```
;*-----------------------------------------------------------------------*
;*                                                                       *
;*                 Programme du nanoreseau du MO5                        *
;*                                                                       *
;*       Programme desassembler et commenté par J. BRIGAUD               *
;*                    Septembre 1999                                     *
;*                                                                       *
;* Ce programme est la propriété de Léanord.                             *
;* Le but du desassemblage du programme est d'adapter le programme du    *
;*  nanoreseau à un cartouche equipée d'un 6850 pour faire du reseau MIDI.*
;*                                                                       *
;*-----------------------------------------------------------------------*
                .title    "Nanoreseau"

                .include  "REG_OS.INC"     ; Registres mémoire utilisés par l'OS du MO5
                .include  "APP_SYS.INC"    ; Appel systeme du MO5
                .include  "MEM_MO5.INC"    ; Organisation mémoire du MO5
                .include  "ADR_NANO.INC"   ; Adresses utilisé par le nanoreseau

; Différentes option de code CON,OVR,REL,ABS,NOPAG,PAG
;                .area     PROGRAM (REL,CON)
                .area     PROGRAM (ABS,OVR,NOPAG)

                .org         0xa000
                .setdp       0xa700


L_C1        =   0x20C1

L_E5        =   0x20E5
L_E7        =   0x20E7
L_E9        =   0x20E9
L_EB        =   0x20EB
L_EC        =   0x20EC
L_ED        =   0x20ED
L_F0        =   0x20F0
L_F5        =   0x20F5
L_F6        =   0x20F6
L_F7        =   0x20F7
L_F8        =   0x20F8
L_F9        =   0x20F9
L_FA        =   0x20FA
L_FB        =   0x20FB

;*-----------------------------------------------------------------------*
;*                       DEBUT DU PROGRAMME PRINCIPAL                    *
;*                             INITIALISATION                            *
;*-----------------------------------------------------------------------*
            .ascii      "REC"           ; 0x52 0x45 0x43
            .db         0x55+0x52+0x45+0x43
.A004:      jmp         BOOT_NANO
            jmp         .A60E           ; Boot du nanoreseau
            jmp         .A6E2           ; Reset a froid du nanoreseau
            jmp         .A20A           ;
            jmp         .A236           ;
            jmp         .A333           ;
            jmp         .A229           ;
            jmp         .A294           ;
            jmp         .A2EF           ;
            jmp         .A352           ;
            jmp         .A1CD           ;
.A025:      jmp         .A67C           ; RESEAU - Point d'entrée principal - Appel
initial
            jmp         .A41B           ; PRLGN  - Prise de la ligne
            jmp         .A69F           ; RELACH - Relâche de la ligne
;*-----------------------------------------------
.A02E:      ldb         #0xB0           ; EMVE - Vas-y émets
            .db         0x8c            ; Codeop de CMPX
.A031:      ldb         #0x80           ; EMVR - Vas-y reçois
            .db         0x8c            ; Codeop de CMPX
.A034:      ldb         #0xc0           ; EMDISC - Déconnecte-toi
            .db         0x81            ; codeop de cmpa        #0x5F
.A037:      clrb                        ; Appel sous attente
.A038:      jmp         .A561           ;
```

```
.A03B:      jmp         .A643
.A03E:      jmp         .A5E7           ; OEE - Ordre d'envoyer l'écran

;* Interuption du nanoreseau ($A041)
;*-------------------------------------------------
INTERUPT:
            ldx     #PIA_JEUX
            ldd     5,x                 ; A=($A7D1) B=($A7D2)
            bpl     .A0A5               ; Bit 7 de A positionné (Reception adresse) (non ->
A0A5)
            cmpb    *NUMPOSTE           ;  oui ->Compare B avec le Numéro de poste
            bne     .A0A1               ; Si c'est différents, ->$A0A1
INT_1:      inca
            beq     .A0A1               ; Si A valait $FF ->$A0A1
            ldb     5,x                 ; B=($A7D1)
            bpl     INT_1               ; On boucle quand que adresse recu
            ldd     6,x                 ; A=($A7D2) B=($A7D3) : A=Code OP et B=Destinataire
            std     *CODE_OP            ; Stock la longueur du message en $2054
            lda     -0x0C,x             ; A=($A7C0)
            anda    #0x01
            pshs    a
            jsr     .A76B               ; Fixe DP =$A7, U=$1F5F, Y=$2052(NUMPOSTE)
            clr     -8,u                ; Met le numero de bloc en attente à 0
            jsr     .A164
            blo     .A099
            cmpa    #0xf0
            ble     .A0A9
            jsr     *(.A740 - .A700)
            stb     -0x0b,u
            clr     0x1ff3
            clr     -3,u                ; Efface $1F5C
            clr     -6,u                ; Efface $1F59
            bsr     .A0F2
            blo     .A090
            bsr     .A038
            blo     .A090
            jsr     .A52A
            ldb     -6,u
            bne     .A090
            bcc     .A08E
            ldb     #0x0f
            stb     -3,u
            jsr     .A41B               ; Prise de la ligne
.A08E:      bsr     .A034               ; Déconnexion
.A090:      tst     0x1FF3
            beq     .A099
            jsr     [0x1ff1]
.A099:      lda     *PIA_SYS
            anda    #0xfe
            ora     ,s+
            sta     *PIA_SYS
.A0A1:      jsr     .A69F
            rti

;*-------------------------------------------------
.A0A5:      jsr     .A68A
            rti

;*-------------------------------------------------
.A0A9:      eora    #0xc0
            bita    #0xf0
            bne     .A099
            jsr     *(.A71F - .A700)        ; Acquittement de la reception
            bra     .A099
.A0B3:      jsr     [ADCHPAG]
            bra     .A0D5
.A0B9:      jsr     *(.A77F - .A700)
            bra     .A0D5

;*-------------------------------------------------*
;*         Analyse de la consigne courante         *
;*-------------------------------------------------*
;* Entrée : U : pointe sur le message ($1F5F)
;* Sortie : A : Type de trame : $F0 : Tout est OK
```

```
;*                              $F4 : La longueur du message est impaire
;*                              $F5 : La longueur du message est de 1
;*                              $F8 : La longueur du message est nul
;* Reg modifié: CCR,X,U,A,B
;*-------------------------------------------------*
.A0BD:      ldx     3,u                     ; Lecture de la longueur des donné dans la
consigne
            ldu     6,u                     ; Lecture de l'adresse destination/source dans la
consigne
            lda     0x1f64                  ; Lecture du numero de la page
            beq     .A0D5                   ; Si page 0 saut en A0D5
            deca
            beq     .A0D5                   ; Si page 1 saut en A0D5
            deca
            beq     .A0B9                   ; Si page 2 saut en A0B9
            deca
            bne     .A0B3                   ; Si page 4 et + saut en A0D5
            ldd     [ADPUTIL]               ; Page 3: appel de ADPUTIL
            leau    d,u                     ; U = U+13
.A0D5:      tfr     x,d                     ; Transfert de la longueur du message dans D
            leax    ,x                      ; Test de la longueur
            beq     .A0E8                   ; Si longueur nulle saut en A0E8
            lda     #0xf0                   ; Initialise le code d'erreur de retour à $F0
            lsrb                            ; b = b/2 (traitement par mot)
            bcc     .A0E7                   ; Si pas carry (Nombre de mot paire) Saut en A0E7
            ora     #04                     ; A = $F4
            leax    -1,x                    ; Longueur - 1
            bne     .A0E7                   ; si different de 0 (longueur >1) saut en A0E7
            inca                            ; A = $F5
.A0E7:      rts                             ; Fin

;*-------------------------------------------------
.A0E8:      lda     #0xf8                   ; A = $F8
            rts                             ; Fin

.A0EB:      ldd     #0x00f8
            pshs    b
            bra     .A123

.A0F2:      ldb     2,y
            lslb
            lslb
            andb    #0x3c
            stb     -1,u
            jsr     *(.A78B - .A700)
            bra     .A10D
.A0FE:      bsr     .A10B
            blo     .A107
            cmpa    -8,u
            beq     .A107
            coma
.A107:      rts

;*-------------------------------------------------
.A108:      lda     #0xf8
            .db     0x8c                ; Codeop de CMPX
.A10B:      bsr     .A0BD

;*-------------------------------------------------
;* Lecture d'une trame sur la liaison série
;*
;* Entrée : U : pointe sur le message
;*          X : Longueur du message (peut etre nulle)
.A10D:      pshs    a                   ; Sauvegarde A
            ldd     #0xc066             ; Mot d'initialistion du 6854
            std     *MC_6854
            stb     1,y                 ; Sauvegarde le mot de commande ($2053)
            asla                        ; Debloc RX (a=$80)
            sta     *MC_6854
            ldd     #0x1010             ; A=Compteur d'erreur; B=Masque (CTS)
.A11C:      deca
            beq     .A178
            bitb    *MC_6854            ; Lecture du status du 6854
            bne     .A11C               ; Si CTS=1, on boucle
```

```
.A123:      ldb     #1
.A125:      inca
            beq     .A178
            bitb    *(MC_6854+1)        ; Attente de la reception du 1er octet de la trame
(adresse)
            beq     .A125
            ldb     *(MC_6854+2)        ; lecture de la donnée
            cmpb    ,y                  ; Est-ce egal à notre numero de poste
            bne     .A178               ; non, on sort
.A132:      inca
            beq     .A178
            ldb     *(MC_6854+1)        ; Attente de l'arrivé de la donnée suivante (Data
Available)
            bpl     .A132
            ldd     *(MC_6854+2)        ; A=CodeOp; B=No du poste source
            cmpb    3,y                 ; Le No source est correcte?
            bne     .A178               ; Non -> on sort
            sta     2,y                 ; Stoc le CodeOp en mémoire ($2054)
            puls    cc                  ; Retrait des flag du type de message
            bmi     .A162               ; Si Flag N positionné, on attendait qu'un entête (on
sort)
            bne     .A14D               ; Si Flag Z Positionné
            lda     *(MC_6854+2)        ;  alors lecture d'un octet en plus
            sta     ,u+                 ; stockage de la donnée
            blo     .A162               ; si donnée négative, on sort
.A14D:      ldb     *(MC_6854+1)
            bmi     .A15A
            lda     #6
.A153:      deca
            beq     .A17A
            ldb     *(MC_6854+1)
            bpl     .A153
.A15A:      ldd     *(MC_6854+2)
            std     ,u++
            leax    -2,x
            bne     .A14D
.A162:      jsr     *(.A76F - .A700)
.A164:      clra
            ldd     #0x0802
.A168:      deca
            beq     .A17A
            bitb    *(MC_6854+1)
            beq     .A168
            clr     *(MC_6854+1)
            lda     *(MC_6854+1)
            bmi     .A17A
            lda     2,y
            rts

;*-------------------------------------------------
.A178:      puls    a
.A17A:      coma
            jmp     *(.A76F - .A700)

;*-------------------------------------------------
;* Envoi d'une trame sur la liaison série
;*
;* Entrée : U : pointe sur le message
;*          X : Longueur du message (peut etre nulle)
;           Y : Pointe sur le destinataire, source, codeop
;           CC: dans la pile Type de trame à emettre
;               N=1     : Emission de l'entete uniquement: SOURCE CODEOP DESTINATAIRE
;               N=0 Z=0 : Emission de l'entete + trame dont la longueur est X (U=$1F60)
;               N=0 Z=1 : Emission de l'entete + 1 octet + trame dont la longueur est X
(U=$1F60) (longueur sur 2octets)
.A17D:      jsr     .A682               ; Lecture du Numero de poste du Nanoreseau (Y=$2052)
            ldd     3,y
            sta     *(MC_6854+2)
            stb     *(MC_6854+2)
            lda     ,y
            sta     *(MC_6854+2)        ; envoi du Numéro de poste
            puls    cc
            bmi     .A1AF
            bne     .A1A5
```

```
            pulu    a
            sta     *(MC_6854+2)
            blo     .A1AF
.A196:      ldb     *(MC_6854)
            bne     .A1A5
            lda     #0x10
.A19C:      ldb     *(MC_6854)
            bne     .A1A5
            deca
            bne     .A19C
            bra     .A1AF
.A1A5:      pulu    a,b
            sta     *(MC_6854+2)
            stb     *(MC_6854+2)
            leax    -2,x
            bne     .A196
.A1AF:      lda     #0x9e
            sta     *(MC_6854+1)
.A1B3:      inca
            beq     .A1BA
            ldb     *MC_6854
            beq     .A1B3
.A1BA:      jmp     *(.A76F-.A700)
;*-------------------------------------------------
.A1BC:      .db     0xff,0xff,0xff,0xff,0xff,0xff
.A1C2:      .ascii  "SCRATCH DOS"
;*-------------------------------------------------
.A1CD:      ldb     *L_F0
            cmpb    #2
            beq     .A1F6
            dec     *L_F0
            bsr     .A236
            blo     .A209
            tstb
            beq     .A1E1
            lbsr    .A333
            blo     .A209
.A1E1:      inc     *L_F0
            bsr     .A236
            blo     .A209
            ldb     #10
            ldx     *L_E7
.A1EB:      lda     b,x
            sta     b,y
            decb
            bge     .A1EB
            bsr     .A229
            blo     .A209
.A1F6:      lda     #2
            sta     *DK_SEC
            ldb     #0x14
            clra
            std     *DK_TRK
            ldd     *L_ED
            std     *DK_BUF
            bsr     .A229
            blo     .A209
            clr     *L_F0
.A209:      rts

;*-------------------------------------------------
.A20A:      ldx     *L_ED
            stx     *DK_BUF
            lda     #2
            bra     .A21E
.A212:      sta     *L_E5
            coma
            rts
.A216:      clra
            rts

;*-------------------------------------------------
.A218:      lda     #3
            ldx     *L_E9
```

```
            stx     *DK_BUF
.A21E:      sta     *DK_SEC
            ldb     #0x14
            clra
            std     *DK_TRK
            lda     #2
            bra     .A22B

;*-------------------------------------------------
.A229:      lda     #8
.A22B:      sta     *DK_OPC
            ldy     *L_E9
            lbsr    .A004
            lda     #3
            rts
;*-------------------------------------------------
.A236:      bsr     .A218
.A238:      blo     .A212
            ldx     #4
            ldy     *L_E9
.A240:      ldu     *L_E7
            ldb     *L_F0
            cmpb    #3
            bne     .A24C
            leau    .A1C2,pcr
.A24C:      clrb
.A24D:      cmpb    #0x0b
            bcc     .A275
            lda     b,y
            cmpa    #0xff
            beq     .A272
            incb
            cmpa    ,u+
            beq     .A24D
            leay    0x20,y
            leax    -1,x
            bne     .A240
            inc     *DK_SEC
            lda     *DK_SEC
            cmpa    #0x10
            bhi     .A272
            lbsr    .A004
            lda     #3
            bra     .A238
.A272:      clrb
            bra     .A290
.A275:      ldb     0x0b,y
            cmpb    *L_EB
            bne     .A272
            ldb     0x0c,y
            cmpb    *L_EC
            bne     .A272
            ldb     *DK_SEC
            lda     0x0d,y
            sta     *L_F6
            clr     *L_F5
            ldx     0x0e,y
            stx     *L_F7
            sty     *L_FA
.A290:      stb     *L_F9
            bra     .A216
;*-------------------------------------------------
.A294:      ldy     *L_ED
            bsr     .A309
.A299:      blo     .A238
            stb     *L_F6
            lbsr    .A218
.A2A0:      blo     .A299
            ldy     *L_E9
            ldx     #4
.A2A8:      ldb     ,y
            beq     .A2CD
            lda     #5
            cmpb    #0xff
```

```
            beq     .A2CD
            leay    0x20,y
            leax    -1,x
            bne     .A2A8
            inc     *DK_SEC
            lda     *DK_SEC
            cmpa    #0x10
            bhi     .A2C8
            lbsr    .A004
            lda     #3
            bra     .A2A0
.A2C8:      lda     #5
.A2CA:      jmp     .A212
.A2CD:      ldx     *L_E7
            ldb     *L_F0
            cmpb    #3
            bne     .A2D9
            leax    .A1C2,pcr
.A2D9:      ldb     #0x0a
.A2DB:      lda     b,x
            sta     b,y
            decb
            bge     .A2DB
            lda     *L_EB
            sta     0x0b,y
            lda     *L_EC
            ldb     *L_F6
            std     0x0c,y
            lbra    .A229
;*-------------------------------------------------
.A2EF:      ldb     *L_F6
            cmpb    #0x28
            bhi     .A303
.A2F5:      tstb
.A2F6:      beq     .A309
            lda     b,y
            cmpa    #0xff
            beq     .A32B
            decb
            cmpb    #0x28
            bls     .A2F5
.A303:      addb    #2
            cmpb    #0x51
            bra     .A2F6
.A309:      clrb
            leay    0x28,y
.A30D:      lda     #5
            cmpb    #0x28
            lbhi    .A212
            lda     b,y
            cmpa    #0xff
            beq     .A326
            negb
            lda     b,y
            cmpa    #0xff
            beq     .A326
            negb
            incb
            bra     .A30D
.A326:      addb    #0x28
            leay    -0x28,y
.A32B:      clr     b,y
            decb
            stb     *L_F9
.A330:      lbra    .A216
;*-------------------------------------------------
.A333:      lda     0x0d,y
            sta     *L_F6
            clr     ,y
            lbsr    .A229
            blo     .A2CA
            ldy     *L_ED
            ldb     *L_F6
.A343:      incb
```

```
            lda     b,y
            clr     b,y
            dec     b,y
            tfr     a,b
            cmpa    #0xc0
            blo     .A343
            bra     .A330
;*-------------------------------------------------
.A352:      ldb     *L_F6
            clra
            lsrb
            std     *L_FB
            inca
            sta     *L_F5
            bcc     .A35F
            lda     #0x09
.A35F:      sta     *L_FA
            rts
;*-------------------------------------------------
.A362:      .db     0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
            .db     0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
            .db     0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
            .db     0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff
            .db     0xff,0xff
;*-------------------------------------------------
.A384:      dec     8,u
            beq     .A3BC
            jsr     *(.A77F - .A700)
            ldx     #0
.A38D:      lda     ,x
            ldb     ,x
            anda    #0x78
            asla
            eora    #0x80
            sta     ,x
            andb    #0x87
            bmi     .A39E
            orb     #8
.A39E:      andb    #0x0f
            orb     ,x
            stb     ,x+
            cmpx    #0x1f40
            bne     .A38D
            jsr     *(.A776 - .A700)
            lda     2,u
            tst     8,u
            bpl     .A3B3
            ora     #4
.A3B3:      lsra
            lsra
            lsra
            blo     .A3BA
            ora     #0x10
.A3BA:      sta     2,u
.A3BC:      ldb     #0xe1
            andb    *PIA_SYS
            orb     2,u
            stb     *PIA_SYS
            rts
;*-------------------------------------------------
.A3C5:      com     2,u
            bpl     .A384
            jsr     *(.A740 - .A700)
            ldb     #0x1e
            andb    *PIA_SYS
            comb
            lda     #0x84
            std     1,u
            bsr     .A3E8
            blo     .A3EB
            ldd     #0x2018
            std     3,u
            clr     6,u
            clr     7,u
```

```
            inc     5,u
            jsr     .A037
            blo     .A3EB
.A3E8:      jmp     .A031               ; Receptions de message
.A3EB:      rts
;*-------------------------------------------------
.A3EC:      sta     -4,u
            clr     -8,u
            jsr     *(.A71F - .A700)    ; Acquittement de la reception
            lda     1,u
            eora    #0x80
            jsr     .A52C
            tst     0x1FF3
            beq     .A402
            jsr     [0x1ff1]
.A402:      jsr     .A69F
            lda     -6,u
            beq     .A417
            puls    x
            lda     -4,u
            cmpa    #0x0f
            beq     .A430
            ora     ,s
            sta     ,s
            puls    cc,dp,x,pc
;*-------------------------------------------------
.A417:      inc     -6,u
            puls    x,pc
;*-------------------------------------------------
;*            Prise de la ligne
.A41B:      ldd     #0x2010             ; A=Nombre de tentative min; B=Masque : CTS
            ora     ITCMPT              ; Masque A avec le compteur d'IT
            anda    #0x3f               ; Limite le temps max à 63
.A423:      bitb    MC_6854             ; Test le bit CTS du 6854 (Presence Horloge)
            beq     .A41B               ; S'il est nulle il faut attendre
            deca                        ; CTS=1; On decremente le compteur d'attente
            bne     .A423               ; Compteur!=0; on attend la fin de la tempo
            orcc    #0x50               ; Blocage des IT FIRQ et IRQ
            jmp     .A749               ; Débloque la Liaison Série
;*-------------------------------------------------
.A430:      jsr     .A69F
.$A433:      puls    cc,dp,x
            swi
            .db     INCHR               ; Lecture du clavier
            cmpb    #3
            bne     .A43F
.A43B:      jmp     [ADCTRLC]
;*-------------------------------------------------

;* Tempo aléatoire en cas d'erreur *
;*---------------------------------*
.A43F:      lda     ITCMPT              ; Lecture du compteur d'interuptions
            tfr     a,b
            anda    #0x1f
            inca
.A447:      subd    #1
            bne     .A447
.A44C:      dec     0x2057              ; Décrémente le compteur d'essai d'emission
            beq     .A43B               ; Si on est à zéro, alors saut a la routine de
traitement des erreurs
            andcc   #0xf0
            pshs    x,dp,cc             ; sauve X,DP,CC
            jsr     .A76B               ; Fixe DP à $A7, U=$1F5F, Y=$2052
            stb     -6,u                ; Range B en $1F59
            ldy     2,s                 ; Recupere la valeur de X mis dans Y
            lda     ,y                  ; Lit la longueur du message
            inca                        ; A=longueur+1
            leau    -1,u                ; u-- (U=$1F5E)
            jsr     *(.A760 - .A700)    ; recopîe du message en $1F5F
            lda     ,u
            ldb     #0xf0
            std     3,y                 ; stock D en $2055
            stb     -0x0b,u             ; stock B en $1F54
            clr     -8,u                ; efface le compteur de sequence courant ($1F57)
```

```
            bsr     .A41B               ; prise de ligne
            tst     -6,u                ; Test $1F56
            beq     .A430
            jsr     .A5C5
            bmi     .A430
            jsr     *(.A79E - .A700)    ; Envoie du message sur la liaison série
            ldb     #0x90               ; Code op attendu en retour
            jsr     *(.A7B0 - .A700)    ; Attente du message venant de la liaison série
            blo     .A430
            bsr     .A4BD               ; Envoie d'un message d'acquittement
.A483:      jsr     *(.A737 - .A700)
            ldd     #0x8010
            sta     *MC_6854
            lda     *L_C1
.A48C:      leax    1,x
            bne     .A494
            eora    #0x01
            sta     *(PIA_SYS+1)
.A494:      bitb    *MC_6854
            bne     .A48C
            jsr     .A0EB               ; Attente d'un message sur la liaison série
            blo     .A483
            tfr     a,b
            lsrb
            lsrb
            lsrb
            andb    #0x0e
            anda    #0x0f
            ldx     #.A4AD
            jsr     [b,x]
            bra     .A483
;*-------------------------------------------------
.A4AD:      .dw     .A501               ; Vas-y reçois
            .dw     .A4BD               ; Réenvoie du message
            .dw     .A516               ; RTS ne rien faire
            .dw     .A4E9               ; Vas-y emet
            .dw     .A3EC               ; Deconnexion
            .dw     .A517               ; Excution d'une tache reseau
            .dw     .A516               ; RTS ne rien faire
            .dw     .A516               ; RTS ne rien faire
;*-------------------------------------------------
.A4BD:      ldd     #0x01a1             ; A contient le No du prochain bloc, B contient le
Code op d'acquitement
            sta     -8,u                ; Rangement du No de bloc attendue
            jmp     *(.A798 - .A700)    ; Envoie du message
;*-------------------------------------------------
.A4C4:      jsr     .A034               ; Deconnexion
            bra     .A483

;* Vérification du Numéro de bloc reçu *
;*-------------------------------------*
.A4C9:      clr     -7,u                ; Efface ???????
            cmpa    -8,u                ; Comparaison au numero de bloc attendue
            beq     .A4D7               ; Si equivalence, OK on sort
            cmpa    -0x0b,u             ; Comparaison au Numero de bloc précédent
            bne     .A4D8               ; S'il n'y a pas d'équivalence erreur
            sta     -8,u                ; Stockage du numero en tant que numero attendue
            inc     -7,u                ; Incrémente????????
.A4D7:      rts
;*-------------------------------------------------
.A4D8:      coma                        ; Erreur : Invertion de A
            rts                         ; Sortie
;*-------------------------------------------------
.A4DA:      bsr     .A4C9               ; Vérification du numéro de bloc recu
            bls     .A4E2               ; Si inférieur ou égal, on sort
.A4DE:      ldd     -0x0a,u             ; Récupération de l'adresse du bloc précédent
            std     6,u                 ; stockage dans la case de l'adresse a remplir
.A4E2:      rts                         ; on sort
;*-------------------------------------------------
.A4E3:      bsr     .A4DE
.A4E5:      ldb     -0x0b,u
            bra     .A4FE

;*-------------------------------------------------
```

```
.A4E9:      bsr     .A4DA               ; Vérification du numero de bloc
            blo     .A516               ; Si négatif, Erreur on sort
            jsr     *(.A7A4 - .A700)
.A4EF:      ldd     6,u                 ; Récupération de l'adresse de chargement
            std     -0x0a,u             ; Stockage de l'adresse du bloc téléchargé
            addd    3,u                 ; Calcul de la nouvelle adresse: Adresse + offset -> D
            std     6,u                 ; Mémorise la nouvelle adresse ou faut stocker les
prochaines informations
.A4F7:      ldb     -8,u                ; Recupére le No du bloc recu
            stb     -0x0b,u             ; Mémorise en tant que No recu
            incb                        ; No de bloc+1
            andb    #0x07               ; Masque sur No (Pas de bloc > 7)
.A4FE:      stb     -8,u                ; Memorise le Numéro de bloc a attendre
            rts                         ; On sort
;*------------------------------------------------
;* Vas-y reçois
.A501:      jsr     *(.A740 - .A700)    ; Initialisation de la liaison serie
            lda     2,y
               ; Lecture du CodeOp
            anda    #0x0f
              ; Garde que le No de bloc
            bsr     .A4DA
              ; Verifie le No de bloc et recupere l'adresse de telechargement
            blo     .A516
              ; Si erreur -> On sort
            jsr     .A0FE
            blo     .A516
            bsr     .A4EF
.A512:      jmp     *(.A71F - .A700)    ; Acquittement de la reception
;*------------------------------------------------
.A514:      clr     1,u
.A516:      rts
;*------------------------------------------------
;* Traitement des tache réseau de 0 à 6
.A517:      jsr     *(.A740 - .A700)    ; Initialisation de la liaison serie
            jsr     .A0F2               ; Reception d'un message de longueur egal à 4*(Codeop
& 0x0f)
            blo     .A514               ; si erreur on sort
            bsr     .A4C9               ; Verification du numero de bloc recu
            blo     .A514               ; si erreur on sort
            bsr     .A4F7               ; incrementation du numero de la sequence
            bsr     .A512               ; envoie d'une réponse OK avec le prochain No de bloc
attendue
            lda     -7,u                ; Lecture du flag indiquant qu'on a recu le numero de
bloc/sequence
            bne     .A516               ; Si on a recu le meme numero de bloc, on sort
(pourquoi???)
.A52A:      lda     1,u                 ; Lecture du code tache réseau
.A52C:      bmi     .A516               ; Si négatif, on sort
            bita    #0xf8               ; test les bits 3-7
            bne     .A548               ; Si != 0 (Code Tache > 7) saut en A548
            asla                        ; Code * 2 (pour indexation dans la table)
            ldx     #.A538              ; X pointe sur la table de fonction
            jmp     [a,x]               ; Saut à la routine concernant le code Tache reseau

;*------------------------------------------------
.A538:      .dw     .A516               ; Rien RTS (permets d'initialiser la consigne)
            .dw     .A4C4               ; Mise en attente
            .dw     .A54C               ; Execution du code recu dans la consigne
            .dw     .A5FA               ; Affichage de la chaine de caractère de la consigne
(6,u)
            .dw     .A3C5               ; Envoi de l'ecran du poste
            .dw     .A031               ; Reception de message
            .dw     .A759               ; Recopie du compte rendue
            .dw     .A548               ;
;*------------------------------------------------
.A548:      jmp     [ADTRCTR]           ; Saut à la procedure de traitement des fonctions > 7
.A54C:      jmp     0x0a,u
.A54E:      jsr     *(.A723 - .A700)
            coma
            puls    b,dp,pc
;*------------------------------------------------
.A553:      bsr     .A4E3
            .db     0x8c                ;   code op pour :  cmpx    #0x8d8d
```

```
.A556:      bsr     .A4E5
.A558:      dec     -5,u
            beq     .A54E
            jsr     .A41B               ; Prise de la ligne
            bra     .A56D
;*-------------------------------------------------
; B contient le code à executer :
;   $B0 : EMVE - Vas-y émets
;   $80 : EMVR - Vas-y reçois
;   $C0 : EMDISC - Déconnecte-toi
;   $00 : Appel sous attente (EMAP)
.A561:      pshs    dp,b
            jsr     .A740
            jsr     .A76B
            lda     #6
            sta     -5,u
.A56D:      ldb     ,s
            beq     .A587
            bpl     .A5B8
            cmpb    #0xb0
            beq     .A59A
            bmi     .A5A7
            orb     -3,u
            clr     -8,u
            jsr     *(.A798 - .A700)
            exg     a,b
            jsr     *(.A7AE - .A700)
.A583:      blo     .A558
            puls    b,dp,pc
;*-------------------------------------------------
;* Appel sous attente (EMAP)
;*-------------------------------------------------
.A587:      ldb     #0xd0
            stb     4,y
            bsr     .A5C5
            bmi     .A558
            jsr     *(.A79E - .A700)
            jsr     .A4F7
            jsr     *(.A7AE - .A700)
            blo     .A556
            puls    b,dp,pc
;*-------------------------------------------------
.A59A:      jsr     *(.A796 - .A700)                ; Envoie d'un acquittement
            jsr     .A0FE
            blo     .A558
            jsr     .A4EF
            clra
            puls    b,dp,pc
;*-------------------------------------------------
.A5A7:      jsr     *(.A796 - .A700)
            bsr     .A5D3
            bmi     .A558
            jsr     *(.A7A4 - .A700)
            jsr     .A4EF
            jsr     *(.A7AE - .A700)
            blo     .A553
            puls    b,dp,pc
;*-------------------------------------------------
.A5B8:      ldb     #0x90
            jsr     *(.A798 - .A700)
            ldd     #0x01a0
            sta     -8,u
            jsr     *(.A7B0 - .A700)
            bra     .A583

.A5C5:      ldb     -1,u
            addb    #3
            andb    #0x3c
            stb     -1,u
            lsrb
            lsrb
            orb     4,y
            jsr     *(.A798 - .A700)
.A5D3:      jsr     *(.A737 - .A700)
```

```
            ldb     #0x0f
.A5D7:      decb
            bmi     .A5E6
            bita    *MC_6854
            bne     .A5D7
            bita    *MC_6854
            bne     .A5D7
            bita    *MC_6854
            bne     .A5D7
.A5E6:      rts
;*-------------------------------------------------
; OEE - Ordre d'envoyer l'écran
.A5E7:      bsr     .A600
            jsr     .A6B7
            ldb     #4
            std     1,u
            ldd     #0x1f40
            std     4,u
            inc     6,u
            jmp     .A679
;*-------------------------------------------------
.A5FA:      ldx     6,u
            bra     .A605

;* Affichage du caractère contenu dans B *
;*---------------------------------------*
.A5FE:      swi
            .db     OUTCHR              ; Affiche le contenu de B a l'écran
.A600:      jmp     .A776               ; Met la memoire ecran Forme

;* Affichage d'une chaine de caractère pointé par X *
;*--------------------------------------------------*
.A603:      swi
            .db     OUTCHR              ; Affiche le contenu de B a l'écran
.A605:      bsr     .A600               ; Met la memoire ecran Forme
            ldb     ,x+                 ; Charge le caractere pointé par B
            cmpb    #4                  ; Est-ce la fin de la chaine de caractère
            bne     .A603               ;  Non->$ affichage du caractère
            rts                         ; Fin de la procedure

;*-------------------------------------------------------*
;*                                                       *
;*                   BOOT du Nanoreseau                  *
;*                                                       *
;*-------------------------------------------------------*
.A60E:      ldx     #0x9c40             ; X=40000 (40000 tentatives de connexion au nanoreseau
.A611:      jsr     .A723               ; Initialisation du 6854
            beq     .A621               ; CTS positionner, Reseau OK, sinon on boucle
            leax    -1,x                ; Decremente le compteur d'essai
            bne     .A611               ; Si ce n'est pas la fin des test, on reessai.
.A61A:      clr     DKFLG               ; DKFLG=0 (Plus de cartouche detecté)
            jmp     [0xeffe]            ; Saut au basic du MO5
.A621:      ldx     #0x1f50             ; X=$1F50
            bsr     .A600               ; Met la memoire ecran Forme
.A626:      clr     ,x+                 ; Effacement du buffer de consigne (1F50-1FF8)
            cmpx    #ADCHPAG
            bne     .A626
            lda     #1
            sta     -2,x                ; 1 mis à l'@ $1FF6 : Type d'ordinnateur 1=MO5
            ldd     #.A5E6              ; $A5E6 mis à l'@ $1FF8, $1FFA, $1FFC : RTS
.A634:      std     ,x++
            cmpx    #ADCTRLC
            bne     .A634
            ldd     #.A61A
            std     ,x                  ; $A61A mis à l'@ $1FFE : Annulation de la cartouche
Nanoreseau passage en basic
            clr     0x2058              ; Effacement de l'adresse $2058
.A643:      ldx     #INTERUPT           ; Adresse de la routine d'interruption.
            stx     0x2064              ; Initialisation du vecteur d'interruption ($2064 =
$A041)
            jsr     .A76F               ; Initialise U=$1F5F, Y=$2052 et met la memoire ecran
Forme
            ldx     #.A6C5              ; X pointe sur une chaine de caractère
            bsr     .A605               ; Affichage de la chaine de caractère
```

```
            bsr     .A682               ; Lecture du No de Poste designé A contient le Numéro
de poste
            ldb     #0x2f
.A655:      incb                        ; B++
            suba    #10                 ; A = No de poste - 10
            bpl     .A655               ; Si No A>$= 10 aller en $A655
            cmpb    #0x30               ; Si No de poste < 10 ne rien faire
            beq     .A660
            bsr     .A5FE               ;  Affichage de B à l'écran
.A660:      tfr     a,b
            addb    #0x3a
            bsr     .A5FE               ;  Affichage de B à l'écran
            bsr     .A605               ; Affichage d'une chaine de caractères ($0D + $0A)
            ldb     #0x3c               ; Longueur de buffer à vider
            bsr     .A6B9               ; Vide la zone $1FA0 + marque la longueur($3C) de la
chaine au début ($AFE0)
            leau    0x0b,u              ; u+=$0b -->$ U=$1FAB
            ldy     #0xefe0
            lda     #0x10
            jsr     .A762               ; Recopie la chaine pointé par Y dans le Buffer
pointé par U, LNG=A(*2) (X modifié)
            bsr     .A68A               ; U=$1F5F, Y=$2052
.A679:      ldx     #0x1fa0
.A67C:      clr     0x2057
            jmp     .A44C

;*
;* Signature en $EFE0 : 41 00 FF 20 3D 4C 01 60 20 3C 4F 01 05 20 3F 9C
;*              $EFF0 : 19 25 03 11 93 15 10 25 32 8A 7E FF E5 FD E9 35
;*
;* Lecture du Numero de poste du Nanoreseau
;*-------------------------------------------------------
.A682:      lda     SWITCH              ; Lecture des switch indiquant le numero de poste NR
            anda    #0x1f               ; 31 postes max
            sta     ,y                  ; Memorisation a l'emplacement pointer par Y
            rts
;*-------------------------------------------------------
.A68A:      ldu     #MC_6854            ; U pointe sur le MC6854
            ldd     #0xc100             ; A=$C1 et B=00
            std     ,u                  ; Ecriture dans le MC6854
            lda     #0x1e
            sta     3,u
.A696:      jsr     .A723               ; Reset du MC6854
            beq     .A6AD
            bita    #0x10
            bne     .A696

;* RELACH - Relâche de la ligne
.A69F:      jsr     .A723
            ldd     #0xc180
            jsr     .A752               ; Stocke D en $A7D0 et B en $2053
            deca                        ; A = $C0
            sta     MC_6854
            rts
;*-------------------------------------------------------
.A6AD:      lda     0x2058
            bne     .A6B6
            lda     #0x82
            sta     ,u
.A6B6:      rts
;*-------------------------------------------------------
.A6B7:      ldb     #0x1c
.A6B9:      ldu     #0x1FA0
            stb     ,u
            incb
.A6BF:      clr     b,u
            decb
            bne     .A6BF
            rts

;* Définition de chaine de caractere
;*-------------------------------------------------------
.A6C5:      .db     0x0d
            .ascii  "NANORESEAU LD USTL V3  p"
```

```
            .db     4
CR_LF:      .db     0x0a,0x0d,4

;*-------------------------------------------------------
.A6E2:      clr     DK_STA              ; DK.STA etat courant du controleur
            lda     #0x0a
            sta     DK_OPC              ; DK.OPC mot de commande du controleur

            .org    0xA6EA
;* Zone précéder d'un ORG pour etre sur d'être dans la PAGE A7   *
;* pour les sauts court en page direct.                          *
;*                                                               *
;*---------------------------------------------------------------*

;* Point d'entré du Nanoreseau ($a6ea)
;*----------------------------
BOOT_NANO:
            pshs    u,y,x,b,a
            ldd     #INTERUPT           ; D=$A041 (Adresse d'interruption du Nanoreseau)
            subd    IRQPT               ; Retrait de l'adresse d'IRQ
            beq     BOOT_1              ; Si le resultat = 0 ->$ init déja faite, saut en
$A6F7
            jsr     .A60E               ;  Initialisation de la cartouche Nanoreseau.
BOOT_1:     jsr     .A76F               ; Initialise U=$1F5F, Y=$2052 et mise en mémoire Forme
            ldb     #0x10
            bsr     .A6B9               ; Vide la zone $1FA0 + marque la longueur de la
chaine au début ($AFE0)
            lda     #8
.A700:      ldb     -10,y
            cmpb    #8
            bne     BOOT_2
            ora     #0x40
BOOT_2:
            std     2,u
            ldd     -3,y
            std     7,u
            ldd     #0x0380
            stb     5,u
            leay    -10,y
            leau    10,u                ; U=$1FAA
            jsr     .A762               ; Recopie la chaine pointé par Y dans le Buffer
pointé par U, LNG=A(*2) (X modifié)
            jsr     .A679
            puls    a,b,x,y,u,pc

;*-------------------------------------------------*
;*    Acquittement de la reception d'une trame     *
;*-------------------------------------------------*
;* Entrée : Rien                                   *
;* Sortie : B : Contient le code op d'acquittement *
;* Reg modifié: CCR,B                              *
;*-------------------------------------------------*
.A71F:      ldb     #0xe0
            bsr     .A796

;*-------------------------------------------------------
;* Initialisation du 6854
;* L'horloge est mise en entrée, et blocage de la ligne TxD : On ne peut plus emettre.
;* $a723
INIT_6854:
.A723:      ldd     #0xc066             ; A=$C0 et B=$66 : A=CR1=Tx/Rx reset + Select CR0
            std     MC_6854                                B:RTS=0; x/Rx status clear; Flag
Idle+2byte transfert
            stb     CODE_INIT           ; $2053 contient le code op envoyé au 6854
            ldd     MC_6854
            cmpx    MC_6854+2
            cmpd    #0x1000
            rts
;*-------------------------------------------------------
.A737:      bsr     INIT_6854
            bne     .A737
            rts

;*-------------------------------------------------------
```

```
.A73C:      ldb     -8,u                ; Charge le No de bloc en cours
.A73E:      stb     4,y
.A740:      ldb     CODE_INIT
            bmi     .A749
.A745:      bsr     INIT_6854
            bne     .A745
.A749:      ldd     #0xc100             ; Tx/Rx reset + Select CR3
            std     MC_6854             ; CR3=00==> bit DTR = 0 donc signal DTR = 1
            ldd     #0x40e6             ; RX_Frame discontinue + Select CR2
.A752:      std     MC_6854             ; RTS=1; Tx/Rx status clear; Flag Idle+2byte transfert
            stb     CODE_INIT
            rts

;*-----------------------------------------------------
.A759:      leay    10,u                ; Y pointe sur la zone DATA de la consigne
            ldu     ADCRDU              ; Lecture de l'adresse du compte rendue
            lda     ,u+                 ; Lecture de la longueur du CR
.A760:      inca                        ; Longueur +1
            lsra                        ; On divise par 2 (traitement par mot de 16 bits)
.A762:      ldx     ,y++                ; Lecture des informations
            stx     ,u++                ; Recopie des infos en mémoire
            deca                        ; compteur - 1
            bne     .A762               ; On boucle tant que le compteur est != 0
            bra     .A76F               ; Réinitialistions des pointeur U, Y, Memoire forme

.A76B:      ldb     #0xa7
            tfr     b,dp

;*-----------------------------------------------------*
;*            Initialisation du contexte               *
;*-----------------------------------------------------*
;* Entrée : Rien
;* Sortie : U : Pointe sur le buffer de la consigne
;*          Y : Pointe sur le numéro du poste
;*          Mémoire écran positionné sur la mémoire Forme
;*
;* Reg modifié: CCR,Y,U,B
;*-----------------------------------------------------*
.A76F:      ldu     #0x1f5f             ; Initialise U à l'adresse $1F5F: Message reseau
            ldy     #NUMPOSTE           ; Y=$2052 Adresse contenant le numéro du poste
.A776:      ldb     PIA_SYS             ; Lecture du Port B du PIA Systeme
            orb     #1                  ; Force la mémoire Ecran Forme
.A77B:      stb     PIA_SYS             ; Positionne la mémoire ECRAN en mode texte
            rts                         ; On sort

;*-----------------------------------------------------*
;*          Passage en mémoire ecran Texte             *
;*-----------------------------------------------------*
;* Entrée : Rien
;* Sortie : Rien
;*          Mémoire écran positionné sur la mémoire Texte
;*
;* Reg modifié: CCR,B
;*-----------------------------------------------------*
.A77F:      ldb     PIA_SYS             ; Lecture du Port B du PIA Systeme
            andb    #0xfe               ; Force la mémoire Ecran Texte
            bra     .A77B

.A786:      ldd     TYPORD              ; A=Type d'ordinateur; B=Type d'application
            std     8,u                 ; Rangement info dans la consigne ($1F67, $1F68)
.A78B:      lda     #0xf0               ; Code pour dire qu'il y a 3+N octets à envoyer
            ldx     -2,u                ; Charge X avec la longueur de la consigne
            leau    1,u                 ; U++ ($1F60)
            bne     .A795               ; Si Différent de 0, Ok c'est bon
            lda     #0xf8               ; Code pour dire qu'il n'y a que 3 octets à envoyer
.A795:      rts

;*-----------------------------------------------------
.A796:      orb     -8,u                ; Ajout dans les poids Faible de B, le No de bloc
suivant attendu
.A798:      bsr     .A73E               ; Initialise la liaison série
            lda     #0xf8               ; Code pour dire qu'il n'y a que 3 octets à envoyer
            bra     .A7A9               ; Envoie de la réponse sur la liaison série.
```

```
.A79E:      bsr     .A73C               ; Init la LS
            bsr     .A786               ; Initialise la consigne, le pointeur de message et
le type de message à envoyer
            bra     .A7A9               ; Envoi de la consigne

.A7A4:      bsr     .A73C               ; Init la LS
            jsr     .A0BD               ; Analyse la trame
.A7A9:      pshs    a                   ; Sauvegarde du type de trame à emmetre
            jmp     .A17D               ; envoie de la trame sur la liaison serie

.A7AE:      ldb     #0xe0
.A7B0:      pshs    b
            jsr     .A108
            eora    ,s+
            blo     .A7BE
            cmpa    -8,u
            beq     .A7BE
            coma
.A7BE:      rts
            rts                         ; Derniere adresse de la cartouche Nanoreseau ($A7BF)
```