
PB3..SEL0* sélection du lecteur interne
PB2..SIDE* sélection face*
PB1..DIR direction
PB0..STEP* pas*

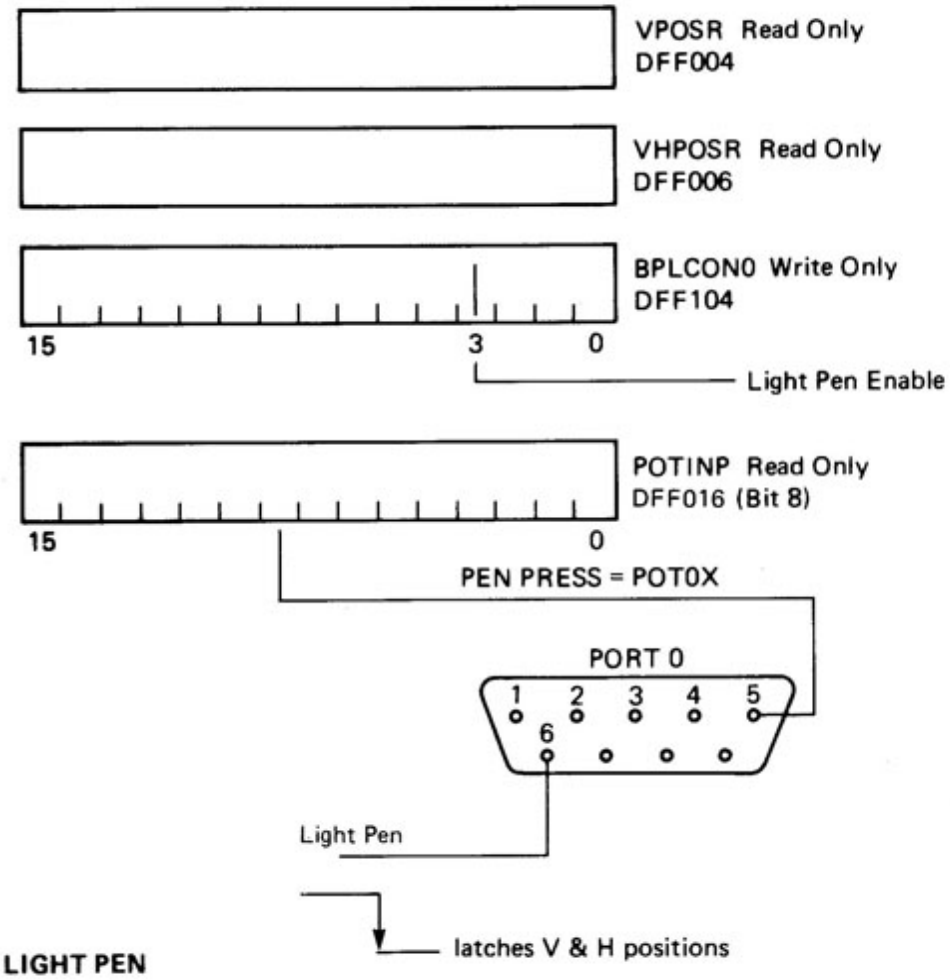
PC...inutilisé
F....INDEX* impulsion d'indice disquette*



POT COUNTERS



Reading FIRE BUTTONS


VPOSR Read Only
DFF004
VHPOSR Read Only
DFF006
BPLCON0 Write Only
DFF104
15
3
0
Light Pen Enable
POTINP Read Only
DFF016 (Bit 8)
15
0
PEN PRESS = POT0X
PORT 0
1
2
3
4
5
6
Light Pen
latches V & H positions

LIGHT PEN

APPENDICE F

ADAPTATEURS D'INTERFACE COMPLEXES

Cette Appendice contient des informations sur les adaptateurs d'interface 8520 destinés aux périphériques.

PUCES 8520 : ADAPTATEURS D'INTERFACE COMPLEXES (CIA)

Chaque système Amiga contient deux puces 8520, les Adaptateurs d'Interface Complex (CIA). Chaque puce possède 16 broches d'entrée/sortie génériques, plus un registre à décalage série, trois minuteries, une broche impulsion de sortie et une entrée qui détecte les pentes. Dans le système Amiga, diverses tâches sont assignées aux moyens intrinsèques de cette puce.

Mappe d'Adresses du CIAA

Adresse d'Octet	Nom du Registre	Bits de données 7	6	5	4	3	2	1	0
BFE001	pra	/FIR1	/FIR0	/RDY	/TK0	/WPRO	/CHNG	/LED	OVL
BFE101	prb	Port parallèle							
BFE201	ddra	Direction du port A (BFE001);1 = sortie (mis à 0x03)							
BFE301	ddrb	Direction du port B (BFE101);1 = sortie (peut être E/S)							
BFE401	talo	Minuterie A du CIAA octet de poids faible (0,715909 MHz, en NTSC; 0,709379 MHz, en PAL)							
BFE501	tahi	Minuterie A du CIAA octet de poids fort							
BFE601	tblo	Minuterie B du CIAA octet de poids faible (0,715909 MHz, en NTSC; 0,709379 MHz, en PAL)							
BFE701	tbhi	Minuterie B du CIAA octet de poids fort							
BFE801	todlo	Compteur d'événements 50/60 Hz, bits 7-0 (Synchro Vert. ou signaux d'horloge ligne)							
BFE901	todmid	Compteur d'événements 50/60 Hz bits 15-8							
BFEA01	todhi	Compteur d'événements 50/60 Hz bits 23-16							
BFEB01		Inutilisé							
BFEC01	sdr	Registre de données série du CIAA (connecté au clavier)							
BFED01	icr	CIAA interrupt control register							
BFEE01	cra	Registre de contrôle A du CIAA							
BFEF01	crb	Registre de contrôle B du CIAA							

Note : le CIAA peut générer des interruptions INT2.

Mappe d'Adresses du CIAB

Adresse d'Octet	Nom du Registre	Bits de données 7	6	5	4	3	2	1	0
BFD000	pra	/DTR	/RTS	/CD	/CTS	/DSR	SEL	POUT	BUSY
BFD100	prb	/MTR	/SEL3	/SEL2	/SEL1	/SEL0	/SIDE	DIR	/STEP
BFD200	ddra	Direction du Port A (BFD000);1 = sortie (mis à 0xFF)							
BFD300	ddrb	Direction du Port B (BFD100);1 = sortie (mis à 0xFF)							
BFE400	talo	Minuterie A du CIAB octet de poids faible (0,715909 MHz, en NTSC; 0,709379 MHz, en PAL)							
BFE500	tahi	Minuterie A du CIAB octet de poids fort							
BFE600	tblo	Minuterie B du CIAFb octet de poids faible (0,715909 MHz, en NTSC; 0,709379 MHz, en PAL)							
BFE700	tbhi	Minuterie B du CIAfb octet de poids fort							
BFD800	todlo	Compteur d'évén. synchro horiz. bits 7-0							
BFD900	todmid	Compteur d'évén. synchro horiz. bits 15-8							
BFDA00	todhi	Compteur d'évén. synchro horiz. bits 23-16							
BFDB00		Inutilisé							
BFDC00	dr	Registre de données série du CIAB (inutilisé)							
BFDD00	icr	Registre de contrôle des interruptions du CIAB							
BFDE00	cra	Registre de contrôle A du CIAB							
BFDF00	crb	Registre de contrôle B du CIAB							

Note : le CIAB peut générer des interruptions INT6.

MAPPE DES REGISTRES DE LA PUCE

Chaque 8520 possède 16 registres qu'on a la possibilité de lire ou d'écrire.
Voici la liste des registres et celle des adresses d'accès de chacun à
l'intérieur de l'espace mémoire imparti au 8520 :

RS3	R52	RS1	RS0	Registre #(hexa)	NOM	SIGNIFICATION
0	0	0	0	0	pra	Registre de données périphériques A
0	0	0	1	1	prb	Registre de données périphériques B
0	0	1	0	2	ddra	Registre de direction de données A
0	0	1	1	3	ddrb	Registre de direction B
0	1	0	0	4	talo	Minuterie A registre de poids faible
0	1	0	1	5	tahi	Minuterie A registre de poids fort
0	1	1	0	6	tblo	Minuterie B registre de poids faible
0	1	1	1	7	tbhi	Minuterie B registre de poids fort
1	0	0	0	8	todlow	Evénement LSB (bit moins significatif)
1	0	0	1	9	todmid	Evénement 8-15
1	0	1	D	A	todhi	Evénement MSB (bit plus significatif)
1	0	1	1	B		Non connecté
1	1	0	0	C	sdr	Registre de données série
1	1	0	1	D	icr	Registre de contrôle d'interruption
1	1	1	0	E	cra	Registre de contrôle A
1	1	1	1	F	crb	Registre de contrôle B

NOTE 'SOFTWARE' :

Le système d'exploitation s'est déjà réservé l'utilisation de plusieurs
minuteries du 8520.

CIAA, minuterie A - clavier (utilisée continuellement pour valider le transfert des touches sur lesquelles on appuie). N'EST PAS DISPONIBLE.

CIAA, minuterie B - Périphérique 'timer' virtuel (employé continuellement si on est sous contrôle d'Exec; employé pour la commutation des tâches, les interruptions et la synchronisation).

CIAA, TOD - minuterie 50/60 Hz employée par le 'timer.device'.
L'A1000 emploie les signaux d'horloge du secteur.
L'A500 emploie la synchro verticale. L'A2000 possède une sélection par cavalier.

CIAB, minuterie A - inutilisée

CIAB, minuterie B - inutilisée

CIAB, TOD - suiveur du faisceau vidéo de la 'graphics.library'.
Cette minuterie compte suivant le débit de la synchro horizontale et l'on s'en sert pour synchroniser des événements graphiques avec le faisceau vidéo.

A noter que les précédentes éditions de ce diagramme étaient incorrectes.

DESCRIPTION FONCTIONNELLE DES REGISTRES

PORTS d'E/S (PRA, PRB, DDRA, DDRB)

Les ports A et B sont composés chacun d'un registre de données périphérique 8 bits (PR) et d'un registre de direction de données (DDR). Si un bit du DDR est mis à 1, la position correspondante d'un bit du PR se transforme en sortie. Si un bit du DDR est mis à 0, le bit correspondant du PR est défini en tant qu'entrée.

En LISANT un registre PR, on lit l'état courant réel des broches d'E/S (PA0-PA7, PB0-PB7), sans se soucier d'avoir mis en place ces broches en tant qu'entrées ou en tant que sorties.

Les ports A et B possèdent des dispositifs élévateurs de charge aussi bien actifs que passifs, car ils permettent la compatibilité TTL et CMOS. Les deux ports ont, chacun, la possibilité de supporter deux charges TTL.

En plus de leurs opérations normales d'E/S, les ports PB6 et PB7 ont aussi des fonctions de sortie pour la minuterie.

TRANSFERT DE DONNEES

Un transfert de données se produit lors d'un transfert qui utilise la broche de sortie PC et la broche d'entrée FLAG. PC passera à l'état bas lors du troisième cycle après un accès au port B. On peut se servir de ce signal pour indiquer "données prêtes" sur le port B ou "données acceptées" en provenance du port B. Le transfert de données, lors d'un transfert 16 bits (qui utilise les deux ports A et B) est possible, si on lit et on écrit en premier dans le port A. FLAG est une entrée négative sensible aux crêtes qu'on peut employer pour réceptionner la sortie PC d'un autre 8520 ou en tant qu'entrée polyvalente. Toute transition négative sur FLAG mettra à 1 le bit d'interruption de FLAG.

REG	NOM	D7	D6	D5	D4	D3	D2	D1	D0
0	PRA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
1	PRB	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
2	DDRA	DPA7	DPA6	DPA5	DPA4	DPA3	DPA2	DPA1	DPA0
3	DDRB	DPB7	DPB6	DPB5	DPB4	DPB3	DPB2	DPB1	DPB0

METRONOMES (MINUTERIE A, MINUTERIE B)

Chaque métronome est composé d'un compteur de minuterie 16 bits destiné exclusivement à la lecture et d'un verrou de minuterie 16 bits destiné seulement à l'écriture. Les données écrites dans la minuterie sont verrouillées dans un verrou de la minuterie, tandis que les données lues à partir de la minuterie constituent le contenu, présent dans le compteur de la minuterie.

Le verrou est aussi dit "pre-scalaire" car il représente la valeur du compte à rebours à compter avant que la minuterie atteigne une condition de dépassement de capacité (plus de valeurs à compter). Cette valeur ("pre-scalaire") verrouillée est un diviseur de la fréquence d'entrée de l'horloge. Les minuteries peuvent être utilisées seules ou ensemble pour des opérations étendues. Différents modes de traitement des minuteries permettent la génération de délais de longue durée, d'impulsions de largeur variable, de trains d'impulsions et de formes d'onde de fréquence variable. En utilisant l'entrée CNT, les minuteries ont la possibilité de compter des impulsions externes ou de mesurer la fréquence, la largeur des impulsions et la durée du délai d'un signal externe.

Chaque minuterie possède un registre de contrôle qui lui est associé et qui permet de contrôler de manière indépendante les fonctions suivantes :

START/STOP (MARCHE/ARRET]

Un bit de contrôle permet au processeur de mettre en marche ou d'arrêter une minuterie à tout moment.

PB ON/OFF [PORT B actif/inactif]

Un bit de contrôle permet de faire apparaître la sortie de la minuterie sur la ligne de sortie du port B (PB6 pour la minuterie A et PB7 pour la minuterie B). Cette fonction est prioritaire par rapport au bit de contrôle DDRB et force la ligne PB appropriée à se transformer en sortie.

TOGGLE/PULSE [INTERRUPTEUR A BASCULE/IMPULSION]

Un bit de contrôle sélectionne la sortie appliquée au port B, si le bit PB se trouve à l'état actif. A chaque dépassement de capacité de la minuterie, la sortie a la possibilité de s'interrompre ou de générer une unique impulsion positive dont la durée sera égale à un cycle. La sortie "interrupteur à bascule" ['toggle'] passe à l'état haut chaque fois qu'on démarre la minuterie et passe à un état bas par l'intermédiaire de RES.

ONE-SHOT/CONTINUOUS [UN COUP/EN CONTINU]

Un bit de contrôle sélectionne un autre mode de la minuterie. En mode "un coup", la minuterie comptera à partir de la valeur verrouillée jusqu'à zéro, générera une interruption, rechargera la valeur verrouillée et s'arrêtera. En mode "continu", la minuterie comptera à partir de la valeur verrouillée jusqu'à zéro, générera une interruption, rechargera la valeur verrouillée et recommencera cette procédure sans arrêt.

En mode "un coup", le fait d'écrire dans le registre de poids fort de la minuterie (registre 5 pour la minuterie A, registre 7 pour la minuterie B) transfèrera le verrou présent dans la minuterie au compteur et fera démarrer le décompte sans se soucier de la présence du bit de démarrage.

FORCE LOAD [TRANSFERT PRIORITAIRE]

Un bit 'strobe' permet de charger le verrou de la minuterie dans le compteur de celle-ci à tout instant, que la minuterie soit ou ne soit pas lancée.

MODES D'ENTREE

Les bits de contrôle permettent la sélection de l'horloge utilisée pour décrémenter la minuterie. La minuterie A peut compter les impulsions de l'horloge 02, les impulsions CNT externes, les impulsions de dépassement de capacité de la minuterie elle-même ou même les impulsions de dépassement quand on garde la broche CNT à l'état haut.

Le verrou de la minuterie est chargé dans celle-ci à chaque dépassement de capacité de la minuterie elle-même ou, si la minuterie est arrêtée, après écriture dans l'octet de poids fort de la valeur pré-scalaire. Si la minuterie est en train de fonctionner, une écriture dans l'octet de poids fort chargera le verrou de la minuterie et non le compteur.

NOM DES BITS du Registre de LECTURE

REG	NOM	D7	D6	D5	D4	D3	D2	D1	D0
4	TALO	TAL7	TAL6	TAL5	TAL4	TAL3	TAL2	TAL1	TAL0
5	TAHI	TAH7	TAH6	TAH5	TAH4	TAH3	TAH2	TAH1	TAH0
6	TBLO	TBL7	TBL6	TBL5	TBL4	TBL3	TBL2	TBL1	TBL0
7	TBHI	TBH7	TBH6	TBH5	TBH4	TBH3	TBH2	TBH1	TBH0

NOM DES BITS du Registre d'ECRITURE

REG	NOM	D7	D6	D5	D4	D3	D2	D1	D0
4	TALO	PAL7	PAL6	PAL5	PAL4	PAL3	PAL2	PAL1	PAL0
5	TAHI	PAH7	PAH6	PAH5	PAH4	PAH3	PAH2	PAH1	PAH0
6	TBLO	PBL7	PBL6	PBL5	PBL4	PBL3	PBL2	PBL1	PBL0
7	TBHI	PBH7	PBH6	PBH5	PBH4	PBH3	PBH2	PBH1	PBH0

HORLOGE TEMPS REEL [TOD]

La broche TOD consiste en un compteur 24 bits binaire. Les transitions sur le bord positif de la broche provoquent une incrémentation du compteur. La broche possède une charge passive.

Une alarme programmable est fournie pour pouvoir générer une interruption à un moment précis. Les registres d'alarme se trouvent aux mêmes adresses que les registres TOD correspondants. L'accès à l'alarme est régi par un bit de contrôle du registre. L'alarme est destinée à l'écriture; une lecture de l'adresse TOD permettra de lire l'heure sans s'occuper de l'état du bit d'accès ALARM.

On doit suivre une séquence précise d'événements pour mettre en place TOD proprement et pour pouvoir le lire. TOD est arrêté automatiquement s'il se produit une écriture dans le registre. L'horloge ne démarrera toujours pas si l'on n'écrit pas dans le registre d'événement le moins significatif. Ceci garantit que TOD démarre toujours au moment choisi.

Comme, d'une phase à l'autre, une retenue pourrait se produire à tout moment, tout en respectant une opération de lecture, une fonction de mémorisation est présente pour garder constantes toutes les informations de TOD, lors d'une séquence de lecture. Lors d'une lecture, tous les registres se verrouillent et restent verrouillés jusqu'à la lecture de l'événement le moins significatif. L'horloge de TOD continue à compter pendant le verrouillage des registres de sortie. Si on ne doit lire qu'un seul registre, il n'y a pas de problème de retenue et le registre pourra être lu "au vol" étant donné que chaque lecture d'événement significatif est suivie de la lecture d'un événement moins significatif afin d'invalider le verrouillage.

NOM DES BITS d'écriture TEMPS REEL/ALARME ou de lecture TEMPS REEL

REG	NOM								
8	Evén. LSB*	E7	E6	E5	E4	E3	E2	E1	E0
9	Evén. 8-15	E15	E14	E13	E12	E11	E10	E9	E8
A	Evén. MSB**	E23	E22	E21	E20	E19	E18	E17	E16

ECRITURE
CRB7 = 0
CRB7 = 1 ALARME

* LSB = événement le moins significatif
** MSB = événement le plus significatif

REGISTRE A DECALAGE SERIE (SDR)

Le port série est un registre à décalage synchrone 8 bits tamponné. Un bit de contrôle sélectionne le mode entrée ou le mode sortie. Dans le système de l'Amiga, un registre à décalage est utilisé pour le clavier, tandis que l'autre n'est pas assigné. Remarquez que le port série compatible RS-232 est contrôlé par la puce Paula; pour plus de détails, voir le Chapitre 8.

MODE ENTREE [INPUT MODE]

En mode entrée, les données sur la broche SP sont décalées dans le registre à décalage lors de la crête ascendante du signal appliqué à la broche CNT. Après huit impulsions CNT, les données du registre à décalage sont vidées dans le registre de données série et une interruption est libérée.

MODE SORTIE [OUTPUT MODE]

En mode sortie, la minuterie A est utilisée comme générateur de débit de transmission. Les données sont décalées sur la broche SP moitié moins vite par rapport au seuil de dépassement de capacité de la minuterie A. Le débit de transmission maximal est égal à 02 divisé par 4, mais la vitesse de transmission maximale qu'on utilise sera déterminée par le chargement de la ligne et par la vitesse à laquelle le receveur répondra aux données d'entrée.

Pour commencer une transmission, on doit d'abord installer la minuterie A en mode continu et faire démarrer la minuterie proprement dite. Une transmission commence après écriture dans le registre de données série. Le signal d'horloge, dérivé de la minuterie A est, pour la broche CNT, un signal de sortie. Les données du registre de données série sont chargées dans le registre à décalage, puis décalées sur la broche SP quand se produit l'impulsion CNT. Les données décalées en sortie deviennent valides lors de la pente descendante successive du CNT et resteront valides jusqu'à la prochaine pente descendante.

Après huit impulsions CNT, une interruption est libérée pour indiquer qu'on peut envoyer d'autres données. Si le registre de données série était chargé à nouveau avec de nouvelles informations avant cette interruption, les nouvelles données seraient automatiquement chargées dans le registre à décalage et la transmission pourrait continuer.

S'il n'y avait pas de données supplémentaires à transmettre, après huit impulsions CNT, le CNT passerait à l'état haut et le SP resterait au niveau du dernier bit de données transmis.

Les données SDR sont expulsées avec en premier le bit le plus significatif. Les données d'entrée série devraient apparaître dans ce même format.

LE COTE BI-DIRECTIONNEL

Les possibilités bi-directionnelles du registre à décalage et de l'horloge CNT permettent de connecter plusieurs 8520 à un bus de communication série commun dans lequel un 8520 travaille comme maître, en émettant les données et le rythme du décalage, tandis que tous les autres 8520 travaillent comme esclaves. Les sorties CNT et SP sont à drain ouvert pour permettre le fonctionnement d'un tel bus commun. On pourra transmettre un protocole pour la sélection maître/esclaves par le bus série ou par des lignes destinées au transfert de données.

REG	NOM	D7	D6	D5	D4	D3	D2	D1	D0
C	SDR	S7	S6	S5	S4	S3	S2	S1	S0

REGISTRE DE CONTROLE D'INTERRUPTION (ICR)

Voici les cinq sources d'interruption sur un 8520 :

- Dépassement de capacité en provenance de la minuterie A (celle-ci compte en décrémentant après zéro)
- Dépassement de capacité en provenance de la minuterie B
- Alarme TOD
- Port série plein/vide
- Drapeau [Flag]

Un unique registre fournit l'information concernant les interruptions et les masques. Le registre de contrôle d'interruption est composé d'un registre MASK (pour l'écriture) et d'un registre DATA (pour la lecture). Chaque interruption mettra à 1 le bit correspondant du registre DATA. Chaque interruption qui aura été validée par un bit mis à 1 dans la même position, dans MASK, mettra à 1 le bit IR (bit le plus significatif) du registre DATA et mettra la broche IRQ à l'état bas. Dans un système multi-circuits, on peut interroger le bit IR pour détecter quel circuit a généré une demande d'interruption.

Lorsqu'on lit le registre DATA, son contenu est réinitialisé (mis à zéro) et la ligne IRQ revient à l'état haut. Du moment que ce registre est réinitialisé lors d'une lecture, on doit s'assurer qu'une interrogation d'interruption ou que le code de maintenance de l'interruption puissent préserver et répondre à tous les bits qu'on aurait pu mettre à 1 dans le registre DATA au moment ou celui-ci est lu. Avec une préservation et une réponse correctes, il est facilement possible de mélanger des méthodes de maintenance des interruptions directes et exprimées.

On a la possibilité de mettre à 1 ou à 0 un ou plusieurs bits du registre MASK sans affecter l'état courant d'aucun des autres bits du registre. On fait ceci en mettant en place l'état approprié du bit le plus significatif, qu'on appelle le bit 'set/clear' [mise à 1/mise à 0]. Dans les bits 6-0, on constituera soi-même un masque indiquant les bits qu'on souhaite affecter. Ensuite, en se servant du bit 7, on spécifie COMMENT devront être affectés les bits présents dans les positions correspondantes du masque.

o Si le bit 7 est un 1, alors chacun des bits 6-0 de l'octet de masque qui est à 1 met à 1 le bit correspondant du registre MASK. Tout bit qu'on aurait mis à 0 fera en sorte que le bit du registre MASK continue à rester dans son état initial.

o Si le bit 7 est un 0, alors chacun des bits 6-0 de l'octet de masque qui est à 1 met à 0 le bit correspondant du registre MASK. En outre, tout bit 0 de l'octet de masque ne produira aucune modification du contenu du bit correspondant du registre MASK.

Si une interruption doit se produire dans une situation particulière, alors le bit de MASK correspondant doit se trouver à 1.

Exemple : Supposons qu'on veuille mettre à 1 le bit d'interruption de la minuterie A (valide l'interruption de cette minuterie) et qu'on veuille s'assurer que toutes les autres interruptions ont été réinitialisées. Voici la séquence qu'on peut utiliser :

```
INCLUDE  "hardware/cia.i"
XREF     ciaa                    ; Dans amiga.lib
lea      ciaa,a0                 ; Défini dans amiga.lib
move.b   401111110,ciaicr(a0)
```

le bit le plus significatif à 0 signifie qu'il faut mettre à 0 tout bit dont la valeur est égale à 1 dans la suite de l'octet.

```
INCLUDE  "hardware/cia.i"
XREF     ciaa                    ; Dans amiga.lib
lea      ciaa,a0                 ; Défini dans amiga.lib
move.b   %100000001,ciaicr(a0)
```

le bit le plus significatif est un 1, ce qui veut dire qu'il faut mettre à 1 tout bit dont la valeur est égale à 1 dans la suite de l'octet (ne pas modifier des valeurs au cas où le bit de valeur déjà présent serait égal à zéro).

LECTURE DU REGISTRE DE CONTROLE D'INTERRUPTION

REG	NOM	D7	D6	D5	D4	D3	D2	D1	D0
D	ICR	IR	0	0	FLG	SP	ALRM	TB	TA

ECRITURE DU MASQUE DE CONTROLE D'INTERRUPTION

REG	NOM	D7	D6	D5	D4	D3	D2	D1	D0
D	ICR	S/C	x	x	FLG	SP	ALRM	TB	TA

REGISTRES DE CONTROLE

Il existe deux registres de contrôle dans le 8520, CRA et CRB. CRA est associé à la minuterie A et CRB est associé à la minuterie B. Voici le format de ces registres :

REGISTRE DE CONTROLE A

BIT	NOM	FONCTION
0	START	1 = démarre la minuterie A, 0 - arrête la minuterie A. Ce bit est réinitialisé automatiquement (= 0) si un dépassement de capacité venait à se produire en mode "un coup".
1	PBON	1 = sortie de la minuterie A sur PB6, 0 = PB6 est l'opération normale.
2	OUTMODE	1 = interrupteur à bascule, 0 = impulsion.
3	RUNMODE	1 = mode "un coup", 0 = mode "en continu".
4	LOAD	1 = force un chargement (il s'agit d'une entrée 'strobe', il n'y aura donc aucun stockage de données; le bit 4 relira toujours un zéro et le fait d'y écrire un zéro ne produira aucun effet).
5	INMODE	1 = La minuterie A compte la transition positive du CNT. 0 = La minuterie A compte les impulsions de l'horloge 02.
6	SPMODE	1 = Port série=sortie (CNT est l'origine de la fréquence de décalage). 0 = Port série=entrée (une fréquence de décalage externe est nécessaire).
7	INUTILISE	

MAPPE DE BITS DU REGISTRE CRA

REG#	NOM	INUTIL.	SPMODE	INMODE	LOAD	RUNMODE	OUTMODE	PBON	START
E	CRA	unused unused	0=input 1=output	0=02 1=CNT	1=force load (strobe)	0=cont. 1=one-shot	0=pulse 1=toggle	0=PB60FF 1-PB60N	0=stop 1=start

|<----------Variables de la minuterie A----------->|

Tous les bits inutilisés du registre ne sont pas affectés par une opération d'écriture et sont forcés à 0 lors d'une lecture.

REGISTRE DE CONTROLE B:

BIT	NOM	FONCTION
0	START	1 = démarre la minuterie B, 0 - arrête la minuterie B. Ce bit est réinitialisé automatiquement (= 0) si un dépassement de capacité venait à se produire en mode "un coup".
1	PBON	1 = sortie de la minuterie B sur PB7, 0 = PB7 est l'opération normale.
2	OUTMODE	1 = interrupteur à bascule, 0 = impulsion.
3	RUNMODE	1 = mode "un coup", 0 = mode "en continu".
4	LOAD	1 = force un chargement (il s'agit d'une entrée 'strobe', il n'y aura aucun stockage de données; le bit 4 relira toujours un zéro et le fait d'y écrire un zéro ne produira aucun effet).
6,5	INMODE	Les bits CRB6 et CRB5 sélectionnent l'un des quatre modes d'entrée possibles avec la minuterie B, de la façon suivante :

CRB6	CRB5	Mode Sélectionné
0	0	La minuterie B compte les impulsions de l'horloge 02.
0	1	La minuterie B compte les transitions positives du CNT.
1	0	La minuterie B compte les impulsions de dépassement de capacité de la minuterie A.
1	1	La minuterie B compte les impulsions de dépassement de capacité de la minuterie A pendant que la broche CNT est à l'état haut.

7	ALARM	1 = l'écriture dans les registres TOD met en place l'Alarme. 0 = l'écriture dans les registres TOD met en place l'horloge TOD. Une lecture des registres TOD lira toujours l'horloge TOD, sans s'occuper de l'état du bit d'Alarme.

MAPPE DE BITS DU REGISTRE CRB

REG#	NOM	ALARM	INMODE	LOAD	RUNMODE	OUTMODE	PBON	START
F	CRB	0=TOD 1=Alarm	00-02 01=CNT 10=Timer A 11=CNT+ Timer A	1=force load (strobe)	0=cont. 1=one- shot	0=pulse 1=toggle	0=PB70FF 1=PB70N	0=stop 1=start

|<----------Variables de la minuterie B---------->|

Tous les bits inutilisés du registre ne sont pas affectés par une opération d'écriture et sont forcés à 0 lors d'une lecture.

ASSIGNATION DES SIGNAUX AUX PORTS

Cette partie donne des précisions sur la manière dont les différents signaux se rapportent aux ports du 8520. Ces informations donnent la possibilité au programmeur de mettre en rapport les adresses du port avec les éléments (ou les signaux de contrôle internes) du monde extérieur qui devront être affectés. Cette partie s'adresse en premier lieu aux programmeurs système et, d'une manière générale, ne devrait pas être utile aux programmeurs d'applications. En règle générale, le 'software' du système est configuré pour gérer la mise en place de signaux particuliers, sans tenir compte des changements pouvant intervenir physiquement dans les connections.

NOTE

Dans un système d'exploitation multitâche, plusieurs tâches différentes peuvent rivaliser lorsqu'elles font appel aux ressources du système. Les programmeurs d'applications devraient suivre les règles établies pour l'accès aux ressources afin de garantir la compatibilité de leurs programmes avec le système.

Adresse BFEr01 bits de données 7-0 (A12*) (INT2)

PA7..port de jeu 1, broche 6 (bouton de mise à feu*)
PA6..port de jeu 0, broche 6 (bouton de mise à feu*)
PA5..RDY* disquette prête*
PA4..TKO* piste 00 de la disquette*
PA3..WPRO* protection contre l'écriture*
PA2..CHNG* changement disquette*
PA1..LED* lumière de la diode luminescente (0=allumée)
PAO..OVL bit de recouvrement mémoire
SP...RDAT données du clavier
CNT..RCLR
PB7..P7 bit de donnée 7
PB6..P6 bit de donnée 6
PB5..P5 bit de donnée 5 Données
PB4..P4 bit de donnée 4 pour l'interface parallèle
PB3..P3 bit de donnée 3 Centronics
PB2..P2 bit de donnée 2
PB1..P1 bit de donnée 1
PBO..P0 bit de donnée 0
PC...drdy* contrôle centronics
F....ack*

Adresse BFDr00 bits de données 15-8 (A13*) (INT6)

PA7..com ligne DTR*, sortie pilotée
PA6..com ligne RTS*, sortie pilotée
PA5..com ligne modem détecté*
PA4..com ligne CTS*
PA3..com ligne DSR*
PA2..SEL Contrôle Centronics
PA1..POUT papier manquant---------+
PA0..BUSY imprim. Occupée----+ |
                                | |
SPBUSY commodore -----------+ |
CNT..POUT commodore ---------------+

PB7..MTR* moteur
PB6..SEL3* sélection du 3ème lecteur externe
PB5..SEL2* sélection du 2ème lecteur externe
PB4..SEL1* sélection du 1er lecteur externe
PB3..SEL0* sélection du lecteur interne
PB2..SIDE* sélection face*
PB1..DIR direction
PB0..STEP* pas* (minimum 3,0 milli-secondes)

PC...inutilisé
F....INDEX* impulsion d'indice disquette*

```
; Un exemple complet de synchronisation avec le 8520. Cet exemple fait
; clignoter la diode de l'alimentation à (exactement) des intervalles de
; 3 milli-secondes. Le programme remplace la machine, soyez attentifs à 50 !
;
; Les fréquences de base du quartz de l'Amiga sont les suivantes :
;         NTSC    28,63636 MHz
;         PAL     28,37516 MHz
;
; Chacune des deux minuteries 16 bits des puces 8520 compte à rebours à
; 1/10 de la fréquence d'horloge du CPU, c'est-à-dire à 0,715909 MHz.
; Ceci se résoud à 1,3968255 micro-secondes pour chaque comptage. En PAL le
; compte à rebours est un peu plus lent, 0,709379 MHz.
;
; Pour attendre 1/100 de seconde, il faudra attendre 10000 micro-secondes.
; On devra mettre le registre de la minuterie à (10000 / 1,3968255 - 7159).
;
; Pour attendre 3 milli-secondes il faudra attendre 3000 micro-secondes.
; On devra mettre le registre à (3000 / 1,3968255 - 2148).
;

      INCLUDE "hardware/cia.i"
      INCLUDE "hardware/custom.i"
;
      XREF _ciaa
      XREF _ciab
      XREF _custom
;
      lea  _custom,a3                ; Base des puces spécialisées
      lea  _ciaa,a4                  ; Prendre l'adresse de base si CIA-A
;
      move.w S7fff,dmacon(a3)        ; Eliminer toutes les interruptions des puces
;
; Installation, à faire une seule fois
; Ceci met tous les bits voulus de la minuterie A en mode "un coup".
;
      move.b ciacra(a4),d0           ; Mettre en place le registre de contrôle A du CIAA
      and.b  #%11000000,d0           ; Ne pas se débarrasser des bits
      or.b   #%00001000,d0           ; dont on ne va pas se servir...
      move.b d0,ciacra(a4)
      move.b #%01111111,ciaicr(a4)          ; Mettre à 0 toutes les interruptions du 8520
;
;     Mettre en place l'horloge temps réel (octet de poids faible
;     PUIS octet de poids fort).
;     Et l'octet de poids faible avec $ff
;     Décaler l'octet de poids fort de 8
;
TIME    equ 2148
      move.b (TIME&$FF),ciatalo(a4)
      move.b (TIME>>8),ciatahi(a4)
;
;     Attendre que la minuterie compte à rebours
busy_wait:
    btst.b #0,ciaicr(a4)             ; Attendre le drapeau de fin de minuterie
    beq.s  busy wait
    bchg.b #CIAB LED,ciapra(a4)   ; La lumière clignote
    bset.b #0,ciacra(a4)             ; Relancer la minuterie
    bra.s  busy_wait

    END
```

DETAIL SUR LES CONNEXIONS 'HARDWARE'

Le 'hardware' du système sélectionne les deux CIA quand les trois bits d'adresse les plus significatifs sont à 101. En outre, le CIAA est sélectionné quand A12 se trouve à l'état bas et A13 à l'état haut; le CIAB est sélectionné quand A12 est à l'état haut et A13 à l'état bas. Le CIAA communique avec les bits de données 7-0, le CIAB avec les bits de données 15-8.

Les bits d'adresse A11, A10, A9 et A8 sont employés pour indiquer le registre auquel on veut accéder parmi les 16 registres internes. C'est ce qu'indique le "r" présent dans l'adresse. Il ne faut pas s'occuper de tous autres bits. Ainsi, on sélectionne le CIAA par l'adresse binaire suivante : 101x xxxx xx01 rrrr xxxx xxx0.
L'adresse du CIAB : 101x xxxx xx10 rrrr xxxx xxx1.

En gardant à l'esprit des extensions futures, nous mous sommes décidés pour les adresses suivantes : CIAA = BFEr01; CIAB = BFDr00.
Le 'software' devra accéder à ces adresses et non à d'autres.

SIGNAUX D'INTERFACE

ENTREE HORLOGE [CLOCK INPUT]

L'horloge 02 est une entrée TTL compatible utilisée pour des traitements internes sur les périphériques ainsi que comme référence de synchronisation pour communiquer avec le bus de données du système. Sur l'Amiga, cette horloge est reliée à l'horloge "E" du 68000. Cette horloge tourne à 1/10 de la vitesse de l'horloge du CPU. Celle-ci fonctionne à 0,715909 MHz, en NTSC, ou à 0,709379 Mhz en PAL.

CS - [CHIP-SELECT INPUT] ENTREE DE SELECTION DE LA PUCE

L'entrée CS contrôle l'activité du 8520. Un niveau bas sur CS, tandis que 02 se trouve à l'état haut, aura comme résultat la réponse du périphérique aux signaux présents sur les lignes (RS) d'adresse et sur les lignes R/W. Un niveau haut sur CS évite que ces lignes contrôlent le 8520.
D'ordinaire, la ligne CS est activée (zéro active) à 02 par une combinaison d'adresses appropriée.

R/W - ENTREE LECTURE/ECRITURE [READ/WRITE INPUT]

Le signal RW est normalement fourni par le micro-processeur et contrôle la direction des transferts de données du 8520. Un niveau haut sur R/W indique une lecture (données transférées à l'extérieur du 8520), tandis qu'un niveau bas indique une écriture (données transférées à l'intérieur du 8520).

RS3-RS0 - ENTREES D'ADRESSES

Les entrées d'adresses sélectionnent les registres internes comme décrit dans la mappe des registres.

DB7-DB0 - ENTREES/SORTIES DU BUS DE DONNEES

Les huit broches de sortie du bus de données transfèrent les informations entre le 8520 et le bus de données du système. Ces broches sont des entrées haute impédance tant que CS est à l'état bas et que R/W et 02 sont à l'état haut, pour lire le périphérique. Pendant la lecture, les tampons de sortie du bus de données sont validés, véhiculant ainsi les données du registre sélectionné au bus de données du système.

IRQ - SORTIE DE LA DEMANDE D'INTERRUPTION [INTERRUPT REQUEST OUTPUT]

IRQ est une sortie à drain ouvert qui est normalement connectée à l'entrée des interruptions du processeur. Une résistance élévatrice externe maintient le signal haut, permettant ainsi de connecter ensemble plusieurs sorties IRQ. D'ordinaire, la sortie IRQ est hors service (haute impédance) et passe à l'état bas, comme nous l'avons indiqué dans la description fonctionnelle.

RES - ENTREE REINITIALISATION [RESET INPUT]

Un état bas sur la broche RES réinitialise tous les registres internes. Les broches des ports sont mises en place en tant qu'entrées et les registres des ports sont mis à zéro (bien qu'une lecture des ports retourne tous les registres à l'état haut à cause des charges passives). Les registres de contrôle de la minuterie sont mis à zéro et les verrous des minuteries sont tous mis à un. Tous les autres registres sont remis à zéro.

APPENDICE G

AUTOCONFIG (MARQUE DEPOSEE)

Le protocole AUTOCONFIG est conçu pour permettre l'assignation automatique des emplacements d'adresses disponibles aux cartes d'extension, en évitant à l'utilisateur de devoir configurer la machine en utilisant des cavaliers. Lors d'une réinitialisation, chaque carte appelle à tour de rôle l'adresse $E80000, avec une information d'identification lisible, information qui est, en règle générale, dans un format complémenté à 1, et qui est stockée dans les demi-octets de poids fort des $40 premier mots ($80 octets) de la carte d'extension. Cette information d'identification comprend la taille de la carte, ses préférences pour ce qui est de l'espace adressable, le type de la carte (mémoire ou autre) et un Numéro 'Hardaware' du Constructeur fourni par le Support Technique d'Amiga Commodore, West Chester, Pennsylvania.

Chaque carte contient le 'hardware' de configuration qui comprend un verrou d'adresse qu'on trouve dans le demi-octet à l' 'offset' $40 et un demi-octet à l' 'offset' $004a. Lorsque les adresses de base assignées à la carte de A23 à A16 sont écrites dans ce registre, la carte se verrouille et se manifeste à l'adresse qu'on lui a assigné, puis transmet un signal qui s'appelle CONFIG-OUT qui aura pour résultat de faire apparaître la carte suivante à $E80000. Pour rendre certains types de cartes moins chers, un registre d'écriture d'une carte d'extension peut être organisé sous la forme d'un registre d'un octet ou sous la forme de deux demi-octets.

Si le registre a une taille d'un demi-octet, il doit alors verrouiller le demi-octet de poids faible de l'adresse qu'on lui a assigné tant que le demi-octet de poids fort (à $48) ne sera pas écrit. Pour fonctionner avec n'importe quel type de carte, ceci autorise l'algorithme suivant :

Ecrire le demi-octet d'adresse de poids faible à l' 'offset' $4A
Ecrire l'octet entier d'adresse à l' 'offset' $48

Alternativement, on pourra demander à beaucoup de cartes de se mettre "en veilleuse" (passer outre CONFIG-OUT et arrêter de répondre) en écrivant à l' 'offset' S004c de la carte. Un bit du demi-octet à l' 'offset' $0008 met un drapeau si la carte accepte la mise en veilleuse.

Toutes les cartes commerciales destinées aux emplacements d'extension de l'Amiga se doivent d'implémenter le protocole AUTOCONFIG. Une information plus ample, spécifique et approfondie de la machine, sur la conception et l'implémentation des cartes AUTOCONFIG, est disponible auprès du Support Technique de Commodore Amiga.

Le système d'exploitation de l'Amiga contient un support pour faire correspondre les lecteurs de disquettes avec des cartes AUTO-CONFIG. Depuis la version 1.3, le système d'exploitation accepte aussi l'initialisation du 'software' des cartes de lecteurs de disquettes en ROM. En règle générale, les applications ne devraient pas s'attaquer aux périphériques d'extension AUTOCONFIG, mais ils devraient plutôt permettre au système 'software' de l'Amiga de gérer toute la configuration automatiquement. Plusieurs cartes contiennent des registres qui, une fois activés, pourraient provoquer des dommages irréparables, par exemple, les données du disque dur de l'utilisateur pourraient être effacées, si la carte d'extension a été configurée de manière incorrecte.

Toutefois, certains types d'applications autonomes de bas niveau, comme les cartes d'extension de mémoire vive, peuvent avoir besoin de configurer le 'hardware', sans avoir recours au système d'exploitation de l'Amiga. De telles applications devraient se limiter à configurer les cartes d'extension RAM (des cartes qui demandent à être ajoutées dans la liste mémoire libre) et des cartes connues dédiées à des applications spécifiques. Toutes les autres cartes devraient se trouver en veilleuse, si ces cartes l'acceptent, ou être configurées pour être ignorées, au cas où elles ne seraient pas acceptées. (Il y a beaucoup de cartes qui n'acceptent pas une mise en veilleuse.) La configuration des cartes devrait être tentée par les applications qui, après réinitialisation, prennent en charge la machine dans sa totalité. La présence d'une carte AUTOCONFIG, qui attend pour être configurée, se détermine en comparant les demi-octets qui apparaissent à l'adresse initiale d'AUTOCONFIG avec les valeurs valides de ces demi-octets présents dans les spécifications.

Les caractéristiques d'AUTOCONFIG exigent que les cartes soient configurées dans des limites qui concordent avec leurs exigences d'espace. Par exemple, une carte d'extension mémoire d'1 Mo devrait être configurée dans les limites d'un 1 Mo. Il existe deux exceptions à la règle : les cartes avec un espace adressable de 4 Mo acceptent d'être placées à S200000 et S600000 ou alors d'être placées dans la limite de 4 Mo. Ces exceptions sont nécessaires car l'espace de 8 Mo, réservé à des extensions de l'actuelle machine, commence à $200000.

DEBOGAGE DES CARTES AUTOCONFIG

S'il y a un vice dans l'information de configuration, la carte peut être ignorée, peut se mettre en veilleuse ou peut provoquer l'effondrement du système de telle manière que tout diagnostic serait difficile à émettre. Il existe une astuce qui permet de tester l'information de configuration. Coupez la ligne CONFIGIN* sur la carte et câblez un interrupteur sur la ligne. Câblez l'interrupteur de telle façon que -s'il est mis en fonction dans un sens, la ligne CONFIGIN* passera du bus à la carte.

Ceci permettra à la carte d'extension de répondre au processus AUTOCONFIG.
Si l'interrupteur est mis en fonction dans l'autre sens, il devra être câblé
de telle façon que la carte soit forcée à l'état haut.
Ce qui aura pour effet d'inactiver AUTOCONFIG sur la carte.

Mettez l'interrupteur en position pour que la ligne CONFIGIN* soit forcée à
l'état haut, mettez ensuite en route le système. La carte sera invisible au
'software' du système. Activez un débogueur et tournez l'interrupteur. La
carte devrait maintenant répondre à l'adresse normale $E80000. Votre vue de
la carte est identique à celle du système lorsqu'il configure la carte.
Vous pourrez comparer les bits avec les valeurs attendues.

NOTE

La carte à déboguer devra être la dernière carte du système (la plus proche
des emplacements PC, loin de l'alimentation générale). Les cartes en aval de
la carte à déboguer ne seront pas configurées par le système.

TABLEAU DE SPECIFICATION D'ADRESSES

Le tableau qui suit décrit l'information d'identification de la carte et les
registres AUTOCONFIG qui apparaissent dans les premiers $80 octets d'une
carte AUTOCONFIG, lors de la configuration.

NOTES

o L'information d'identification est stockée dans les demi-octets de poids
fort des adresses (de mots) paires au début d'une carte AUTOCONFIG. Par
exemple, les deux premiers mots d'une carte pourraient contenir $Cxxx 1xxx.
L'information valide de ces deux premiers mots serait le $C (demi-octet de
poids fort à l' 'offset' $00), puis le $1 (demi-octet de poids fort du mot à
l' 'offset' $02). On interprète beaucoup d'informations en réunissant
plusieurs demi-octets, avec des demi-octets qui vont d'une adresse basse à
une adresse haute et qui contiennent des parties de la valeur résultante
dans un ordre qui va de l'octet de poids fort à l'octet de poids faible.

o Tous les demi-octets d'information, à l'exception de ceux qui se trouvent
aux 'offsets' $00/02 et $40/42, sont stockés sous forme inversée
(complémentés à 1) et subissent une opération logique OR avec $F, avant
qu'on puisse les interpréter, selon le tableau ci-dessous. On présume que
les demi-octets inutilisés (les trois autres demi-octets de chaque mot)
peuvent ne pas contenir une valeur précise. Toutes les valeurs écrites dans
la zone AUTOCONFIG, y compris les adresses assignées, sont écrites selon un
format inversé.

o Toutes les adresses indiquées ici sont des 'offsets' à partir de
l'adresse de base $E80000, où les cartes apparaissent lors de la
configuration, ainsi l' 'offset' $02 se trouve à $E80002, l' 'offset' $04
à l'adresse $E80004, etc..

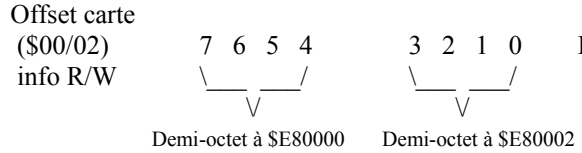


Planche G-1 : Comment Lire le Tableau de Spécification d'Adresses

NOTE

La numérotation des bits (7 6 5 4 3 2 1 0) est à utiliser quand deux
demi-octets doivent être interprétés ensemble comme un octet. Physiquement,
chaque demi-octet est le demi-octet de poids fort du mot à son adresse (par exemple, bits 15 14 13 12).

Tableau G-1 : Tableau de Spécification d'Adresses

'OFFSET' :	Adresse 1	Adresse 2	Description
($00/02) Lire Non Inversé	7 6 5 4	3 2 1 0	Board size: 000=8 Mo 100=512 Ko; 001=64 Ko 101= 1 Mo; 010=128 Ko 110= 2 Mo; 011=256 Ko 111= 4 Mo 1 = La prochaine carte est aussi sur cette même carte 1 = Vecteur optionnel en ROM valide 1 = A ajouter dans la liste de mémoire libre (RAM) Type carte: 00 = Réservé; 01 = Réservé; 10 = Réservé; 11 = Type courant
($04/06) Lire Inversé	7 6 5 4	3 2 1 0	Les fabricants choisissent le numéro du produit D.-oct. fort / D.-oct. faible
($08/0A) Lire Inversé	7 6 5 4	3 2 1 0	(Rappel - à lire inversés) Réservé - Actuellement devraient être à 0 0 = cette carte peut être mise en veilleuse; 1 = cette carte ignore la mise en veilleuse 0 = tout l'espace est OK; 1 = zone de 8 Mo préférée

'OFFSET' :	Adresse 1	Adresse 2	Description
(SOC/OE) Lire Inversé	7 6 5 4 |__|__|__|_____	3 2 1 0 |__|__|__|_____	Réservé - doit être un zéro
($10/12) Lire Inversé	7 6 5 4 ___ ___/ V D.-oct. fort	3 2 1 0 ___ ___/ V D.-oct. faible	Octet de poids fort de l'unique numéro de fabricant assigné au constructeur. (il ne s'agit pas du numéro de développeur !)
(S14/16) Lire Inversé	7 6 5 4 ___ ___/ V D.-oct. fort	3 2 1 0 ___ ___/ V D.-oct. faible	Octet de poids faible de l'unique numéro de fabricant assigné au constructeur. (il ne s'agit pas du numéro de développeur !)

NOTE

Le numéro de fabricant est assigné par le Support Technique Amiga Commodore, West Chester, Pennsylvania (CATS). Contacter CATS pour plus d'informations.

($18/1A)	7 6 5 4	3 2 1 0	N.o série en option, 1er octet (msb)
($1C/1E)	7 6 5 4	3 2 1 0	N.o série en option, 2e octet
($20/22)	7 6 5 4	3 2 1 0	N.o série en option, 3e octet
($24/26) Lire Inversé	7 6 5 4	3 2 1 0	N.o série en option, 4e octet (lsb)
(S28/2A) Lire Inversé	7 6 5 4 ___ ___/ V D.-oct. fort	3 2 1 0 ___ ___/ V D.-oct. faible	Octet de poids fort du vecteur optionnel en ROM.
(S2C/2E) Lire Inversé	7 6 5 4 ___ ___/ V D.-oct. fort	3 2 1 0 ___ ___/ V D.-oct. Faible	Octet de poids faible du vecteur optionnel en ROM. Si le bit de "vecteur valide en ROM" est mis à 1 dans le demi-octet au début de la carte, ce vecteur optionnel en ROM est l' 'offset' à partir de la base de la carte jusqu'au structures du pilote de la ROM.
(S30/32) R/W Inversé	7 6 5 4	3 2 1 0	Lire - Réservé, doit être 00 Ecrire – Réinitialisation optionnelle du registre de base de la carte à l'adresse de configuration.
($34/36)	7 6 5 4	3 2 1 0	Réservé, doit être 00
(S38/3A)	7 6 5 4	3 2 1 0	Réservé, doit être 00
(S3C/3E) Inversé	7 6 5 4	3 2 1 0	Réservé, doit être 00

'OFFSET' :	Adresse 1	Adresse 2	Description	
(S40/42)	7 6 5 4	3 2 1 0	Ecrire	Lire
R/W	| | | |	| | | |		
Non Inversé	| | | |	| | | ___	Valid. interpt.	Valid. interpt.
	| | | |	| | _____	Définiss. par l'util.	Indéfini
	| | | |	| _____	Réinitial. locale	Doit être à 0
	| | | |	_____	Définiss. par l'util.	Indéfini
	| | | _____		Définiss. par l'util.	INT2 en attente
	| | _____		Définiss. par l'util.	INT6 en attente
	| _____		Définiss. par l'util.	INT7 en attente
	_____		Définiss. par l'util.	INT d'extraction de carte

NOTE

L'emploi des registres $40/42 est une caractéristique en option qu'on peut implémenter sur des cartes qui génèrent des interruptions. Ces cartes rendent cela possible aux serveurs d'interruption d'une carte spécifique afin de pouvoir déterminer si l'interruption courante a été générée par leur propre carte ou par tout autre 'hardware' qui emploie la même ligne d'interruption.

(S44/46) Réservé, en lecture doit être 00. Indéfini en écriture
R/W
Inversé

(S48/4A)	7 6 5 4	3 2 1 0	Reg. d'adr. de base (seul. écr.)
Ecr. seul.	___ ___/	___ ___/	Ces bits sont comparés avec les bits de A23 à A16 (ou moins)
Non Inversé	\/	\/	pour déterminer l'adresse de base de la carte.
	D.-oct. fort	D.-oct. faible	

($4C/4E)	7 6 5 4	3 2 1 0	Registre optionnel de mise en veilleuse. Toute écriture dans
Ecr. seul.	___ ___/	___ ___/	$4C aura pour effet de faire passer la carte outre CONFIG-OUT
	\/	\/	et de ne plus la voir répondre à aucune adresse jusqu'à la
	_____\		REINITIALISATION.
	/		Un bit du demi-octet $08 met un drapeau si on peut mettre
			la carte en veilleuse.

(de S50 à S7E) Réservé, doit être 00
Inversé

Il faut se rappeler que tout demi-octet, sauf $00/02 et $40/42 apparaîtront réellement inversés par rapport aux demi-octets du tableau ci-dessus.
Par exemple, un demi-octet "doit être un zéro" apparaîtra en tant que $F et les valeurs hexadécimales seront aussi inversées (par ex., une valeur égale à $1 se lira $E, etc.).

```
/*

* Examine toutes les cartes AUTOCONFIG (marque déposée) du système.

*/

include "exec/types.h"
include "libraries/configvars.h"

struct Library *OpenLibrary();
struct ConfigDev *FindConfigDev();
struct Library *ExpansionBase;

void main()
{
struct ConfigDev *myCD=0;

ExpansionBase=OpenLibrary("expansion.library",0L);

while(myCD=FindConfigDev(myCD,-1L,-1L)) /* Cherche n'importe quelle
                                structure ConfigDev */
   {
   printf("\n---ConfigDev structure found at location $%1x---\n",myCD);

   /* Ces valeurs sont lues directement à partir de la carte */
   printf("er Manufacturer        =");
      printf("%d,",myCD->cd Rom.er Manufacturer);
      printf("S%x,",myCD->cd Rom.er Manufacturer);
      printf("(-$%4x)\n",-myCD->cd Rom.er Manufacturer);

   printf("er Product             =");
      printf("%d,",myCD->cd Rom.er Product);
      printf("$%x,",myCDÑ>cd Rom.er Product);
      printf("(-$%x)\n",-myCD->cd Rom.er Product);

   printf("er Type                =$%x\n",myCD->cd Rom.er Type);

   printf("er Flags               =");
      printf("$%x\n",myCD->cd Rom.er Flags);

   /* Ces valeurs sont générées lorsque le 'software' d'AUTOCONFIG
    * déplace la carte
    */
   printf("cd BoardAddr           =$%1x\n",myCD->cd BoardAddr);
   printf("cd BoardSize           =$%1x (%ldK)\n",
   myCD->cd BoardSize,((ULONG)myCD->cd BoardSize)/1024);

   printf("cd Flags               =$%x\n",myCD->cd Flags);
   }
CloseLibrary(ExpanionBase);
}
```

APPENDICE H

LE CLAVIER

Cette appendice contient des spécifications sur l'interface clavier de l'A1000, de l'A500 et de l'A2000.

Le clavier est connecté à l'ordinateur Amiga par le biais d'un câble avec quatre connections principales. Ces quatre fils fournissent l'alimentation 5 Volts, la masse et des signaux qu'on appelle KCLK (horloge clavier) et KDAT (données clavier). KCLK est uni-directionnel et est toujours piloté par le clavier; KDAT est piloté à la fois par le clavier et par l'ordinateur. Les deux signaux sont à collecteur ouvert, il y a des résistances élévatrices dans le clavier (dans le micro-processeur du clavier) et dans l'ordinateur.

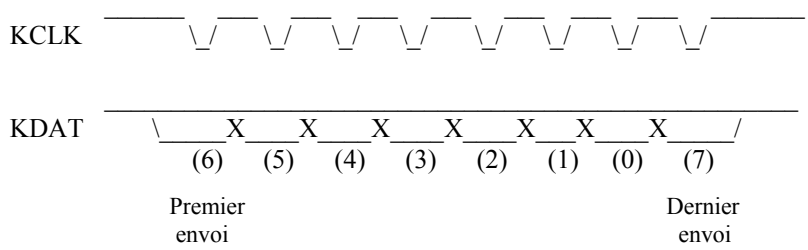
LA COMMUNICATIONS DANS LE CLAVIER

Le clavier transmet des mots de données 8 bits en série à l'unité principale. Avant le début de la transmission, KCLK et KDAT sont à l'état haut. Le clavier commence la transmission en envoyant le premier bit de données (sur KDAT), suivi d'une impulsion sur KCLK (signal haut, puis bas); ensuite il envoie le deuxième bit de données et une impulsion à KCLK jusqu'à ce qu'il ait envoyé tous les huit bits de données. A la fin de la dernière impulsion KCLK, le clavier remet encore une fois KDAT à l'état haut.

Lorsque l'ordinateur a reçu le huitième bit, il doit émettre une impulsion KDAT à l'état bas, pendant au moins 1 (une) micro-seconde en tant que signal poignée de main au clavier. La détection de ce signal du côté du clavier utilise typiquement un verrou 'hardware'. Le clavier doit avoir la possibilité de détecter une impulsion supérieure ou égale à 1 micro-seconde. Le 'software' DOIT envoyer une impulsion à la ligne à l'état bas pendant 85 micro-secondes pour garantir la compatibilité avec tous les modèles de clavier.

Tous les codes transmis à l'ordinateur subissent une rotation d'un bit avant transmission. Par conséquent, l'ordre transmis sera 6-5-4-3-2-1-0-7. L'explication est fournie par le fait qu'on transmet en dernier le drapeau haut/bas pour générer un code touche relâchée à transmettre au cas où le clavier serait forcé de restaurer une perte de synchro (expliquée plus en détail ci-dessous).

La ligne KDAT est zéro-active; ce qui signifie qu'un niveau haut (+5 Volts) est interprété en tant que 0 et qu'un niveau bas (0 Volts) est interprété en tant que 1.



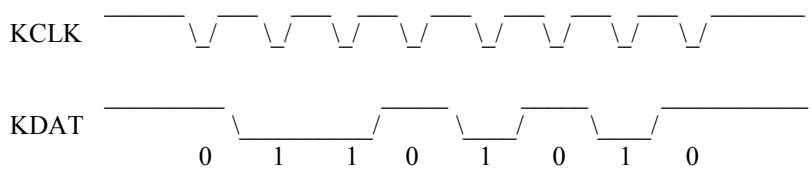
Le processeur du clavier met à 1 la ligne KDAT pendant 20 micro-secondes environ avant de mettre KCLK à l'état bas. KCLK reste à l'état bas 20 micro-secondes, puis passe à nouveau à l'état haut. Le processeur attend 20 micro-secondes supplémentaires avant de modifier KDAT.

Par conséquent, le débit de bits pendant la transmission est d'environ 60 micro-secondes par bit, c'est-à-dire 17 kilo-bits/seconde.

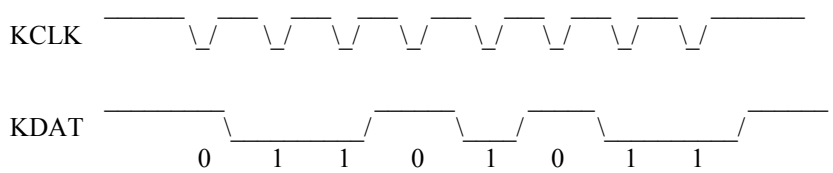
CODE DES TOUCHES

Chaque touche possède un code qui lui est associé (cf. le tableau joint). Le code des touches a toujours une longueur de 7 bits. Le huitième bit est un drapeau "touche appuyée/touche relâchée"; un 0 (niveau haut) signifie qu'on a appuyé sur une touche, tandis qu'un 1 (niveau bas) signifie qu'on vient de relâcher la touche (la touche CAPS LOCK est différente - voir ci-dessous).

Par exemple, voici un diagramme de la touche "B" sur laquelle on vient d'appuyer. Le code de la touche "B" est égal à $35 = 00110101; mais, suite à la rotation, les bits transmis seront 01101010.



Dans l'exemple qui suit, on relâche la touche "B". Le code de la touche sera toujours $35, si l'on excepte que le bit 7 est mis à 1 pour indiquer "touche relâchée". Ce qui donne un code égal à $B5 = 10110101. Après rotation, la transmission donne 01101011 :



LA TOUCHE 'CAPS LOCK'

Cette touche est différente de toutes les autres car elle ne génère un code de touche que si l'on appuie sur elle et jamais quand on la relâche. Toutefois, on utilise toujours le bit haut/bas. Quand on appuie sur la touche CAPS LOCK, celle-ci allume une diode luminescente incorporée, le bit haut/bas se mettra à 0; quand on la relâche, CAPS LOCK éteint la diode, le bit haut/bas se mettra à 1.

LA CONDITION "PERTE DE SYNCHRO"

Le bruit ou d'autres incidents peuvent provoquer la perte de synchro du clavier avec l'ordinateur. Ce qui signifie que le clavier a terminé de transmettre un code, mais que l'ordinateur en est toujours au milieu de la réception.

Si cela devait se produire, le clavier ne recevrait pas son impulsion poignée de main à la fin de la transmission. Si cette impulsion n'arrive pas dans les 143 milli-secondes depuis le dernier signal d'horloge de transmission, le clavier présume que l'ordinateur est toujours en train d'attendre la suite de la transmission et qu'il y a eu, par conséquent, perte de synchronisation. Le clavier, alors, essaiera de restaurer la synchro en passant en "mode re-synchro". Dans ce mode, le clavier envoie un signal composé de 1 et continue à attendre. Ce processus continuera tant qu'une impulsion poignée de main ne se sera pas produite.

Une fois la synchro restaurée, le clavier aura envoyé un caractère pour rien à l'ordinateur. C'est la raison pour laquelle le drapeau touche appuyée-touche relâchée est toujours transmis en dernier. Etant donné que le clavier envoie un signal composé de 1 pour restaurer la synchro, le caractère "pour rien" ainsi transmis apparaîtra en tant que touche relâchée, ce qui est moins dangereux que de transmettre une touche appuyée.

Au cas où le clavier détecterait qu'il a perdu la synchro, il présumera que l'ordinateur n'est pas arrivé à réceptionner le code de touche qu'il avait essayé de lui transmettre. Etant donné que l'ordinateur est incapable de détecter une perte de synchro, il incombe au clavier d'informer l'ordinateur de la catastrophe. Il le fait en transmettant à l'ordinateur un code de "perte de synchro" (valeur $F9 = 11111001). Ensuite, il transmet à nouveau le code qui a été "embrouillé".

NOTE

La seule raison de transmettre à l'ordinateur un code de "perte de synchro" est d'informer le 'software' que quelque chose pourrait déraper. Le code "perte de synchro" n'aide pas à rétablir le processus de récupération étant donné qu'on ne peut effacer le code de touche supplémentaire et qu'on pourrait simplement transmettre à nouveau un code de touche correct sans informer l'ordinateur qu'une erreur s'est produite dans le code précédent.

SEQUENCE A LA MISE SOUS TENSION

Dans des circonstances normales, il y a deux façons d'alimenter le clavier : <1> l'ordinateur peut être sous tension avec son clavier branché, ou <2> on peut brancher le clavier sur un ordinateur déjà sous tension. Le clavier et l'ordinateur doivent pouvoir gérer les deux situations sans rien déranger.

Lors d'une mise sous tension, la première chose que fait le clavier est une opération d'auto-test. celle-ci comprend un test de la somme de contrôle de la ROM, un test simplifié de la RAM et un test de la minuterie de garde. Quand on met le clavier sous tension (ou qu'on le relance - cf. plus bas), il ne doit rien transmettre tant qu'il n'a pas achevé sa synchronisation avec l'ordinateur. Il le fait en envoyant lentement des impulsions de bits à 1, comme décrit ci-dessus, tant qu'il ne reçoit pas une impulsion poignée de main.

Si on branche le clavier avant la mise sous tension, celui-ci peut continuer ce processus pendant plusieurs minutes étant donné que l'ordinateur lutte pour s'amorcer et commencer à tourner. Le clavier doit continuer à envoyer des 1 pendant tout le temps nécessaire tant qu'il ne reçoit pas son signal poignée de main.

Si on branche le clavier après la mise sous tension, il faudra pas moins de huit impulsions pour mettre en place la synchro. Par conséquent, dans ce cas, l'ordinateur peut se trouver dans un état imaginable quelconque mais ne devra pas être affecté défavorablement par le caractère en plus qu'il pourra recevoir. En outre, comme il reçoit une touche relâchée, le dommage devrait être minime. Le gestionnaire de clavier doit pouvoir anticiper cet événement et être à même de le gérer, comme devrait le faire toute application utilisant les codes de touches bruts ['raw'].

NOTE

Le clavier ne devra pas transmettre un code "perte de synchro" après une re-synchronisation due à une mise sous tension ou à un re-démarrage, mais seulement après une re-synchronisation due à un délai d'attente poignée de main.

Une fois que le clavier et l'ordinateur sont synchronisés, le clavier doit informer l'ordinateur des résultats de l'auto-test. Si, pour une raison quelconque, celui-ci a échoué, il transmet un code de type "l'auto-test a échoué" (valeur $FC = 11111100) (le clavier n'attend pas une impulsion poignée de main après avoir envoyé le code du type "l'auto-test a échoué". Après ceci, le processeur du clavier entre dans une boucle au cours de laquelle il fait clignoter la diode de la touche CAPS LOCK pour informer l'utilisateur qu'il vient d'échouer. Les clignotements sont codés sous forme d'une suite d'un, deux, trois ou quatre clignotements, approximativement une suite par seconde :

One blink	Mauvaise somme de contrôle ROM.
Two blinks	Test de la RAM a échoué.
Three blinks	Test minuterie de garde a échoué.
Four blinks	Un cour-circuit existe entre deux lignes de rangées horizontales ou l'une des sept touches spéciales (non implémenté).

Si l'auto-test réussit, le clavier alors continue à transmettre chaque touche sur laquelle on appuiera. En premier lieu, il envoie un code de "flux de touche pour amorcer la mise sous tension" (valeur $FD = 11111101), suivi des codes de touche de toutes les touches relâchées (avec un drapeau touche appuyée/touche relâchée mis à "bas" pour chaque touche), puis envoie un code de "flux pour désamorcer la touche" (valeur $FE = 11111110). Pour finir, il met en veilleuse la diode luminescente incorporée dans la touche CAPS LOCK. Ceci marque la fin de la séquence de démarrage, puis le processus normal se poursuit.

Par conséquent, la séquence habituelle des événements sera la suivante : mise sous tension; synchronisation; transmission de "flux de touche pour amorcer la mise sous tension" ($FD); transmission de "flux pour désamorcer la touche" ($FE).

AVERTISSEMENT DE REINITIALISATION [REINITIALISATION A CHAUD]

NOTE

Cette caractéristique, disponible sur certains claviers de l'A1000 et de l'A2000, n'apparait pas sur tous les Amiga.

Le clavier a comme tâche supplémentaire celle de réinitialiser l'ordinateur suite à une commande de l'utilisateur. Celui-ci amorce une Réinitialisation à Chaud en appuyant simultanément sur la touche CTRL et sur les deux touches AMIGA.

Le clavier répond à cette entrée en synchronisant toute opération de transmission en attente. Le clavier, ensuite, envoie un "avertissement de réinitialisation" à l'Amiga. Cette opération informe le 'software' de l'Amiga qu'il doit mettre fin à tout traitement en suspens (tel qu'un DMA de l'unité de disquette) et qu'il doit se préparer à la réinitialisation du système.

Une séquence spécifique d'opérations garantit que l'Amiga est en état de répondre à un avertissement de réinitialisation. Le clavier envoie deux véritables codes de touche "avertissement de réinitialisation". L'Amiga doit envoyer un signal poignée de main au premier code, comme pour une frappe de touche normale, autrement le clavier passe directement en état de Réinitialisation Matérielle. Lors du deuxième code "avertissement de réinitialisation", l'Amiga doit mettre KDAT à l'état bas en 250 milli-secondes, autrement le clavier passe directement en état de Réinitialisation Matérielle. Si tous les tests se passent convenablement, l'Amiga dispose d'un total de 10 secondes pour exécuter un traitement d'urgence. Lorsque l'Amiga met à nouveau KDAT à l'état haut, le clavier est enfin en état d'exécuter une réinitialisation matérielle du système.

Si l'Amiga échoue dans sa tâche de mettre KDAT à l'état haut dans les dix secondes, la Réinitialisation Matérielle est néanmoins validée.

REINITIALISATION MATERIELLE [HARD RESET]

NOTE

Ceci se produit après une Réinitialisation Matérielle. Les explications qui suit sont valables pour tous les claviers, à l'exception de celui de l'A500.

Le clavier réinitialise matériellement l'Amiga en mettant KCLK à l'état bas et en faisant démarrer une minuterie pour 500 milli-secondes. Lorsqu'on relâche une ou plusieurs touches ET que les 500 milli-secondes se sont écoulées, le clavier "relâchera" KCLK. 500 milli-secondes sont le temps minimal pendant lequel on peut garder KCLK à l'état bas. Le temps maximal dépendra de la durée pendant laquelle l'utilisateur gardera appuyées les trois touches ensemble.

NOTE

L'ensemble des circuits de la carte mère de l'Amiga détecte une impulsion KCLK de 500 milli-secondes.
Après avoir relâché KCLK, le clavier sautera à son code de démarrage (REINITIALISATION interne). Ceci aura pour effet d'initialiser le clavier de la même façon que l'aurait fait une mise sous tension à froid.

NOTE
Le clavier doit renvoyer le "flux de touche de mise sous tension" !

CODES SPECIAUX

Les codes spéciaux que le clavier emploie pour communiquer avec l'unité centrale sont récapitulés ci-dessous.

NOTE

Les codes spéciaux sont des chiffres 8 bits; ceux-ci n'ont pas de drapeau haut/bas qui leur est associé.
Toutefois, l'ordre de transmission des bits est identique à celui décrit précédemment.

CODE	NOM	EXPLICATION
78		Avertissement de réinitialisation. On a appuyé sur CTRL-AMIGA-AMIGA. L'ordinateur sera réinitialisé au bout de 10 secondes (cf. texte).
F9		Le dernier code de touche était incorrect, le prochain code qui sera transmis sera identique au précédent (utilisé quand le clavier et l'unité centrale perdent leur synchronisation).
FA		Dépassement de capacité du tampon de sortie du clavier.
FB		Inutilisé (indiquait un échec du contrôleur).
FC		L'auto-test du clavier a échoué.
FD		Flux de touche pour amorcer la mise sous tension (touches appuyées à la mise sous tension).
FE		Flux de touche pour désamorcer la mise sous tension.
FF		Inutilisé (indiquait une interruption).

TABLE DE LA MATRICE

Column	Row 5 (Bit 7)	Row 4 (Bit 6)	Row 3 (Bit 5)	Row 2 (Bit 4)	Row 1 (Bit 3)	Row 0 (Bit 2)
15 (PD.7)	(spare) (0E)	(spare) (1C)	(spare) (2C)	(spare) (47)	(spare) (48)	(spare) (49)
14 (PD.6)	* note 1 (5D)	<SHIFT> note 2 (30)	CAPS LOCK (62)	TAB (42)	~ ` (00)	ESC (45)
13 (PD.5)	+ note 1 (5E)	Z (31)	A (20)	Q (10)	! 1 (01)	(note 1 (5A)
12 (PD.4)	9 note 3 (3F)	X (32)	S (21)	W (11)	@ 2 (02)	F1 (50)
11 (PD.3)	6 note 3 (2F)	C (33)	D (22)	E (12)	# 3 (03)	F2 (51)
10 (PD.2)	3 note 3 (1F)	V (34)	F (23)	R (13)	$ 4 (04)	F3 (52)
9 (PD.1)	. note 3 (3C)	B (35)	G (24)	T (14)	% 5 (05)	F4 (53)
8 (PD.0)	8 note 3 (3E)	N (36)	H (25)	Y (15)	^ 6 (06)	F5 (54)
7 (PC.7)	5 note 3 (2E)	M (37)	J (26)	U (16)	& 7 (07)	) note 1 (5B)
6 (PC.6)	2 note 3 (1E)	< , (38)	K (27)	I (17)	* 8 (08)	F6 (55)
5 (PC.5)	ENTER note 3 (43)	> . (39)	L (28)	O (18)	(9 (09)	/ note 1 (5C)

Column	Row 5 (Bit 7)	Row 4 (Bit 6)	Row 3 (Bit 5)	Row 2 (Bit 4)	Row 1 (Bit 3)	Row 0 (Bit 2)
4 (PC.4)	7 note 3 (3D)	? / (3A)	: ; (29)	P (19)	) 0 (0A)	F7 (56)
3 (PC.3)	4 note 3 (2D)	(spare) (3B)	" ' (2A)	{ [(1A)	_ - (0B)	F8 (57)
2 (PC.2)	1 note 3 (1D)	SPACE BAR (40)	<RET> note 2 (2B)	}] (1B)	+ = (0C)	F9 (58)
1 (PC.1)	0 note 3 (0F)	BACK SPACE (41)	DEL (46)	RETURN (44)	| \ (0D)	F10 (59)
0 (PC.0)	- note 3 (4A)	CURS DOWN (4D)	CURS RIGHT (4E)	CURS LEFT (4F)	CURS UP (4C)	HELP (5F)

note 1 : Seulement les claviers de l'A500 et de l'A2000 (pavé numérique).
note 2 : Seulement les claviers internationaux (ces touches sont découpées dans les touches de la version ASCII américaine qui sont plus larges). La touche qui génère $30 est découpée dans la touche 'shift' gauche. La touche $2B est découpée de la touche 'return'. Ces touches portent des inscriptions qui sont propres à chaque pays.
note 3 : Pavé numérique.

Le tableau qui suit indique les touches qu'on a la possibilité de lire séparément. Ces touches ne génèrent jamais de caractères répétés ou fantômes.

(Bit 6)	(Bit 5)	(Bit 4)	(Bit 3)	(Bit 2)	(Bit 1)	(Bit 0)
LEFT AMIGA (66)	LEFT ALT (64)	LEFT SHIFT (60)	CTRL (63)	RIGHT AMIGA (67)	RIGHT ALT (65)	RIGHT SHIFT (61)

APPENDICE I

SPECIFICATIONS SUR L'INTERFACE DU CONNECTEUR DE L'UNITE DE DISQUETTES EXTERNE

GENERALITES

Le connecteur femelle 23 broches, sur l'arrière de l'unité principale de l'ordinateur, est utilisé pour interfacer et contrôler des périphériques qui génèrent et reçoivent des données MFM. On peut parvenir à cette interface soit par une ressource soit sous le contrôle d'un lecteur de disquettes.
Les pages qui suivent décrivent cette interface dans les deux situations.

TABLEAU RECAPITULATIF

Broche#	Nom	Notes	
1	RDY-	E/S	ID et disquette prête.
2	DKRD-	E	Entrée MFM.
3	GRND	G	Masse
4	GRND	G	Masse
5	GRND	G	Masse
6	GRND	G	Masse
7	GRND	G	Masse
8	MTRXD-	S	Contrôle du moteur.
9	SEL2B-	S*	Sélection lecteur 2
10	DRESB-	S	Réinitialisation
11	CHNG-	E/S	Changement de disquette
12	+5v	PWR	540 milli en moyenne, 870 milli en pointe
13	SIDEB-	S	Face 1 si 0 actif
14	WRPRO-	E/S	Protection contre l'écriture
15	TK0-	E/S	Piste 0
16	DKWEB-	S	Autorisation écriture
17	DKWDB-	S	Ecriture données
18	STEPB-	S	Pas
19	DIRB	S	Direction (état haut en sortie)
20	SEL3B-	S*	Sélection lecteur 3
21	SEL1B-	O*	Sélection lecteur 1
22	INDEX-	E/S	Index
23	+12v	PWR	120 milli en moyenne, 370 milli en pointe

EXPLICATION DES ABREVIATIONS :

G MASSE, noter le blindage du connecteur mis à la masse.

E entrée élevée à 5 Volts par une résistance de 1 KOhm.

E/S entrée dans le lecteur, mais entrée bi-directionelle (resistance élévatrice d'1 KOhm)

S sortie élevée à 5 Volts par le biais d'une résistance d'1 KOhm

S* sortie, ressources indépendantes

PWR disponible pour une utilisation externe, mais actuellement employé par le lecteur de disquettes externe.

SIGNAUX EN RAPPORT AVEC LE PILOTAGE D'UNE UNITE DE DISQUETTES

Ce qui suit décrit l'interface sous le contrôle d'un lecteur de disquettes :

SEL1B-, SEL2B-, SEL3B-
Sélectionne les lignes pour les trois lecteurs de disquettes externes, lignes "zéro actives".

TK0-
Le lecteur sélectionné met ce signal à l'état bas au cas où sa tête de lecture se trouverait sur la piste 00.

RDY-
Lorsque le moteur d'un lecteur de disquettes est en service, cette ligne indique que l'unité de disquettes sélectionnée est installée et que son moteur tourne à la bonne vitesse. Le lecteur ignore ce signal. Lorsque le moteur est hors service, cette ligne est employée en tant que ligne pour les données d'ID. Voir plus loin.

WPRO- (Broche #14)
Un lecteur de disquettes sélectionné met ce signal à l'état bas si on a inséré une disquette protégée contre l'écriture.

INDEX- (Broche #22)
Le lecteur sélectionné met ce signal à l'état bas à chaque tour complet du moteur.

SIDEB- (Broche #13)
Le système envoie ce signal à tous les lecteurs se trouvant à l'état bas pour la face 1, à l'état haut pour la face 2.

STEPB- (Broche #18)
Impulsion pour faire avancer d'un pas la tête du lecteur sélectionné.

DIRB (Broche #19)
Le système met ce signal à l'état haut ou bas pour informer de quelle manière doit avancer le lecteur sélectionné lorsqu'il reçoit une impulsion STEPB-. "Bas" signifie "step in", "Haut" signifie "step out"

DKRD- (Broche #2)
C'est sur cette ligne que le lecteur sélectionné enverra en sortie les données lues.

DKWDB- (Broche #17)
Par le biais de ce signal, le système envoie les données à écrire à toutes les unités de disquettes. Les données sont écrites seulement si DKWEB- est zéro active (état bas). Les données ne seront écrites que dans les unités de disquettes qui auront été sélectionnées.

DKWEB- (Broche #16)

Ce signal provoque le début de l'écriture (par DKWDB-) des données sur la disquette sélectionnée.

CHNG- (Broche #11)

Un lecteur sélectionné mettra le signal à l'état bas si on a mis à 1 son verrou interne de "changement de disquette".
Ce verrou est mis à 1, si le lecteur de disquettes est mis sous tension ou s'il n'y a pas de disquette dans le lecteur.
Pour débloquer le verrou, le système devra sélectionner le lecteur et faire avancer la tête de lecture/écriture.
Naturellement, le verrou ne se débloquera pas s'il n'y a pas de disquette dans le lecteur.

MTRXD- (Broche #8)

C'est la ligne de contrôle du moteur des quatre lecteurs de disquettes.
Quand le système souhaite activer le moteur d'un lecteur de disquettes, il invalide d'abord le lecteur (si celui-ci avait été sélectionné), met MTRXD- à l'état bas et sélectionne ensuite le lecteur. Pour mettre le moteur hors service, le système invalide le lecteur, met MTRXD- à l'état haut, puis sélectionne le moteur. Le système mettra toujours MTRXD- à 1, pendant au moins 1,4 micro-secondes, avant de sélectionner un lecteur, et ne modifiera pas l'état de MTRXD-, pendant au moins 1,4 micro-secondes, après l'avoir sélectionné. Tous les lecteurs externes devront posséder une logique équivalente à celle d'une bascule D, dont une entrée D serait le signal MTRXD- et dont une entrée d'horloge serait activée par la transition en/hors service (passage de l'état haut à l'état bas) du signal SELxB-. Comme nous l'avons fait remarquer ci-dessus, le temps de mise en route et celui d'attente de MTRXD -tout en respectant SELxB- dureront au moins 1,4 micro-secondes. La sortie de cette bascule contrôle le moteur du lecteur de disquettes.
De cette manière, le système a la possibilité de contrôler les quatre moteurs en n'employant qu'un seul signal sur la ligne (MTRXD-).

DRESB- (Broche #10)

Ce signal est la version tamponnée du signal de réinitialisation du système.
Trois facteurs peuvent activer ce signal (le mettre à l'état bas) :

- Mise sous tension du système (DRESB- passera à l'état bas, de façon approximative, pendant une seconde);
- L'unité centrale du système exécute une instruction RESET (DRESB- passera à l'état bas approximativement pendant 17 micro-secondes);
- Réinitialisation matérielle à partir du clavier (dure tant qu'on appuie sur les touches de réinitialisation du clavier).

Les lecteurs de disquettes externes devraient répondre à DRESB- en débloquant eux-mêmes les bascules de leurs moteurs, ainsi que la protection contre l'écriture.

Une tension de 3,75 Volts ou une tension inférieure sur la broche +5 Volts exige que les unités externes s'occupent de leur propre protection contre l'écriture, ainsi que de la réinitialisation de leur moteur sur la ligne.

I.D. DU PERIPHERIQUE

Cette interface accepte une méthode qui permet d'établir le type de disquette(s) qui lui est fournie. La séquence I.D. est la suivante.

1. Mettre MTRXD- à l'état bas.
2. Mettre SELxB- à l'état bas
3. Mettre SELxB- à l'état haut.
4. Mettre MTRXD- à l'état haut.
5. Mettre SELxB- à l'état bas.
6. Mettre SELxB- à l'état haut.
7. Mettre SELxB- à l'état bas.
8. Lire et sauvegarder l'état de RDY.
9. Mettre SELxB- à l'état haut.

Répéter les points de 6 à 9 quinze fois.

Convertir les 16 valeurs de RDY- en un mot 16 bits. Le bit le plus significatif sera la première valeur et ainsi de suite. La valeur 16 bits est l'I.D du périphérique.

Les I.D. ci-dessous sont définis :

0000	0000	0000	0000	Réservé
1111	1111	1111	1111	Standard Amiga 3.25
1010	1010	1010	1010	Réservé
0101	0101	0101	0101	48 TPI [pistes par pouce], double densité, double face
1000	0000	0000	0000	Réservé
0111	1111	1111	1111	Réservé
0000	1111	xxxx	xxxx	Libre pour les utilisateurs
1111	0000	xxxx	xxxx	Extension réservée
xxxx	0000	0000	0000	Réservé
xxxx	1111	1111	1111	Réservé
0011	0011	0011	0011	Réservé
1100	1100	1100	1100	Réservé

APPENDICE J

FICHIER 'INCLUDE' POUR LES EXEMPLES 'HARDWARE'

Cette appendice contient un fichier 'include' qui relie les noms des registres 'hardware', donnés dans l'Appendice A et l'Appendice B, à des noms qui pourront être résolus par les fichiers 'include' standard. L'emploi de ces noms, dans les sections de code de ce manuel met l'accent sur ce qu'est en train de faire le programme plutôt que de s'enliser dans les noms des fichiers 'include'.

Tous les exemples de programme de ce manuel se réfèrent aux noms que nous donnons dans ce fichier.

```
                IFND    HARDWARE_HW_EXAMPLES_I
HARDWARE_HW_EXAMPLES_I  SET     1
**
**
**
**      NomDeFichier : hardware/hw_examples.i
**      $Edition : 1.3 $
**
**      (C) Copyright 1985,1986,1987,1988,1989 Commodore-Amiga, Inc.
**          Tous Droits réservés
**
**
*****************************************************************************

   IFND HARDWARE_CUSTOM_I
   INCLUDE "hardware/custom.i"
   ENDC

*****************************************************************************
*
* Ce fichier 'include' est conçu pour être employé avec les exemples
* 'hardware' du manuel. Ce fichier définit le nom des registres en
* s'appuyant sur le fichier de définition hardware/custom.i. Il n'existe
* pas de version C du fichier.

*****************************************************************************
*
* Cette instruction pour le Copper lui permettra d'attendre à l'infini étant donné que la commande 'wait',
* décrite dans le corps du programme, ne se produira jamais.
*
COPPER_HALT             equ     $FFFFFFFE
*
*****************************************************************************
*
* Celui-ci est l' 'offset' dans l'espace adressable du 680x0 par rapport aux registres des puces spécialisées.
* On a le même 'offset' si on fait une édition de liens avec l'AMIGA.lib.
*
CUSTOM                  equ     $DFF000
*
* Différents registres de contrôle
*
DMACONR                 equ     dmaconr         ; Mis en lettres majuscules...
VPOSR                   equ     vposr           ; "     "
VHPOSR                  equ     vhpor           ; "     "
JOY0DAT                 equ     joy0dat         ; "     "
JOY1DAT                 equ     joy1dat         ; "     "
CLXDAT                  equ     clxdat          ; "     "
ADKCONR                 equ     adkconr         ; "     "
POT0DAT                 equ     pot0dat         ; "     "
POT1DAT                 equ     pot1dat         ; "     "
POTINP                  equ     potinp          ; "     "
SERDATR                 equ     serdatr         ; "     "
INTENAR                 equ     intenar         ; "     "
INTREQR                 equ     intreqr         ; "     "
REFPTR                  equ     refptr          ; "     "
VPOSW                   equ     vposw           ; "     "
VHPOSW                  equ     vhposw          ; "     "
SERDAT                  equ     serdat          ; "     "
SERPER                  equ     serper          ; "     "
POTGO                   equ     potgo           ; "     "
JOYTEST                 equ     joyteat         ; "     "
STREQU                  equ     strequ          ; "     "
STRVBL                  equ     strvbl          ; "     "
```

```
STRHOR          equ     atrhor          ; "     "
STRLONG         equ     atrlong         ; "     "
DIWSTRT         equ     diwatrt         ; "     "
DIWSTOP         equ     diwatop         ; "     "
DDFSTRT         equ     ddfatrt         ; "     "
DDFSTOP         equ     ddfatop         ; "     "
DMACON          equ     dmacon          ; "     "
INTENA          equ     intena          ; "     "
INTREQ          equ     intreq          ; "     "
*
* Registres de contrôle disquette
*
DSKBYTR         equ     dakbytr         ; Mis en lettres majuscules...
DSKPT           equ     dakpt           ; "     "
DSKPTH          equ     dakpt
DSKPTL          equ     dakpt+$02
DSKLEN          equ     daklen          ; "     "
DSKDAT          equ     dakdat          ; "     "
DSKSYNC         equ     dakaync         ; "     "
*
* Registres du 'blitter'
*
BLTCON0         equ     bltcon0         ; Mis en lettre majuscules...
BLTCON1         equ     bltcon1         ; "     "
BLTAFWM         equ     bltafwm         ; "     "
BLTALWM         equ     bltalwm
BLTCPT          equ     bltcpt          ; "     "
BLTCPTH         equ     bltcpt
BLTCPTL         equ     bltcpt+$02
BLTBPT          equ     bltbpt          ; "     "
BLTBPTH         equ     bltbpt
BLTBPTL         equ     bltbpt+$02
BLTAPT          equ     bltapt          ; "     "
BLTAPTH         equ     bltapt
BLTAPTL         equ     bltapt+$02
BLTDPT          equ     bltdpt          ; "     "
BLTDPTH         equ     bltdpt
BLTDPTL         equ     bltdpt+$02
BLTSIZE         equ     bltaize         ; "     "
BLTCMOD         equ     bltcmod         ; "     "
BLTBMOD         equ     bltbmod         ; "     "
BLTAMOD         equ     bltamod         ; "     "
BLTDMOD         equ     bltdmod         ; "     "
BLTCDAT         equ     bltcdat         ; "     "
BLTBDAT         equ     bltbdat         ; "     "
BLTADAT         equ     bltadat         ; "     "
BLTDDAT         equ     bltddat         ; "     "
*
* Registres de contrôle du Copper
*
COPCON          equ     copcon          ; Mis en lettres majuscules...
COPINS          equ     copina          ; "     "
COPJMP1         equ     copjmp1         ; "     "
COPJMP2         equ     copjmp2         ; "     "
COP1LC          equ     cop1lc          ; "     "
COP1LCH         equ     cop1lc
COP1LCL         equ     cop1lc+$02
COP2LC          equ     cop2lc          ; "     "
COP2LCH         equ     cop2lc
COP2LCL         equ     cop2lc+$02
*
* Registres des canaux audio
*
```

```
ADKCON          equ     adkcon          ; Mis en lettres majuscules...

AUD0LC          equ     aud0
AUD0LCH         equ     aud0
AUD0LCL         equ     aud0+$02
AUD0LEN         equ     aud0+$04
AUD0PER         equ     aud0+$06
AUD0VOL         equ     aud0+$08
AUD0DAT         equ     aud0+$0A

AUD1LC          equ     aud1
AUD1LCH         equ     aud1
AUD1LCL         equ     aud1+$02
AUD1LEN         equ     aud1+$04
AUD1PER         equ     aud1+$06
AUD1VOL         equ     aud1+$08
AUD1DAT         equ     aud1+$0A

AUD2LC          equ     aud2
AUD2LCH         equ     aud2
AUD2LCL         equ     aud2+$02
AUD2LEN         equ     aud2+$04
AUD2PER         equ     aud2+$06
AUD2VOL         equ     aud2+$08
AUD2DAT         equ     aud2+$0A

AUD3LC          equ     aud3
AUD3LCH         equ     aud3
AUD3LCL         equ     aud3+$02
AUD3LEN         equ     aud3+$04
AUD3PER         equ     aud3+$06
AUD3VOL         equ     aud3+$08
AUD3DAT         equ     aud3+$0A
*
* Les registres des plans de bits
*
BPL1PT          equ     bplpt+$00
BPL1PTH         equ     bplpt+$00
BPL1PTL         equ     bplpt+$02
BPL2PT          equ     bplpt+$04
BPL2PTH         equ     bplpt+$04
BPL2PTL         equ     bplpt+$06
BPL3PT          equ     bplpt+$08
BPL3PTH         equ     bplpt+$08
BPL3PTL         equ     bplpt+$0A
BPL4PT          equ     bplpt+$0C
BPL4PTH         equ     bplpt+$0C
BPL4PTL         equ     bplpt+$0E
BPL5PT          equ     bplpt+$10
BPL5PTH         equ     bplpt+$10
BPL5PTL         equ     bplpt+$12
BPL6PT          equ     bplpt+$14
BPL6PTH         equ     bplpt+$14
BPL6PTL         equ     bplpt+$16

BPLCON0         equ     bplcon0         ; Mis en lettres majuscules...
BPLCON1         equ     bplcon1         ; "         "
BPLCON2         equ     bplcon2         ; "         "
BPL1MOD         equ     bpl1mod         ; "         "
BPL2MOD         equ     bpl2mod
```

```
DPL1DATA        equ     bpldat+$00
DPL2DATA        equ     bpldat+$02
DPL3DATA        equ     bpldat+$04
DPL4DATA        equ     bpldat+$06
DPL5DATA        equ     bpldat+$08
DPL6DATA        equ     bpldat+$0A
*
* Registres de contrôle des sprites
*
SPR0PT          equ     Sprpt+$00
SPR0PTH         equ     SPR0PT+$00
SPR0PTL         equ     SPR0PT+$02
SPR1PT          equ     sprpt+$04
SPR1PTH         equ     SPR1PT+$00
SPR1PTL         equ     SPR1PT+$02
SPR2PT          equ     sprpt+$08
SPR2PTH         equ     SPR2PT+$00
SPR2PTL         equ     SPR2PT+S02
SPR3PT          equ     sprpt+$0C
SPR3PTH         equ     SPR3PT+$00
SPR3PTL         equ     SPR3PT+$02
SPR4PT          equ     sprpt+$10
SPR4PTH         equ     SPR4PT+$00
SPR4PTL         equ     SPR4PT+$02
SPR5PT          equ     sprpt+$14
SPR5PTH         equ     SPR5PT+$00
SPR5PTL         equ     SPR5PT+$02
SPR6PT          equ     sprpt+$18
SPR6PTH         equ     SPR6PT+$00
SPR6PTL         equ     SPR6PT+$02
SPR7PT          equ     sprpt+$1C
SPR7PTH         equ     SPR7PT+$00
SPR7PTL         equ     SPR7PT+$02
;
; Note : SPRxDATB est défini comme se trouvant à +$06 à partir de SPRxPOS.
; Par conséquent, on devrait définir sd_datab en tant que $06, dans le
; fichier 'include' assembleur hardware/custom.i, où il est défini de
; manière incorrecte en tant que $08.
;
SPR0POS         equ     spr+$00
SPR0CTL         equ     SPR0POS+sd_ct1
SPR0DATA        equ     SPR0POS+sd dataa
SPR0DATB        equ     SPR0POS+$06    ; devrait utiliser sd_datab ...

SPR1POS         equ     spr+$08
SPR1CTL         equ     SPR1POS+sd_ctl
SPR1DATA        equ     SPR1POS+sd_dataa
SPR1DATB        equ     SPR1POS+$06    ; devrait utiliser sd_datab ...

SPR2POS         equ     spr+$10
SPR2CTL         equ     SPR2POS+sd_ctl
SPR2DATA        equ     SPR2POS+sd_dataa
SPR2DATB        equ     SPR2POS+$06    ; devrait utiliser sd_datab ...

SPR3POS         equ     spr+$18
SPR3CTL         equ     SPR3POS+sd ctl
SPR3DATA        equ     SPR3POS+sd dataa
SPR3DATB        equ     SPR3POS+$06    ; devrait utiliser sd_datab ...

SPR4POS         equ     spr+$20
SPR4CTL         equ     SPR4POS+sd ctl
SPR4DATA        equ     SPR4POS+sd dataa
SPR4DATB        equ     SPR4POS+$06    ; devrait utiliser sd_datab ...
```

```
SPR5POS         equ     spr+$28
SPR5CTL         equ     SPR5POS+ad_ctl
SPR5DATA        equ     SPR5POS+Ad_dataa
SPR5DATB        equ     SPR5POS+$06     ; devrait utiliser sd_datab

SPR6POS         equ     spr+$30
SPR6CTL         equ     SPR6POS+ad_ctl
SPR6DATA        equ     SPR6POS+sd_dataa
SPR6DATB        equ     SPR6POS+$06     ; devrait utiliser sd_datab

SPR7POS         equ     spr+$38
SPR7CTL         equ     SPR7POS+sd_ctl
SPR7DATA        equ     SPR7POS+sd_dataa
SPR7DATB        equ     SPR7POS+$06     ; devrait utiliser sd_datab
*
* Registres de couleur...
*
COLOR00         equ     color+$00
COLOR01         equ     color+$02
COLOR02         equ     color+$04
COLOR03         equ     color+$06
COLOR04         equ     color+$08
COLOR05         equ     color+$0A
COLOR06         equ     color+$0C
COLOR07         equ     color+$0E
COLOR08         equ     color+$10
COLOR09         equ     color+$12
COLOR10         equ     color+$14
COLOR11         equ     color+$16
COLOR12         equ     color+$18
COLOR13         equ     color+$1A
COLOR14         equ     color+$1C
COLOR15         equ     color+$1E
COLOR16         equ     color+$20
COLOR17         equ     color+$22
COLOR18         equ     color+$24
COLOR19         equ     color+$26
COLOR20         equ     color+$28
COLOR21         equ     color+$2A
COLOR22         equ     color+$2C
COLOR23         equ     color+$2E
COLOR24         equ     color+$30
COLOR25         equ     color+$32
COLOR26         equ     color+$34
COLOR27         equ     color+$36
COLOR28         equ     color+$38
COLOR29         equ     color+$3A
COLOR30         equ     color+$3C
COLOR31         equ     color+$3E

*************************************************************************
**
**
                ENDC    ; HARDWARE_HW_EXAMPLES_I
```

LEXIQUE

AGNUS
L'une des trois puces spécialisées de l'Amiga. Contient le 'blitter', le Copper et l'ensemble des circuits de DMA.

ALIASING DISTORTION [DISTORSION]
Un effet secondaire de l'échantillonnage du son, par lequel se produisent deux fréquences additionnelles qui font apparaître de la distorsion sur la sortie sonore.

ALT KEYS [TOUCHES ALT]
Les deux touches du clavier à gauche et à droite des deux touches Amiga.

AMIGA KEYS [TOUCHES AMIGA]
Les deux touches du clavier à gauche et à droite de la barre d'espacement.

AMIGADOS
Le système d'exploitation de l'Amiga.

AMPLITUDE
La tension ou le courant de sortie exprimé comme volume d'un haut-parleur.

AMPLITUDE MODULATION [MODULATION EN AMPLITUDE]
Un moyen de produire un effet audio qui "épaissit" le son en employant un canal audio pour en moduler un autre.

ATTACH MODE [MODE 'ATTACH']
Pour les sprites, un mode dans lequel les sprites emploient deux canaux DMA pour disposer de plus de couleurs. Pour la production du son, la combinaison de deux canaux audio pour la modulation d'amplitude/de fréquence ou pour un son stéréo.

AUTOMATIC MODE [MODE AUTOMATIQUE]
Dans l'affichage des sprites, c'est le mode normal par lequel le canal DMA de sprite, une fois mis en route, prélève et affiche automatiquement toutes les données d'un sprite. En audio, c'est le mode normal par lequel le système prélève automatiquement les données sonores par le biais des DMA.

BARREL SHIFTER [DEPHASEUR A TAMBOUR]
Circuit du 'blitter' qui permet de déplacer des images à la limite du pixel.

BAUD RATE [TAUX D'EMISSION]
La vitesse de transmission des données dans un port série.

BEAM COUNTERS [COMPTEUR DE FAISCEAU]
Registres qui gardent trace de la position du faisceau vidéo.

BIT-MAP [MAPPE DE BITS]
La définition totale d'un affichage en mémoire, composée d'un plan de bits ou de plusieurs, comprenant aussi l'information sur la manière d'organiser un affichage rectangulaire.

BITPLANE [PLAN DE BITS]
Une suite contigüe de mots en mémoire destinée à l'affichage, traitée comme s'il s'agissait d'une forme rectangulaire.

BIT-PLANE ANIMATION [ANIMATION DES PLANS DE BITS]
Sorte d'animation de l'affichage par déplacement de données d'aires de jeu par l'intermédiaire du 'blitter'.

BLANKING INTERVAL [INTERVALLE DE SUPPRESSION]
Période de temps pendant laquelle le faisceau vidéo se trouve en dehors de la zone d'affichage.

BLITTER
Canal DMA utilisé pour copier des données et dessiner des lignes.

CHIP MEMORY [MEMOIRE 'CHIP']
Mémoire accessible aux puces spécialisées de l'Amiga. Sur la génération de machines actuelles, cette partie de mémoire commence à l'adresse 0. Voir aussi FAST MEMORY.

CLEAR [EFFACEMENT]
Donner à un bit la valeur 0.

CLI
Voir COMMAND LINE INTERFACE.

CLIPPING [DETOURAGE]
Il se produit si une partie de sprite se trouve à l'extérieur d'une fenêtre d'affichage et que, par conséquent, n'est pas visible.

COLLISION
Un moyen de détecter si les sprites, les aires de jeu ou les objets aires de jeu essaient de se chevaucher à une même position de pixels ou s'ils tentent de traverser des limites définies au préalable.

COLOR DESCRIPTOR WORDS [MOTS DU DESCRIPTEUR DE COULEUR]
Couple de mots qui définit chaque ligne d'un sprite.

COLOR INDIRECTION [INDIRECTION DE COULEUR]
Méthode employée par l'Amiga pour colorier individuellement des pixels, méthode par laquelle le chiffre binaire -formé de l'ensemble des bits qui définissent un pixel donné- fait référence à l'un des 32 registres de couleur.

COLOR PALETTE [PALETTE DES COULEURS]
Voir COLOR TABLE.

COLOR REGISTER [REGISTRE DE COULEUR]
L'un des 32 registres renfermant des couleurs qu'on peut définir.

COLOR TABLE [TABLE DE COULEURS]
L'ensemble des 32 registres de couleur.

COMMAND LINE INTERFACE [INTERFACE DE COMMANDE EN LIGNE]
Utilitaire et commandes de l'interface de commande en ligne avec le système.

COMPOSITE VIDEO [VIDEO COMPOSITE]
Un signal vidéo, transmis dans un seul câble coaxial comprenant des informations sur l'image et sur la synchro.

CONTROLLER [CONTROLEUR]
Périphérique 'hardware', tel qu'une souris ou un crayon optique, employé pour déplacer le pointeur ou pour avoir une autre entrée dans le système.

COORDINATES [COORDONNES]
Un couple de chiffres qu'on indique sous la forme (x,y), où x est un 'offset' à partir du côté gauche d'une fenêtre d'affichage et y un 'offset' à partir du haut de cette même fenêtre.

COPPER
Co-processeur d'affichage synchronisé qui se trouve dans l'une des puces spécialisées de l'Amiga et qui préside à l'affichage graphique.

COPROCESSOR [CO-PROCESSEUR]
Processeur qui ajoute son jeu d'instructions à celui du processeur principal.

CURSOR KEYS [TOUCHES POUR LE CURSEUR]
Touches pour déplacer quelque chose à l'écran.

DATA FETCH [PRELEVEMENT DE DONNEES]
Le nombre de mots prélevés à chaque ligne d'affichage.

DELAY [DELAI]
Dans un déroulement horizontal de l'aire de jeu, il indique de combien de pixels se décalera une image à chaque champ d'affichage. Le délai contrôle la vitesse du déroulement.

DENISE
L'un des trois puces spécialisées principales de l'Amiga. Contient l'ensemble des circuits pour la palette de couleur, les sprites et la sortie vidéo.

DEPTH [PROFONDEUR]
Nombre de plans de bits par affichage.

DIGITAL-TO-ANALOG CONVERTER [CONVERTISSEUR NUMERIQUE/ANALOGIQUE]
Un dispositif qui convertit une quantité binaire en niveau analogique.

DIRECT MEMORY ACCESS [ACCES DIRECT EN MEMOIRE]
Un artifice par lequel des périphériques intelligents ont la possibilité de lire ou d'écrire en mémoire, sans avoir à interrompre le processeur.

DISPLAY FIELD [CHAMP D'AFFICHAGE]
Un balayage total du faisceau vidéo de haut en bas dans l'écran d'affichage vidéo.

DISPLAY MODE [MODE AFFICHAGE]
L'un des principaux types d'affichage; par exemple, haute ou basse résolution, entrelacé ou non entrelacé, simple ou double aire de jeu.

DISPLAY TIME [TEMPS D'AFFICHAGE]
Le temps nécessaire à produire un champ d'affichage, approximativement 1/60e de seconde.

DISPLAY WINDOW [FENETRE D'AFFICHAGE]
La partie de la mappe de bits sélectionnée pour un affichage. Aussi, les dimensions réelles d'un affichage à l'écran.

DMA
Voir DIRECT MEMORY ACCESS.

DUAL-PLAYFIELD MODE [MODE DOUBLE AIRE DE JEU]
Un mode d'affichage qui permet de gérer deux mémoires d'affichage séparées, qui donne deux affichages qu'on a la possibilité de contrôler séparément en même temps.

EQUAL-TEMPERED SCALE [GAMME TEMPEREE]
Une gamme musicale où chaque note se trouve à 1/12 de la racine carrée de 2 au-dessus de la note plus grave.

EXEC
Primitives de bas niveau qui acceptent le système d'exploitation de l'Amiga.

FAST MEMORY [MEMOIRE 'FAST']
Mémoire à laquelle ne peuvent accéder les puces spécialisées. Pour ces dernières, il faut faire attention à ne leur présenter que des adresses se trouvant en mémoire 'chip'. Voir CHIP MEMORY.

FONT [FONTE]
Un ensemble de lettres, chiffres et symboles qui ont en commun une même taille et un même concept sous-jacent.

FREQUENCY [FREQUENCE]
Nombre de fois par seconde pendant lesquelles se répète une forme d'onde.

FREQUENCY MODULATION [MODULATION EN FREQUENCE]
Un moyen de modifier la qualité d'un son en utilisant un canal audio qui affecte la période d'une forme d'onde produite par un autre canal. La modulation en fréquence augmente ou diminue la hauteur d'un son.

GENLOCK
Un accessoire en option qui permet d'introduire dans un affichage des graphismes à partir d'une source vidéo extérieure.

HIGH RESOLUTION [HAUTE RESOLUTION]
Un mode d'affichage horizontal où, dans un affichage de dimensions normales, s'affichent 640 pixels par ligne horizontale.

HOLD-AND-MODIFY
Un mode d'affichage qui autorise une sélection de couleurs plus étendue, pouvant aller jusqu'à 4096 couleurs à l'écran en même temps.

INTERLACED MODE [MODE ENTRELACE]
Un mode d'affichage vertical où, dans un affichage de dimensions normales, s'affichent 400 lignes de haut en bas dans un affichage vidéo.

JOYSTICK [MANETTE DE COMMANDE]
Un périphérique du contrôleur qui fait des rotations et des mouvements de gauche à droite à partir de la base du manche; employé pour positionner quelque chose à l'écran.

LIGHT PEN [CRAYON OPTIQUE]
Un périphérique du contrôleur composé d'un crayon et d'une tablette, utilisé pour dessiner quelque chose à l'écran.

LOW RESOLUTION [BASSE RESOLUTION]
Un mode d'affichage horizontal où, dans un affichage de dimensions normales, s'affichent 320 pixels par ligne horizontale.

MANUAL MODE [MODE MANUEL]
Sortie qui ne fait pas appel au DMA. Dans un affichage de sprites, c'est un mode où, à chaque opération, on écrit une ligne individuelle du sprite. En sortie audio, c'est un mode par lequel on écrit un à la fois les mots de données audio en sortie.

MIDI
Une interface normalisée destinée aux instruments de musique utilisée par beaucoup d'instruments.

MICROSECOND (µs) [MICRO-SECONDE]
Un millionième de seconde (1/1000000).

MILLISECOND (ms) [MILLI-SECONDE]
Un millième de seconde (1/1000).

MINTERM
L'une des huit combinaisons logiques possibles de bits de données à partir de sources de données différentes.

MODULO
Un chiffre qui définit quelles sont les données en mémoire qui appartiennent à chaque ligne d'affichage horizontale. Se réfère au nombre d'octets en mémoire entre le dernier mot d'une ligne horizontale et le premier mot de la ligne suivante.

MOUSE [SOURIS]
Un périphérique du contrôleur qu'on peut faire rouler pour déplacer quelque chose à l'écran; il possède aussi des boutons pour servir à d'autres types d'entrée.

MULTITASKING [MULTITACHE]
Un système où plusieurs tâches peuvent être opérationnelles en même temps; aucune tâche n'est forcée de savoir qu'il y a d'autres tâches.

NANOSECOND (ns) [NANO-SECONDE]
Un billionième de seconde (1/1 000 000 000).

NON-INTERLACED MODE [MODE NON ENTRELACE]
Un mode d'affichage vertical où, dans un affichage de dimensions normales, s'affichent 200 lignes de haut en bas lors d'un affichage vidéo.

NTSC
Spécification du signal composite donnée par le National Television Standards Committee [Comité National des Standards de Télévision]. En NTSC, la fréquence de base du quartz de l'Amiga est égale à 28,63636 MHz.

OVERSCAN
Zone balayée par le faisceau vidéo mais non visible à l'écran d'affichage vidéo.

PADDLE CONTROLLER [CONTROLEUR DU MANCHE A BALAI]
Un contrôleur de jeu qui utilise un potentiomètre (résistance variable) pour positionner des objets à l'écran.

PAL
Un standard de télévision européen similaire au (mais incompatible avec le) NTSC. C'est une abréviation de "Phase Alternate Line" [Ligne à Phases Alternées]. En PAL, la fréquence de base du quartz de l'Amiga est égale à 28,37516 MHz.

PARALLEL PORT [PORT PARALLELE]
Un connecteur à l'arrière de l'Amiga employé pour connecter des imprimantes parallèles et d'autres extensions parallèle.

PAULA
L'une des trois puces spécialisées principales de l'Amiga. Contient les circuits pour l'audio, pour l'unité de disquettes et les interruptions.

PITCH [HAUTEUR]
La qualité d'un son exprimée en tant que hauteur ou profondeur.

PIXEL
L'un des plus petits éléments qui créent l'affichage vidéo. Le plus petit élément adressable d'un affichage vidéo.

PLAYFIELD [AIRE DE JEU]
L'un des éléments de base du graphisme de l'Amiga; c'est l'arrière-plan de tous les autres éléments de l'affichage.

PLAYFIELD OBJECT [OBJET AIRE DE JEU]
Sous-section d'une aire de jeu employée dans l'animation des aires de jeu.

PLAYFIELD ANIMATION [ANIMATION DE L'AIRE DE JEU]
Voir BIT-PLANE ANIMATION.

POINTER REGISTER [REGISTRE POINTEUR]
Registre incrémenté sans cesse pour pointer sur une suite d'emplacements en mémoire.

POLARITY [POLARITE]
Etat vrai ou faux d'un bit.

POTENTIOMETER [POTENTIOMETRE]
Un dispositif électrique analogique employé pour régler certaines valeurs variables.

PRIMITIVES
Fonctions de la bibliothèque graphique, de texte et animation de l'Amiga.

QUANTIZATION NOISE [BRUIT DE QUANTIFICATION]
Bruit audio introduit par des erreurs d'arrondi lorsqu'on essaie de reproduire un signal par approximations successives.

RAM [MEMOIRE VIVE]
Mémoire (volatile) à accès aléatoire.

RASTER [TRAME]
La zone en mémoire qui definit totalement l'affichage d'une mappe de bits.

READ-ONLY [EXCLUSIVEMENT DESTINE A LA LECTURE]
Décrit un registre ou une zone de mémoire où l'on peut lire mais où on ne peut écrire.

RESOLUTION
Dans un affichage vidéo, le nombre de pixels qui peuvent s'afficher dans des directions horizontale et verticale.

ROM [MEMOIRE MORTE]
Voir READ-ONLY MEMORY.

SAMPLE [ECHANTILLON]
L'un des segments dans l'axe du temps de la forme d'onde.

SAMPLING RATE [TAUX D'ECHANTILLONNAGE]
Nombre d'échantillons par seconde.

SAMPLING PERIOD [PERIODE D'ECHANTILLONNAGE]
Valeur qui détermine combien de cycles d'horloge sont nécessaires pour jouer un échantillon de données.

SCROLLING [DEROULEMENT]
Déplacement en douceur d'une aire de jeu dans une direction verticale ou horizontale.

SERIAL PORT [PORT SERIE]
Un connecteur à l'arrière de l'Amiga employé pour connecter des modems et d'autres extensions série.

SET [METTRE A 1]
Donner à un bit la valeur 1.

SHARED MEMORY [MEMOIRE PARTAGEE]
La MEMOIRE VIVE employée par l'Amiga pour l'affichage et l'exécution des programmes.

SPRITE [SPRITE]
Objet graphique qu'on peut facilement déplacer, qui est produit par l'un des huit canaux DMA de sprites et qui est indépendant de l'aire de jeu.

STROBE ADDRESS [ADRESSE 'STROBE']
Une adresse qu'on met sur le bus pour permettre à une autre action de s'installer; les données réelles lues ou écrites sont ignorées.

TASK [TACHE]
Module du système d'exploitation ou programme d'application. Chaque tâche semble avoir un contrôle total sur une machine virtuelle possédant un 68000.

TIMBRE
Qualité du ton d'un son.

TRACKBALL [BOULE ROULANTE]
Un périphérique du contrôleur qu'on fait tourner à la main pour déplacer quelque chose à l'écran; il peut avoir des boutons pour servir à d'autres types d'entrées.

TRANSPARENT
Une définition particulière d'un registre de couleur qui permet de voir à travers une couleur d'arrière-plan. Utilisée en mode double aire de jeu.

UART
Le circuit qui contrôle la liaison série vers des périphériques, forme abrégée pour Universal Asynchronous Receiver/Transmitter [Emetteur/Récepteur Universel Asynchrone].

VIDEO PRIORITY [PRIORITE VIDEO]
Définit les objets (aires de jeu et sprites) qui s'afficheront au premier plan et les objets qui s'afficheront à l'arrière-plan. Les objets avec une priorité élevée s'affichent devant les objets de priorité moins élevée.

VIDEO DISPLAY [AFFICHAGE VIDEO]
Tout ce qui s'affiche à l'écran d'un moniteur vidéo ou d'une télévision.

WRITE-ONLY [DESTINE EXCLUSIVEMENT à L'ECRITURE]
Décrit un registre où l'on peut écrire mais que l'on ne peut lire.