Transmission de données 100ko/min pour Sharp MZ

Nous avons besoin de 3 signaux sur le Sharp MZ :

- WDATA	Adresse $E002	Bit 1	PC1	Bit en Sortie
- MOTOR	Adresse $E002	Bit 4	PC4	Bit en Entrée
- RDATA	Adresse $E002	Bit 5	PC5	Bit en Entrée

Initialisation : On positionne WDATA=0, MOTOR=0 et RDATA=0 (Point A sur le graphe).

Séquence de transmission rapide :

1 - Le Sharp MZ demande le positionnement d'une donnée : WDATA passe de 0 a 1 (B).

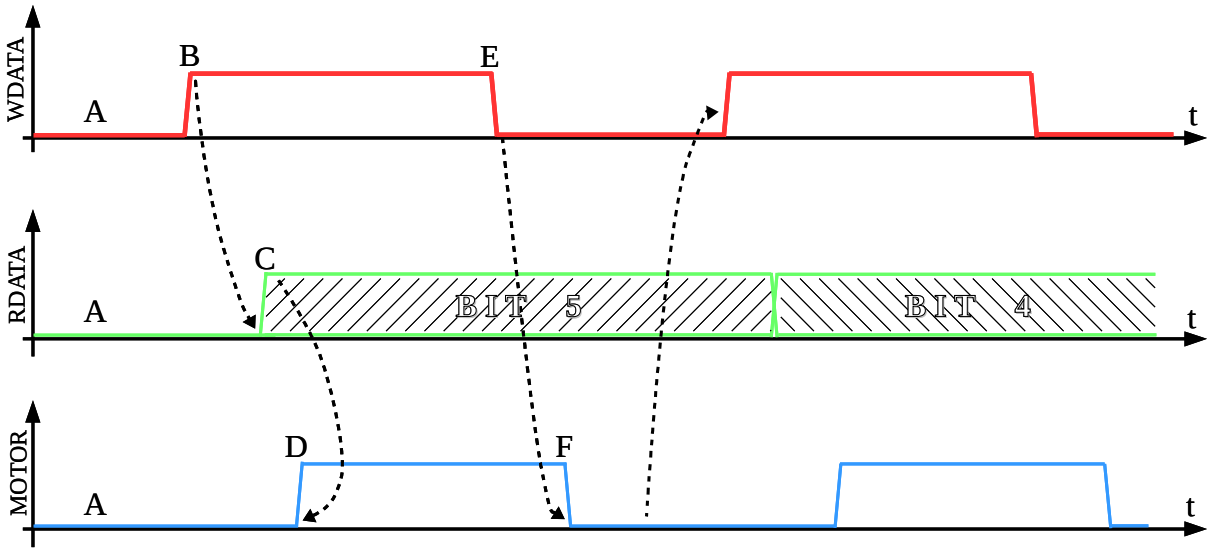
2 - L'arduino attend ce signal et positionne un bit a 0 ou 1 en utilisant la ligne RDATA (C).

3 - L'arduino informe le Sharp MZ que le bit est positionné et prêt à lire : MOTOR passe de 0 a 1 (D).

4 - Le Sharp MZ lit le bit contenu dans RDATA, puis informe de la fin de lecture en positionnant WDATA a 0 (E).

5 - L'arduino accuse réception de ce dernier signal en positionnant MOTOR a 0 (F).

Le cycle peut recommencer pour un nouveau bit.



Afin d'avoir un taux de transfert le plus grand possible, j'ai utiliser les méthodes suivantes :

- Le transfert des bits se fait dans l'ordre 5 puis 4 puis 3 puis 2 puis 1 puis 0 puis 7 puis 6. Le Sharp MZ, à chaque réception, effectue une rotation de bit avec l'instruction RLCA, ce qui fait revenir le premier bit reçu sur la position 5 de l'octet (correspondant a PC5 de la ligne RDATA). Donc nous n'avons pas besoin de faire d'autres rotation de bit afin de récupérer l'octet transmis.

- Le programme de transfert se situe dans les commentaires du programme MZF appelé en mode conventionnel par le moniteur Sharp MZ car il ne fait que 53 octets, et le chargement initial est fortement accéléré :

  Sur Sharp MZ-700, on a pour l'entête :
  2s d'attente : 2000000
  100 bits de GAP : 100*504
  40 bits a 1 : 40*958
  40 bits a 0 : 40*504
  2 bits a 1 : 2*958
  Entête : Maximum 80 octets de $FF avec 1 bit de synchronisation : 9*80*958
  CheckSum : Maximum 2 octets de $FF avec 1 bit de synchronisation : 2*9*958
  1 bit de fin : 1*958

  soit 2818758 µs donc 2,82s max

  Après chargement, le programme se trouve en $1108, dans les commentaires donc.

  Pour que le programme puisse s'exécuter automatiquement par le moniteur, on est obligé de charger 3 octets en mémoire, de mnémonique, JP $1108 soit C3 08 11 car dans le moniteur 1Z-013A on a :

```
0126   2A 06 11      LD HL,(EXADR)
0129   7C            LD A,H
012A   FE 12         CP 12H
012C   38 E1         JR C,LOAD-2
```

  Tout programme dont l'adresse d'exécution est inférieure a 12xxH ne s'exécutera pas.

  Sur Sharp MZ-700, on a pour le programme :

  2s d'attente : 2000000
  100 bits de GAP : 100*504=50400
  20 bits a 1 : 20*958=19160
  20 bits a 0 : 20*504=10080
  2 bits a 1 : 2*958=1916
  3 octets (C3 08 11 soit 11000011 1 00001000 1 00010001 1) : (7+3)*958+17*504=18148
  CheckSum (01110000 1 00000000 1) : (3+2)*958+13*504=11342
  1 bit de fin : 1*958=958

  soit 2112004 µs donc 2,11 s

  Total pour transférer le petit programme et l'exécuter : 2818758+ 2112004=4930762 µs donc 4,93 s

  Ensuite on passe en mode transfert rapide.

Exemple de taux de transfert :
Programmes EUGEA ou SIDEROLL-F de 44544 octets
Transfert en 20,64 s
Soit au total 20,64+4,93=25,6 s
Sur 60-4,93=55,07 s on est sur un transfert rapide soit 118848 octets par min réel.

ANNEXES

Sharp MZ : Programme stocké en $1108

```
ORG $1108

01 SIZE_H SIZE_L        LD BC, SIZE
21 ADR_H  ADR_L         LD HL, ADRESS

3E 02                   LD A, $02           INIT
32 02 E0                LD ($E002), A       DWRITE = 0
            Boucle1:
E5                      PUSH HL
21 02 E0                LD HL, $E002
11 08 00                LD DE, $0008

            Boucle2:
7E                      LD A, (HL)          WAIT /MOTOR = 0
E6 10                   AND $10
20 FB                   JR NZ, Boucle2

AF                      XOR A
77                      LD (HL), A          DWRITE =  1

            Boucle3:
7E                      LD A, (HL)          WAIT /MOTOR = 1
E6 10                   AND $10
28 FB                   JR Z, Boucle3

7E                      LD A, (HL)          READ BIT
E6 20                   AND $20
B2                      OR D
07                      RLCA
57                      LD D, A             AND STORE

3E 02                   LD A, $02           DWRITE = 0
77                      LD (HL), A

1D                      DEC E
20 E8                   JR NZ, Boucle2

E1                      POP HL
72                      LD (HL), D          STORE DATA TO RAM

0B                      DEC BC              SIZE--
79                      LD A, C             TEST IF BC=0
B0                      OR B
23                      INC HL              ADRESS++

20 D9                   JR NZ, Boucle1      NOT END ? REPEAT !

C3 EX_H  EX_L           JP EXECUTION        PROGRAM EXECUTION
```

Arduino 2560 + PCB + RepRapDiscount Full Graphic Smart Controller

Extrait du programme

```
for (i = 0 ; i < mzf_taille ; i++)
 {
  donnee = (uint8_t)(entree.read () & 0xFF) ;
  for (j = 0 ; j < 8 ; j++)
   {
    // Attente demande donnee
    while (digitalRead (MZ_CASSETTE_WRITE) == 0) { }

    // Positionne le bit
    digitalWrite (MZ_CASSETTE_READ, ((donnee & ordre [j]) != 0) ? LOW : HIGH) ;

    // Informe le positionnement du bit
    digitalWrite (MZ_CASSETTE_SENSE, LOW) ; // signal /SENSE a 0 (Lecteur disponible SENSE=1)

    // Attente de la lecture de la donnee
    while (digitalRead (MZ_CASSETTE_WRITE) == 1) { }

    // Reinitialisation
    digitalWrite (MZ_CASSETTE_SENSE, HIGH) ; // signal /SENSE a 1 (Lecteur non disponible SENSE=0)
   }
 }
```