Les microprocesseurs sont des « êtres » pluridisciplinaires. L'une de leurs tâches premières est d'acquérir des données, de leur faire subir un traitement quelconque pour ensuite visualiser les résultats de cette opération. On a besoin, pour cette dernière phase, à l'importance souvent capitale, d'un affichage. Le présent montage prouve qu'il est possible, avec peu de composants, de réaliser, à peu de frais, un affichage bien contrasté et partant parfaitement lisible.

Projet : Ingo Gerlach

affichage à LED flexible

à combiner avec nombre de systèmes à microprocesseur

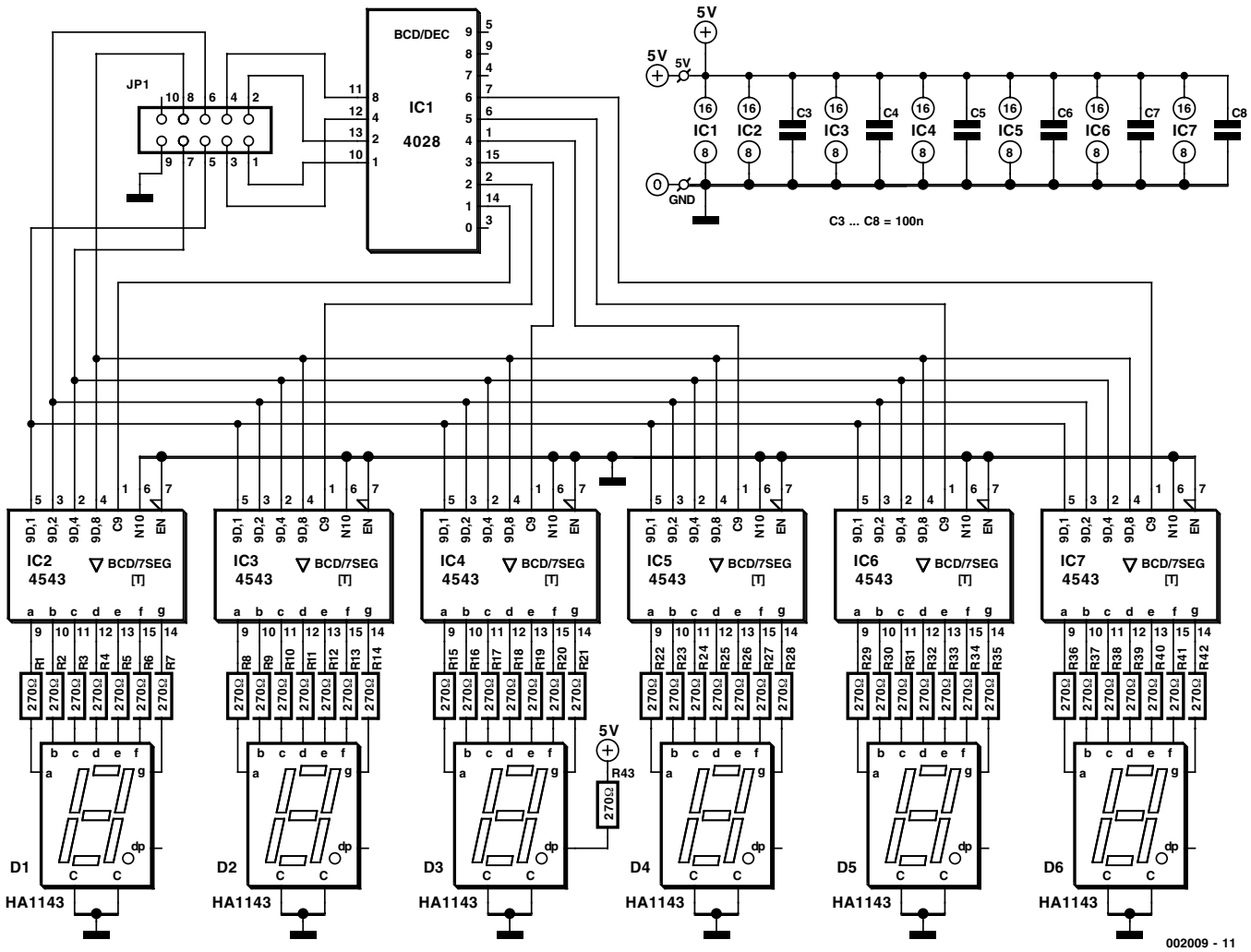

Figure 1. Le schéma de l'affichage à LED. Il n'utilise que des composants courants. L'approche adoptée donne un affichage flexible ne requérant que peu de temps processeur.

L'absence d'un affichage de bonne qualité rendrait nombre de systèmes à microprocesseurs difficilement utilisables. Par nature, les microprocesseurs sont des « boîtes noires » capables, sans que cela ne soit visible de l'extérieur, d'effectuer, très rapidement, une quantité importante de calculs; ces calculs n'étant pas une fin en eux-mêmes (l'utilisateur désirant voir clairement les résultats finaux), nombre de projets requièrent un affichage. De plus en plus souvent on utilise des affichages à cristaux liquides (LCD). Les raisons de ce choix sont évidentes :

- un affichage LCD peut visualiser des chiffres et des caractères alphanumériques;
- le contrôleur de LCD embarqué se charge du pilotage de l'affichage qui ne requiert de ce fait que peu de temps processeur;
- les affichages LCD ne consomment que peu d'énergie;
- ces affichages sont compacts.

Outre ces avantages, les affichages LCD présentent bien évidemment aussi quelques inconvénients. En effet :

- ne donnant eux-mêmes pas de lumière, ils ont besoin, dans des conditions de luminosité limites, d'un rétro-éclairage;
- leur angle de lecture est limité;
- ils ne sont utilisables que dans une plage de température limitée;
- leur contraste est relativement faible;
- ils coûtent assez cher;
- il peut se faire que chacun d'entre eux ait besoin de son propre pilote;
- les dimensions de l'affichage sont réduites;
- leur durée de vie est limitée.

En résumé : dans certains cas il est bon d'opter pour un affichage LCD, dans d'autres un affichage à LED constitue une option mieux justifiée. Nous verrons, dans cet article, qu'il est possible de réaliser, un affichage à LED à faible coût comportant un maximum de 6 afficheurs. Le cahier des charges auquel était confronté le présent concept était le suivant. Le montage devait :

- utiliser le moins de lignes d'E/S possible;
- ne pas coûter cher;
- ne pas requérir trop de temps processeur;
- être flexible et partant facilement extensible;
- pouvoir être combiné à pratiquement n'importe quel microcontrôleur.

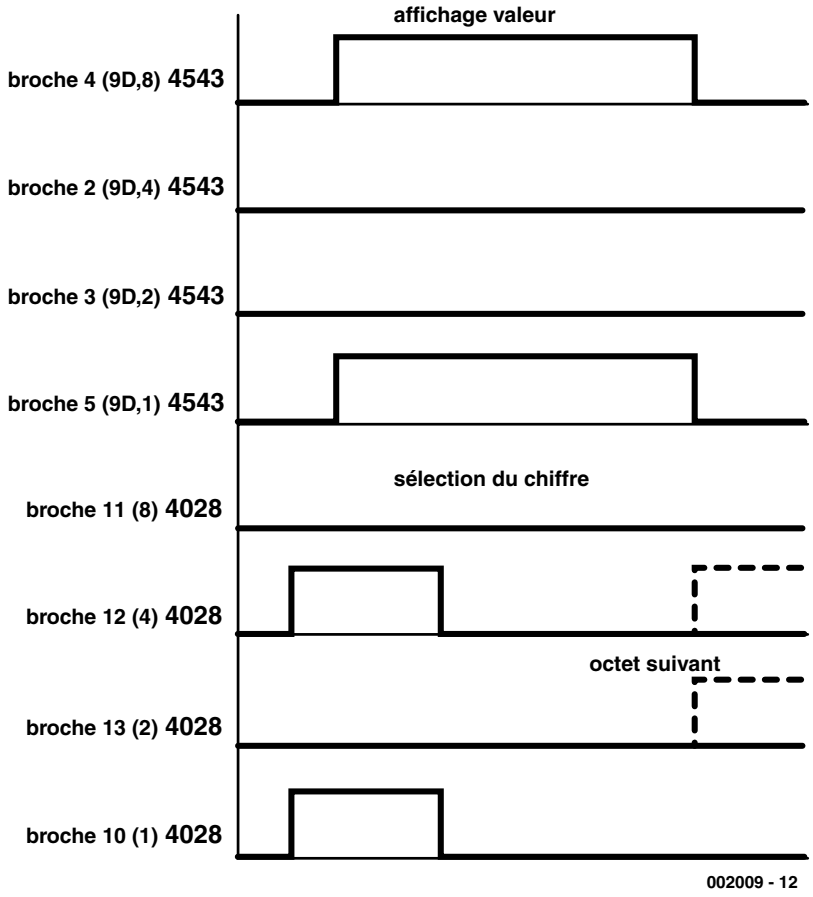

Figure 2. Le chronodiagramme du pilotage. Les 4 signaux supérieurs déterminent le code visualisé par l'afficheur (9 dans le cas présent), les 4 signaux inférieurs l'afficheur attaqué (5 ici).

L'aspect matériel

La **figure 1** nous propose le schéma de cette réalisation. Elle ne requiert que 2 types de circuits intégrés courants et très abordables. IC1 est un 4028, un démultiplexeur tout ce qu'il y a de plus simple, les 6 autres circuits intégrés étant des 4543, un décodeur BCD à verrou (*latch*) et pilote d'affichage (*driver*) intégrés.

Les affichages utilisés sont des afficheurs dits 7 segments à LED, à cathode commune (***C**athode*). Ceci explique que leur broche CC soit reliée à la masse. Le reste des broches, a, b à g, sont reliées directement à la sortie correspondante du pilote d'affichage concerné, IC2 à IC7. Chaque affichage comporte une dernière broche, dp. Cette broche sert à la commande du point décimal. Dans le cas présent, seul le point décimal de l'affichage numéro 3, LED3, est activé. Ceci explique que la broche dp de LED3 soit reliée au +5 V par le biais d'une résistance-talon de 270 Ω, R43. Si l'on préfère mettre le point décimal à un autre niveau, la modification requise est vite faite.

L'ensemble de l'affichage est relié au système à microprocesseur par le biais d'un connecteur 10 points (2 rangées

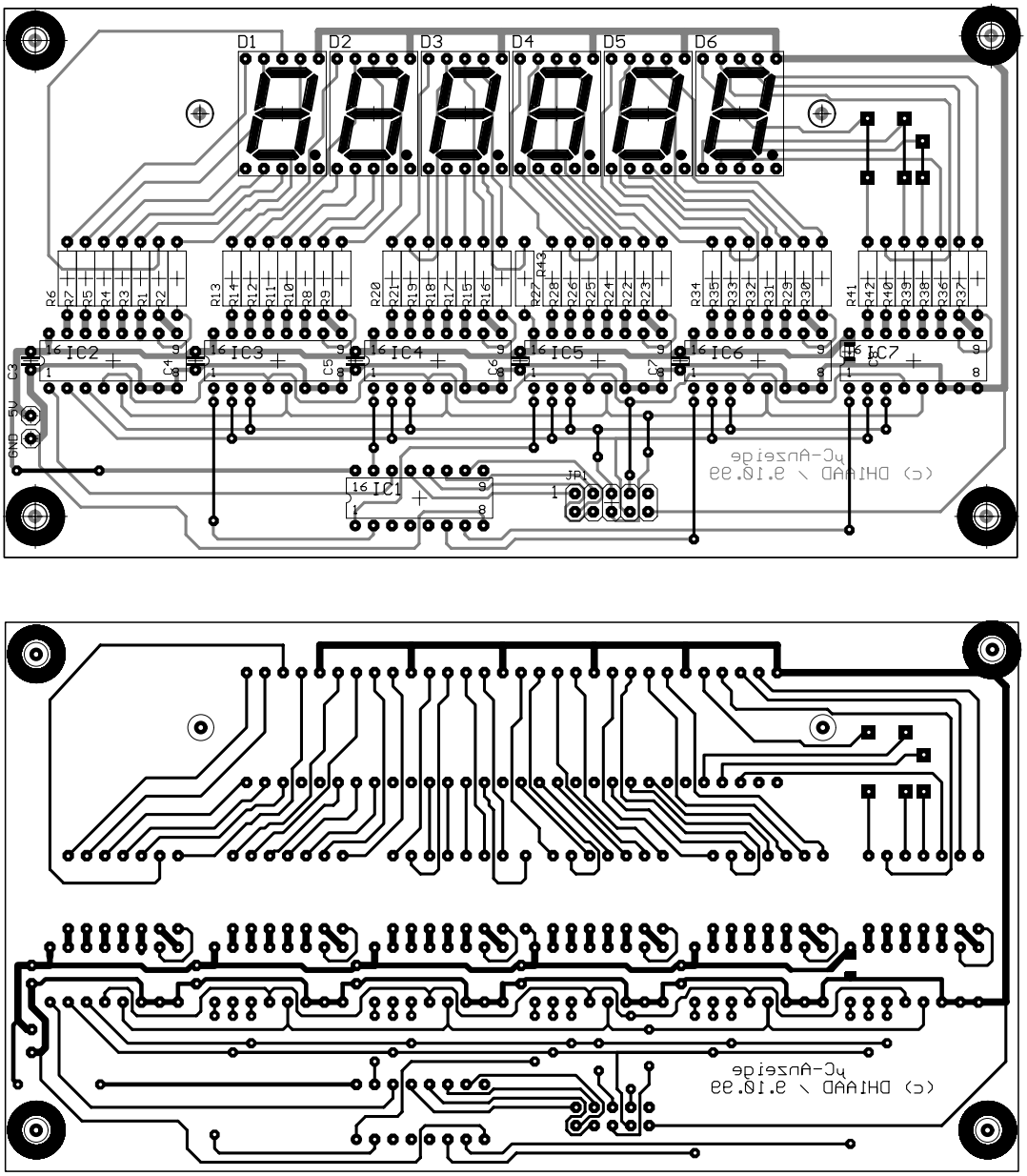

Figure 3. Dessin des pistes et sérigraphie des composants de la platine permettant de réaliser un affichage à LED.

de 5 contacts). 9 seulement de ses broches sont en fait utilisées.
Comme l'une des broches se trouve à la masse, cela signifie que l'on n'utilise que 8 lignes d'E/S pour le pilotage de l'affichage. Les broches 1 à 4 véhiculent un signal à 4 bits servant à la sélection de l'afficheur requis. La combinaison de bits présentées par lesdites broches détermine la sortie de IC1 devant présenter un niveau haut. La combinaison 0000 n'est pas utilisée, la broche 3 (Q0) de IC1 n'étant pas reliée à un afficheur. Les 4 lignes de signal restantes (5 à 8) servent à la génération du code BCD devant attaquer l'afficheur en question.
Dès que le code BCD correct se trouve sur ces lignes, l'entrée LD (***LoaD***, broche 1) du contrôleur concernée est mise au niveau haut. Lors de la transition haut-bas suivante (flanc descendant) le 4543 transfert le code appliqué aux entres 1A à 1D (broches 2 à 5). Une matrice intégrée dans le composant détermine alors les sorties (A à G) à mettre au niveau haut. Les segments correspondants de l'afficheur s'allument. Le chronodiagramme de ce processus est visualisé en **figure 2**.
Les 4 bits du haut montrent que c'est le code BCD 1001 que l'on applique aux entrées du pilote d'affichage. On active ensuite, par le biais du 4028, IC5, le code 0101 (5_D). L'approche adoptée présente 2 avantages importants :
Il n'est pas nécessaire, tant qu'il ne faut pas modifier l'affichage, d'envoyer de nouveaux signaux de commande à l'affichage.
Il est en outre possible de ne modifier qu'un seul afficheur.
Ces 2 caractéristiques réduisent très sensiblement la charge du processeur. Le reste du circuit n'appelle que fort peu de commentaires.
L'alimentation est une alimentation asymétrique de 5 V. Chaque circuit intégré a été doté de son condensateur de découplage.
La valeur des résistances-talon est celle que requiert la tension d'alimentation choisie. On pourra, pour forcer quelque peu la luminosité des afficheurs, abaisser à 180 Ω la valeur des résistances-talon. On pourra au contraire, si l'on préfère une luminosité moindre (et partant une consommation de courant plus faible), augmenter en conséquence leur valeur.

La platine

L'auteur a réalisé un dessin de platine que nous vous proposons en **figure 3**. Il vous faudra la graver vous-même, mais comme il s'agit d'une simple face, cela ne devrait pas poser de problème.
On commencera par la mise en place des ponts de câblage. Ce sera ensuite tour du reste des composants. Les circuits intégrés seront, de préférence, montés sur support. Attention à soigner les soudures. La résistance la plus à droite associée à IC2 à IC7 n'est pas numérotée : il s'agit de la résistance de pilotage du point décimal, notre R43 du schéma. À vous de choisir s'il faut un point décimal et où le placer. Il suffira ensuite de monter la résistance-série (270 Ω) requise pour activer le point décimal correspondant.

Programmes à titre d'exemple

Le pilotage de l'affichage à LED en combinaison avec un microcontrôleur requiert du logiciel. Nous vous proposons, à titre d'exemple, un programme de pilotage (une routine en langage machine dans le cas présent) pour les processeurs AT90S1200 à 8515 d'Atmel. Nous avons opté, de manière à faciliter l'adaptation autant que faire se peut, pour un pilote distinct.
On pourra ainsi appeler ce pilote dans une application. Il suffit dans ces conditions d'écrire le pilote une fois pour toutes pour qu'il soit utilisable avec différents systèmes. Le **listage 1** constitue le pilote proprement dit. Il est

```
;***************************************************
;* File Name    :LEDDisp.inc
;* Title              :Driver for LED Display
;* Date         :Ingo Gerlach / 10.10.99
;* Version            :1.0 / 11.10.99
;* Version            :
;* µC                 :AT90S1200...8515
;*                    :
;* Changes            :
;*
;***************************************************
;
;
; Main program register variables
;-------------------------------------------
;.def   temp    = r16
; Registers  / LED
;-------------------------------------------
;.def   cntr    = r20   ; counter
;.def   dly     = r21   ; delay loop variable
;.def   pos     = r23   ; position
;.def   byte    = r24   ; byte

; Equates
;-------------------------------------------
;.equ   LED_qty = 6             ; number of LEDs
;.equ   LED_Del = 45            ; delay
;.equ   OutPort = PortB

; Functions
; LED_Blank     : switch display on
; LED_Null      : reset display 0 ( Null)
; LED_Show      : show bytes , transport byte (R24),
position (R23)

; **** Switch display off
;*************************
LED_Blank:      ldi     cntr,LED_QTY
LedLoop1:       ldi     temp,192
                add     temp,cntr
                out     OutPort,temp
                Rcall   Led_Delay
                dec     cntr
                brne    LedLoop1
                Ret

;*************************
; **** Reset display
;*************************
LED_Null: ldi  cntr,LED_QTY   ; load number of LEDs
LedLoop2: out  OutPort,cntr
                Rcall   Led_Delay
                dec     cntr
                brne    LedLoop2
                Out     OutPort,cntr
                Ret
;*************************
; **** Show byte
;*************************
LED_Show:  mov  temp,pos ; position in register
                out       OutPort,temp; activate BCD-
to-decimal decoder, LD 4543
                Rcall     Led_Delay ; short delay
                mov       temp,byte ; value in
register
                swap      temp         ; value high
nibble
                add       temp,pos  ; goto position
                out       OutPort,temp; 4028 / 4543
move value to display
                Rcall     Led_Delay ; short delay
                sub       temp,pos
                out       OutPort,temp; LD signal
off, store value
                Rcall     Led_Delay
                Ret
;*************************
; * Internal functions !!!

; *** Delay LED display
;**********************
LED_Delay:      ldi dly,LED_Del
LedLoop:        dec dly
                brne LEDLoop
                ret
```

Listage 1. Le pilote d'affichage à LED écrit en langage machine pourra être facilement intégré à nombre d'applications.

facile d'en suivre le fil.
Le **listage 2** est un petit programme de démonstration. Le programme commence par mettre l'affichage à 0 avant d'afficher le nombre 145675. Si l'on a opté pour un point décimal au niveau du 3ème afficheur, ce nombre sera 145.675.
Vous pouvez maintenant, avec cet affichage à LED flexible, procéder vos propres expériences en toute quiétude. S'il vous faut adapter le code pour d'autres familles de processeurs, les commentaires donnés dans les listages devraient vous faciliter la vie.

(002009-1)

Texte : Hans Steeman

```
;************************************************
;* File Name    :LED.asm
;* Title              :Test program for LED display
;* Date         :Ingo Gerlach / 10.10.99
;* Version            :1.0 / 10.10.99
;* Version            :
;* µC                 :AT90S1200...8515
;*                    :
;* Changes :
;*
;************************************************
;

;***** Directives


.device AT90S1200              ;device type
.NOLIST

.include "1200def.inc"


.list
.listmac



; Show data
; Structure of data

;       MSB                LSB
;       7 6 5 4        3 2 1 0
;       0 0 0 0        0 0 0 0
;
;       Select position 1 = 1. 2 = 2. etc
;       Number in BCD code
;
;e.g.   10010010b = 146d = pos. 2, value 9


;
; Main program register variables
;---------------------------------------
.def    temp           = r16


; Registers  / LED
;---------------------------------------
.def    cntr           = r20   ; counter
.def    dly            = r21   ; delay loop variable
.def    pos            = r23   ; position
.def    byte           = r24   ; byte


; Equates
;---------------------------------------
.equ    LED_qty = 6            ; number of LEDs
.equ    LED_Del = 40           ; delay
.equ    OutPort = PortB


;***** Interrupt vector table

reset:
    rjmp  main  ; main routine
    reti        ; external interrupt0 handle
    reti        ; T/C0 overflow interrupt handle
    reti        ; analogue comparator interrupt handle

;***** Functions
************************************************


;***************************************
;***** Main ******************************

main:
;   ldi   temp, LOW(RAMEND)  ; setup StackPointer for
> 90S1200
;   out   SPL, temp          ; initialize SPL
;   ldi   temp, HIGH(RAMEND)
;   out   SPH, temp          ; initialize SPH


    ldi   temp,255           ; temp = 255
    out   ddrb,temp          ; port B output

    Rcall LED_Null           ; reset display

mainloop:

; Show  145.675

                ldi     Pos,1
                ldi     Byte,1
                Rcall   Led_Show

                ldi     Pos,2
                ldi     Byte,4
                Rcall   Led_Show


                ldi     Pos,3
                ldi     Byte,5
                Rcall   Led_Show

                ldi     Pos,4
                ldi     Byte,6
                Rcall   Led_Show

                ldi     Pos,5
                ldi     Byte,7
                Rcall   Led_Show

                ldi     Pos,6
                ldi     Byte,5
                Rcall   Led_Show


forever:        rjmp forever

; ***** End of main program ******************


; ***   Include Files  ***
.include "LEDDisp.inc"
```

Listage 2. Exemple de programme illustrant l'intégration du pilote dans une application donnée.