

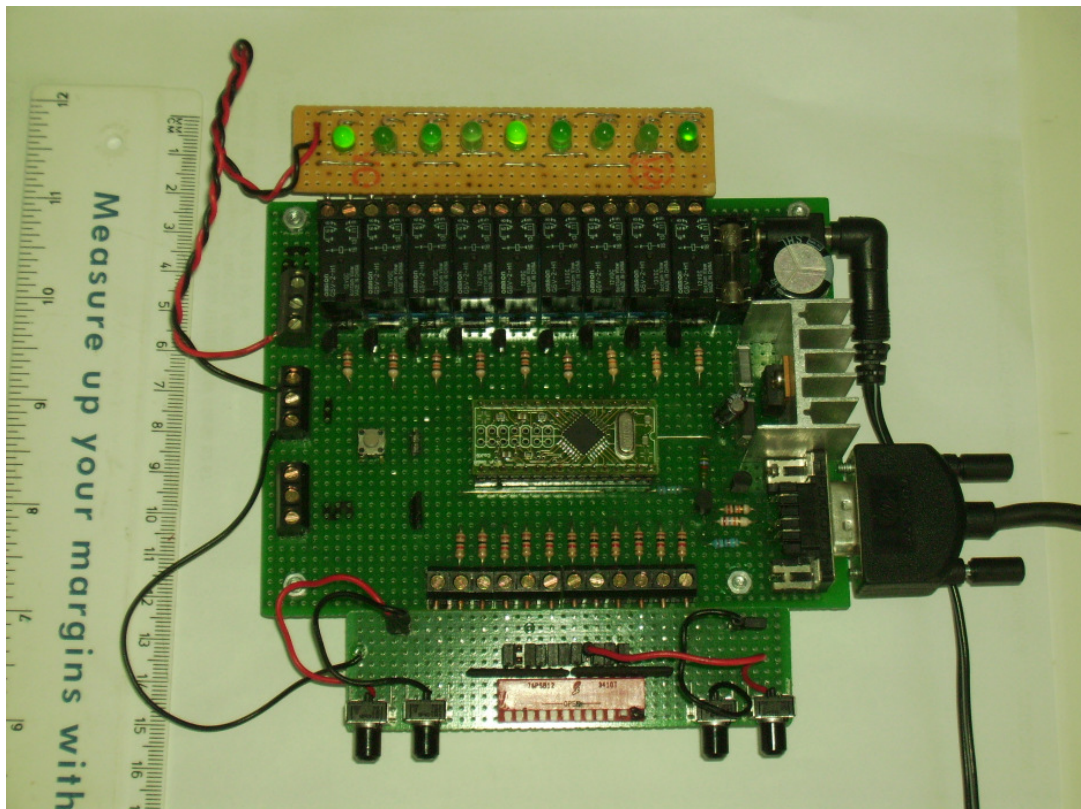
MicroPLC.

De Gérard Jacquemin

1) Description

Le montage réalisé est en fait un micro automate programmable à base de R8C13 avec les caractéristiques suivantes :

- 11 entrées digitales, de i1 à i11.
- Dont 6 sont également des entrées analogiques, ad1 (i1) à ad6 (i6).
- 9 sorties à relais, de o1 à o9. Vous pouvez également créer vos propres modules dip16 avec des leds, octocoupleurs.....
- 32 mémoires internes de m1 à m32.
- 16 temporisations de 0 à 6400 secondes, de t1 à t16.
- Une interface rs232 pour la programmation **et le contrôle on-line des valeurs** de l'automate (valeurs des entrées digitales et analogiques, des sorties, des mémoires, des temporisations à 38400 baud).
- Des valeurs logiques rs232, de a à h et de p à s, qui permettent d'agir sur le programme logique par le biais d'un programme (basic, web). En envoyant un caractère 'A' par l'interface rs232 vous mettez a à '1' et en envoyant un caractère 'a' il devient '0'.
- Des signaux de 1hz (h1), 2hz (h2), 5hz (h5) et de 2 sec m2.
- Le programme logique peut renvoyer du texte sur le port rs232



2) Introduction à la Programmation

Pour le langage de programmation après de longues réflexions, je me suis dit qu'un interpréteur intégré dans le micro contrôleur serait lent et prendrait trop de place. Que faire ?

En regardant mon écran (j'étais dans l'interface de programmation en C), je me suis dit : pourquoi ne pas utiliser le C pour écrire les équations logiques. En effet, le compilateur va alors compiler les équations logiques pour moi.

De plus elles seront en natif dans le cpu et ne seront pas interprétées, mais directement compilées en langage machine et donc plus rapides. Dans ce cas il me sera alors possible d'introduire du code C dans mon programme logique, ce qui me donne une plus grande flexibilité dans la programmation d'automates.

Un exemple:

```
o2= ( i1 or i2 ) and ( i3 or i4) ;
```

Et en utilisant les symboles (c'est plus joli) en C on arrive à :

```
// *** Put your label here to use in your logic program *****

#define chambre1 i1
#define chambre2 i2
#define chambre3 i3
#define chambre4 i4
#define lampe o2

void plc (void)          // Main logic program
*****
{
// *** Put you logic code here. *****

lampe= (chambre1 or chambre2) and (chambre3 or chambre 4) ;

}
//*****
```

Ensuite on compile et on flashe le micro contrôleur et c'est fait, notre automate fonctionne. Il ne faut surtout pas retirer le câble rs232 car on peut l'utiliser pour contrôler le bon fonctionnement du programme, ou même interagir dessus (voir plus loin).

Pour résumer :

- On modifie le programme muplc.c dans le programme de développement C High-performance Embedded Workshop de Renesas Technology Corp. .
- Dans les zones timer on définit la valeur des temporisations en seconde.
- Dans la zone label on associe des labels aux adresses.

- Dans la zone plc on introduit le code logique et éventuellement le code c (! pas de boucle), il faut bien sûr respecter la syntaxe C. Ne modifiez pas le fichier muplc_lib.c si vous ne savez pas ce que vous faites.

- Ensuite on compile sans erreur :

Build Finished

0 Errors, 0 Warnings

- Puis on flashe le code dans le cpu avec FDT Basic.

Ne pas oublier de mettre le jumper de mode et de reseter le montage avec encore un reset après programmation.

- On peut contrôler le bon fonctionnement de l'automate par un programme d'émulation de terminal : connecter sur le port rs232 du montage, c'est le même port que celui de la programmation (38400 baud, 8bit, 1 stop bit, pas de parité) .

Il ne faut pas oublier de se déconnecter du programme d'émulation avant d'essayer de programmer le microcontrôleur, car FDT Basic ne pourra pas ouvrir le port rs232 car il est alors utilisé par le programme d'émulation.

3) Détail de la programmation

3.1) Label

Sous la forme :

```
// *** Put your label here to use in your logic program *****
```

```
#define chambre1 i4
```

```
#define chambre2 i5
```

```
#define chambre3 i6
```

```
#define lampe_chambre o4
```

Facilite la lecture de l'équation logique, et permet une modification centralisée dans le cas de modification d'entrées. Par exemple si on veut que chambre1 devienne l'entrée i2.

3.2) Les entrées

Vous disposez des entrées logiques de i1 à i11, 0v sur l'entrée donne un valeur 0, et 5v donne une valeur 1 (! pas de 12v sur les entrées).

o4= (i2 or i3)xor m2;

Vous pouvez voir l'état des entrées on-line par le port rs232 (38400 baud, 8bit, no parité) utilisé pour programmer le microcontrôleur, en envoyant un caractère 'i'

>Input value

i11-i1: 000 00100010

Vous avez aussi 6 entrées analogiques ad1 (i1) à ad6 (i6), 0v donne un 0, et 5v donne une valeur de 1023, c'est une conversion sur 10 bits. Pour voir les valeurs on-line, tapez 'I'

>Analog to digital value

AD1-AD6: 00001 01023 00000 00832 00000 00000

Pour les utiliser, on peut recourir au C qui donne une plus grande flexibilité de calcul que les langages logiques classiques, exemple :

```
o5= 0;                // Use of ad4 value
if (ad4 > 200) o5= s2 ;    // If ad4 from 200 to 400 o5= 0.5 hz
if (ad4 > 400) o5= s1 ;    // If ad4 from 400 to 600 o5= 1 hz
if (ad4 > 600) o5= h2 ;    // If ad4 from 600 to 800 o5= 2 hz
if (ad4 > 800) o5= h5 ;    // If ad4 from 800 to 1000 o5= 5 hz
if (ad4 > 1000) o5= 1 ;    // If ad4 >100 then o5= 1
```

3.3) Les sorties

Vous disposez de 9 sorties logiques o1 à o9, un 0 donne un contact ouvert et un 1 un contact fermé au relais correspondant.

Pour voir les valeurs on-line, tapez ‘o’

```
>Output value
o9-o1: 1 00000000
```

3.4) Les mémoires

Il y a également des mémoires logiques de m1 à m32, que vous pouvez utiliser pour mémoriser des états logiques.

Tapez ‘m’ pour voir les valeurs online :

```
>Memo value
m8-m1 : 00001000   m16-m9 : 00000000
m24-m17: 00000000   m32-m18: 00000000
```

3.5) Les équations logiques

Sous la forme :

```
o1= i2 or i3 ;
lampe_chambre= (chambre1 or chambre2) and chambre3 ;
```

Vous pouvez utiliser not, and, or, xor dans toutes les combinaisons que vous voulez, avec des parenthèses. Ne pas oublier le ‘;’ à la fin, sinon le compilateur C va faire une erreur.

3.6) Les timers

Vous disposez de 16 timers de 0 à 6400 secondes, vous devez préciser dans la zone timer la valeur de chaque temporisation.

```
void init_tempo (void) // Init tempo *****
{
// *** Set the timer value in sec here ( ! max 6400 sec ) *****
```

```

    val_tempo[1]=5;
    val_tempo[2]=10;
    val_tempo[3]=15;
    val_tempo[4]=20;
    val_tempo[5]=15;
    val_tempo[6]=30;
    val_tempo[7]=35;
    val_tempo[8]=10;
    val_tempo[9]=10;
    val_tempo[10]=10;
    val_tempo[11]=10;
    val_tempo[12]=10;
    val_tempo[13]=10;
    val_tempo[14]=10;
    val_tempo[15]=10;
    val_tempo[16]=10;
}

```

Par exemple la valeur de défaut pour le timer 1 est de 5 sec.

Pour l'utiliser, il suffit de donner une valeur à l'entrée du timer (tix), puis utiliser la sortie du timer, exemple :

```

ti1= switch1;          // Set the input timer1
lampe1= t1;            // Activate the lamp1 (o1) by t1

```

Si switch1 est à 1 et reste à 1, le compteur reste à 5 secondes, ce n'est que lorsque switch1 passe à 0 que le décomptage commence.

Vous pouvez voir les valeurs on-line des temporisations par l'interface rs232 en envoyant un caractère 't' pour voir la valeur des timers en 1/10 de seconde.

```

>Timer value ( 1/10 sec )
t1-t8 : 00000 00016 00112 00000 00000 00000 00000 00000
t9-t16: 00000 00000 00000 00000 00000 00000 00000 00000

```

Et un caractère 'T' pour voir les valeurs logiques de sortie de chaque timer :

```

>Timer value
t8-t1 : 00000000  t16-t9: 00000000

```

Si on veut utiliser le flanc montant, il faut utiliser une mémoire temporaire :

```

ti2= switch2 and m2 ;      // Use a timer2 in positive edge (0 to 1)
m2 = not switch2;         // Memorise the switch2 state
lampe2= t2;               // set output

```

Explication :

```

Switch2  ____-____-
M2        ----_
Ti2       ____-____

```

Ce n'est qu'au passage de switch2 de 0 à 1 que le timer sera activé, c'est de la logique.
Il ne faut pas oublier que le programme est exécuté du haut vers le bas.

Et pour un flanc descendant :

```
ti3= not switch3 and m3 ;      // Use a time3 in negative edge (1 to 0)
m3 = switch3 ;                // Memorise the sitch3 state
lampe3= t3 ;
```

On inverse la mémoire m3 et le tour est joué.

3.7) Les signaux clignotants.

Vous disposez également de signaux clignotants:

De 0,5 hz c'est le signal : s2
De 1 hz c'est le signal : s1
De 2 hz c'est le signal : h2
Et de 5 hz c'est le signal : h5

A vous de les utiliser dans vos combinaisons logiques pour faire clignoter des led

Exemple :

```
if (ad4 > 200) o5= s2 ;        // If ad4 from 200 to 400 o5= 0.5 hz
if (ad4 > 400) o5= s1 ;        // If ad4 from 400 to 600 o5= 1 hz
if (ad4 > 600) o5= h2 ;        // If ad4 from 600 to 800 o5= 2 hz
if (ad4 > 800) o5= h5 ;        // If ad4 from 800 to 1000 o5= 5 hz
if (ad4 > 1000) o5= 1 ;        // If ad4 >100 then o5= 1
```

3.8) Les valeurs rs232

Il vous est possible d'interagir on-line sur le programme logique en utilisant des valeur rs232.
On peut alors en envoyant un caractère rs232 changer le comportement du programme.
Vous disposez des valeurs logiques a,b,c,d,e,f,g,h et p,q,r,s qui peuvent être mises à 1 par le caractère majuscule et être mises à 0 par le caractère minuscule .

Exemple :

```
o6=a and s1 ;                 // Use the a value set by rs232
o7=b and h2 ;                 // Use the b value set by rs232
o8=c and h5 ;                 // Use the b value set by rs232
```

Et en rs232 :

```
>Set a      ( si on tape 'A' -> a=1 au niveaux logique )
>Reset a    ( si on tape 'a' -> a=0 au niveaux logique )
>Set a
>Set b
>Set c
>Set d
```

```

>Set e
>Set f
>Set g
>Set h
>Set p
>Set q
>Set r
>Set s
>Reset a
>Reset b
>Reset c
>Reset d
>Reset e
>Reset f
>Reset g
>Reset h
>Reset p
>Reset q
>Reset r
>Reset s
>

```

3.9) Dump on-line

Pour faciliter la correction d'erreurs on peut afficher les valeurs internes d'un seul coup en tapant le caractère x :

```

>
>Dump all Information

>Input value
i11-i1: 000 00100010

>Output value
o9-o1: 1 00000000

>Analog to digital value
AD1-AD6: 00001 01023 00000 00260 00000 01023

>Memo value
m8-m1 : 00001000 m16-m9 : 00000000
m24-m17: 00000000 m32-m18: 00000000

>Timer value
t8-t1 : 00000000 t16-t9: 00000000

>Timer value ( 1/10 sec )
t1-t8 : 00000 00000 00000 00000 00000 00000 00000 00000
t9-t16: 00000 00000 00000 00000 00000 00000 00000 00000

```

>Loop by sec. = 06840

Remarquez que le programme logique tourne 6.840 x par seconde, donc une réaction inférieure à la milliseconde ; pas mal !

Tapez le caractère '1' pour avoir le nombre de boucles par seconde.

3.10) Envoyer une chaîne rs232

Il est possible d'envoyer une chaîne de caractères sur port rs232 à partir du programme logique en utilisant la fonction `text_tx("chaîne_de_caractères_r\n")`.

Le `\r\n` est le retour à la ligne.

Mais attention il faut utiliser le flanc montant ou descendant des signaux pour ne pas envoyer un message à chaque boucle du programme et saturer le buffer rs232.

Exemple :

```
if (t1 and m4) text_tx("T1 is set \r\n"); // Send rs232 message on front
m4 = not t1 ;                          // Memorise the t1 state
```

Sur le port on a :

>T1 is set

Regardez dans le répertoire demo pour avoir un idée du fonctionnement.

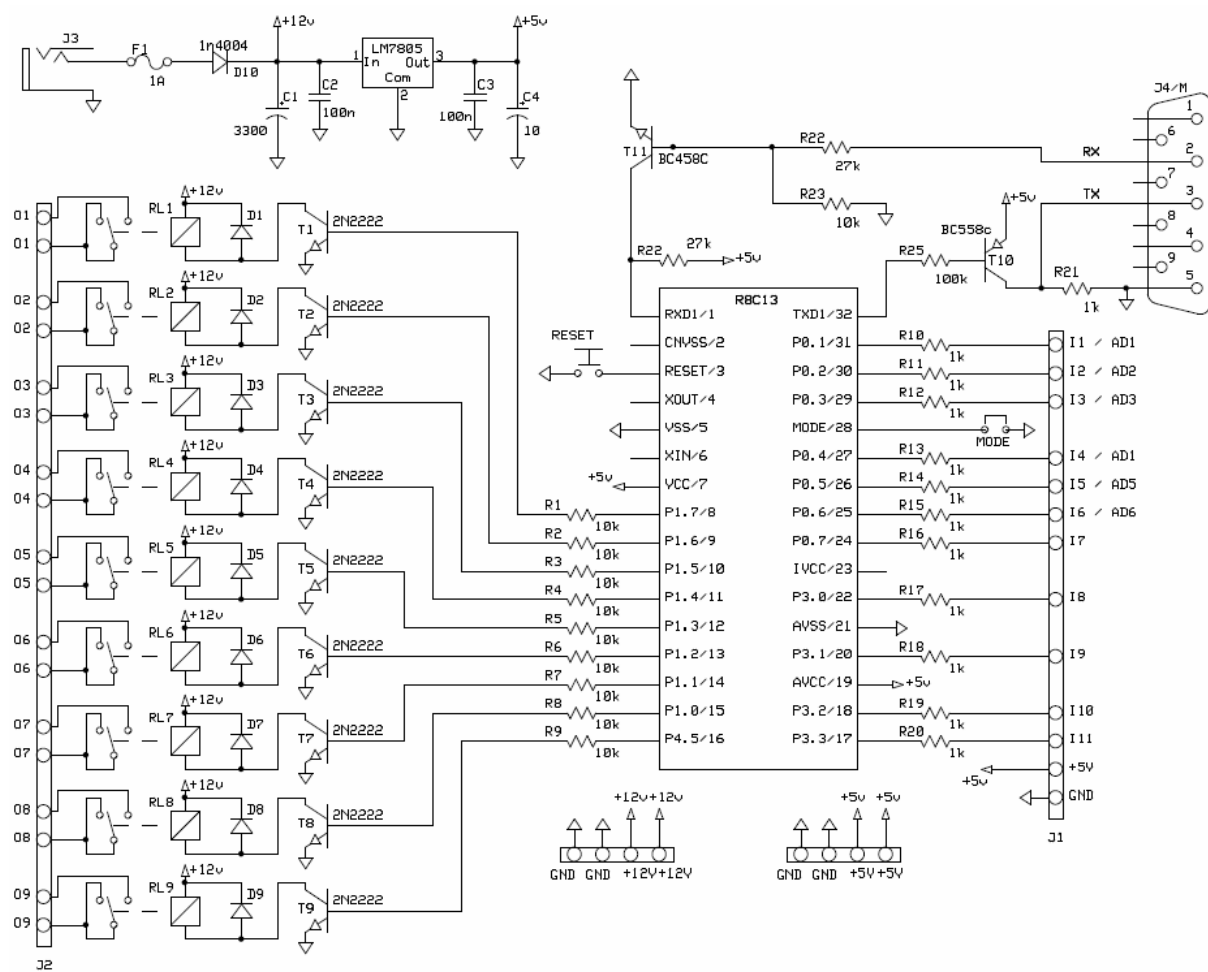
La carte

Je me suis basé sur la carte d'Elektor du mois de Février 2006 légèrement modifiée.

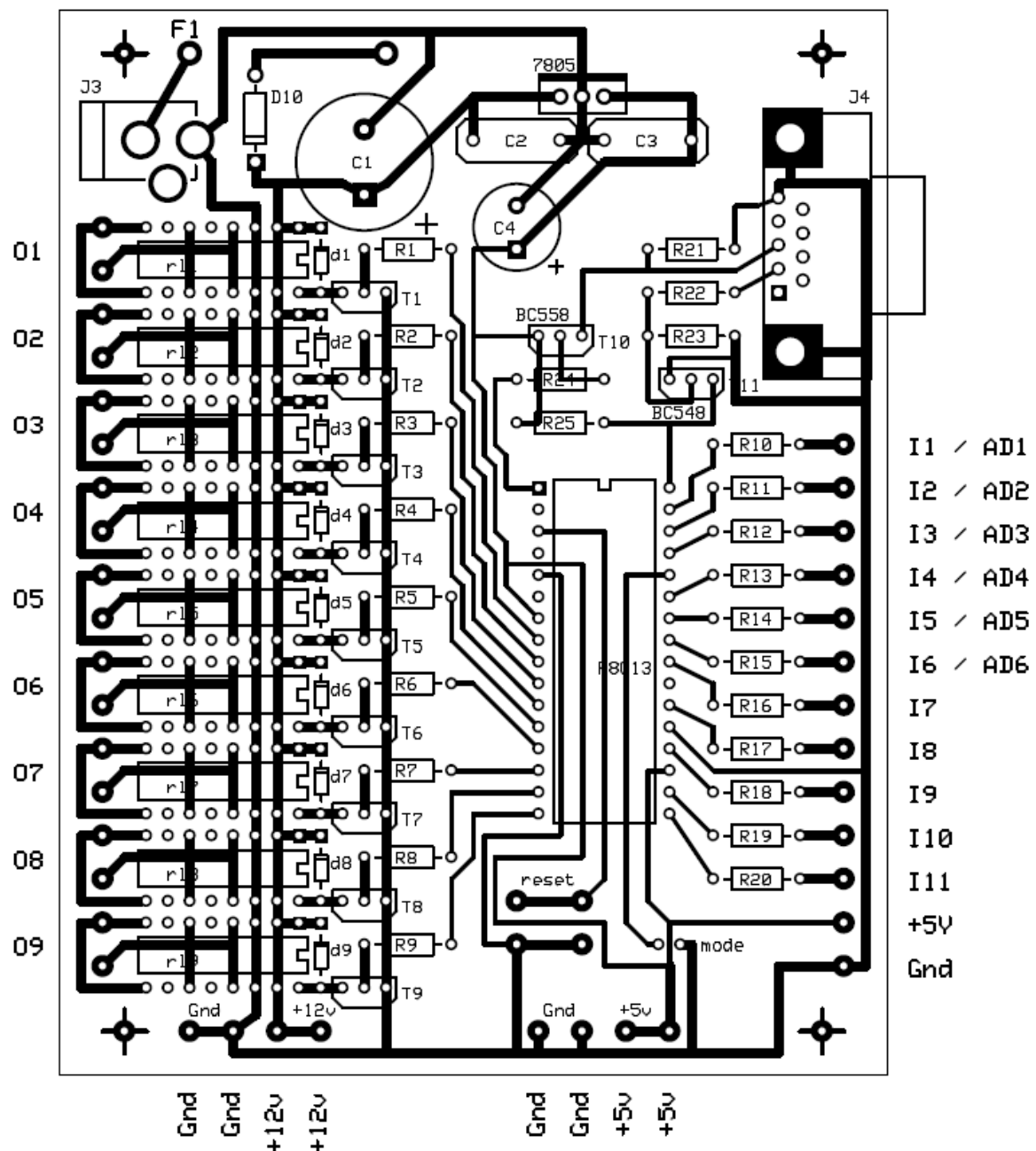
- J'ai supprimé la résistance pull-up de mode de 10K, car elle est déjà présente sur le board R8C/13 – Glyn (R3).
- J'ai appliqué le +5v sur la broche 19 et le 0v sur la broche 21 pour utiliser les entrées analogiques.
- J'ai ajouté une résistance de 1k sur le collecteur du BC558C vers la masse pour monter à une vitesse de 38.400 baud.

Vous avez le circuit et le schéma créé par expresspcb (<http://www.expresspcb.com/>) dans le répertoire board .

Le schéma :



Le circuit :



Je dois encore tester le circuit imprimé, si il y a des modifications je vous tiendrai au courant.

A vous de créer vos propres applications logiques et bon amusement.

Gérard Jacquemin.