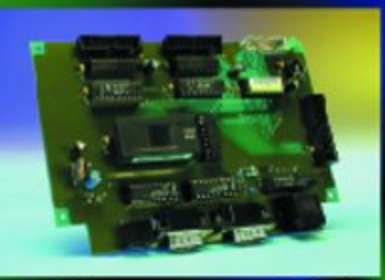
ELEKTOR

ELEKTOR

ELECTRONIQUE et MICRO-INFORMATIQUE

N° 279
SEPTEMBRE 2001
mens uel
30 F/€ 4,57
216 FB/€ 5,95
9,60 FS • 6,25 $Can.
BIN 0181-7450
www.elektor.presse.fr



Mini-serveur Web pour se faire la main

Programmateur Atmel pour les 89Cx051 jusqu'à 4 K

Encore plus loin avec le BS2p

Interface I²C pour servos

Les accus Li-ion

Affichage LCD graphique

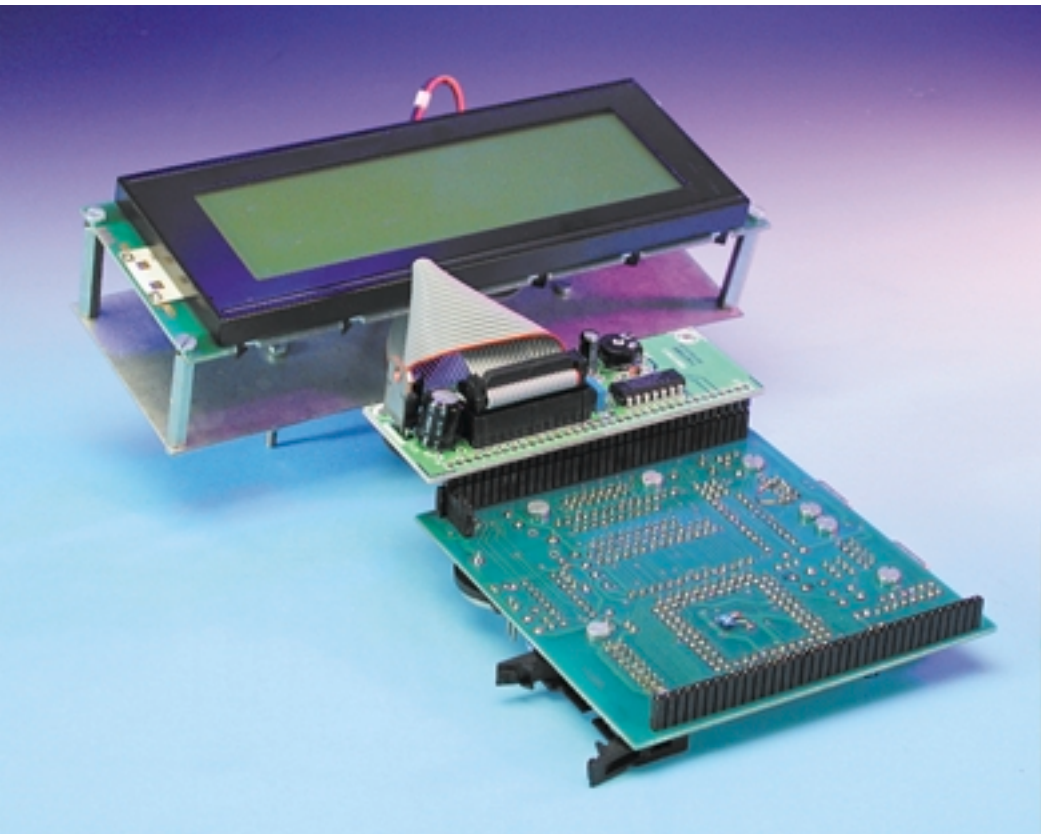
pour carte à 537

Module graphique LCD sur µP 8051

Pixels et octets en goguette

Luc Lemmens

Un de ces modules LCD, doté du µC T6963 de Toshiba, tellement répandu, comment le commander à partir d'un microcontrôleur de la mouvance du 8051 ? Comment mettre à profit la totalité de l'affichage pour en faire un écran graphique et transformer l'image fournie par le PC en pixels de cristaux liquides correspondants ?



Sur les écrans alphanumériques qui nous sont familiers – nous les avons utilisés dans de nombreux projets d'Elektor – on peut dire qu'avec son contrôleur HD44780, Hitachi a défini un vrai standard industriel. Presque tous les fabricants l'on repris dans leur production de modules LCD, et cela, quel que soit le modèle d'afficheur qui leur sont adjoints. À cet égard, c'est plutôt par le connecteur que les marques se différencient entre elles.

Avec les modules **graphiques** monochromes, la situation est assez comparable, bien que de standard, il en soit beaucoup moins question. Un contrôleur fréquemment rencontré, c'est le **T6963** de Toshiba, aussi allons-nous le prendre ici en exemple et examiner comment commander un afficheur de ce genre à partir d'un processeur de la famille 8051. Dans cette optique, nous avons profité d'une platine déjà réalisée, la **Monocarte « 537-Lite »**, équipée du 80C537, décrite dans les numéros 259 et 260 (janvier et février 2000), avec son moniteur en EPROM (**EPS976510-1**), mais en principe tout système à microcontrôleur conviendrait aussi bien.

L'afficheur graphique qui nous a servi de cobaye pour ce projet est un **LM24014** de Sharp, un modèle qui compte 260 x 64 pixels, mais de nouveau, précisons que n'importe quel affichage doté du microcontrôleur T6963 s'acquittera tout autant de

cette tâche d'expérimentation. Pour un afficheur d'une autre définition, il y aura nécessairement une légère adaptation à apporter dans le logiciel proposé ici.
Il y aurait pas mal de choses à raconter à propos de ce type d'affichage, mais la place nous manque dans le cadre de cet article. Si vous envisagez d'autres applications ou possibilités, comme par exemple l'utilisation en mode texte (comparable à tout module alphanumérique), vous pouvez utilement vous en remettre au feuillet de caractéristiques du contrôleur T6963 ou à la documentation du système d'affichage de votre choix, équipé du même processeur.

Le module LCD

Pour commander le LM24014, nous faisons appel à trois registres du module graphique : les registres de données, de commande et d'état. La communication s'établit en écrivant tout d'abord une, deux, voire aucune valeur dans le registre de données, puis en envoyant une instruction au registre de commande. Après quoi, selon l'instruction considérée, on peut lire en réponse une valeur dans le registre de données. Il faut toujours consulter le registre d'état avant chaque opération de lecture ou d'écriture afin de vérifier que l'affichage a mené à bonne fin la tâche précédente et se retrouve en mesure de réceptionner et traiter les nouvelles données.
Il est possible d'adresser directement le registre de commande et celui d'état. Si nous voulons lire l'état courant, il nous faut au préalable expédier au registre de commande une instruction *Read Status*, à la suite de quoi l'octet d'état est mis à disposition dans le registre de données. La **figure 1** permet de voir qu'il y a différents bits d'état pour indiquer quel travail l'affichage peut exécuter.

Interface

Parmi les 20 connexions de l'afficheur, nous trouvons 8 lignes de données bidirectionnelles, une mise à zéro et 4 signaux de contrôle ($\overline{CE}$, $\overline{RD}$, $\overline{WR}$ et C/$\overline{D}$, mais nous allons y revenir). Le module LCD dispose aussi d'une ROM interne pour les caractères, exactement comme les afficheurs alphanumériques bien connus, et la broche 19 donne latitude de choisir

Status Bit	Description		Function
STA0 (BUSY1)	Busy flag to indicate whether T6963C is ready to accept instruction	"0""1"	=NOT READY =READY
STA1 (BUSY2)	Busy flag to indicate whether T693C is ready to accept data read or write	"0" "1"	=NOT READY =READY
STA2 (DARRDY)	Data Auto Read Ready flag (Only valid in Data Auto Read/Write modes (section 8.6))	"0""1"	=NOT READY=READY
STA3(DAWRDY)	Data Auto Read Ready flag (Only valid in Data Auto Read/Write modes (section 8.6))	"0""1"	=NOT READY =READY
STA4	-	-	-
STA5 (CLR)	Clear flag indicating operation of the T6963C	"0""1"	=NOT CLEARED =CLEARED & Operating
STA6 (ERROR)	Error flag for Screen Peeking and Screen Copy commands (section8.8 & 8.9)	"0""1"	=Address Pointer Valid = Address Pointer out of Graphic Area
STA7 (BLINK)	Blink flag to indicate status of Blink condition	"0" "1"	=Display OFF =Normal (Display ON)

000134- 12

Figure 1. Le registre d'état dans tous ses états.

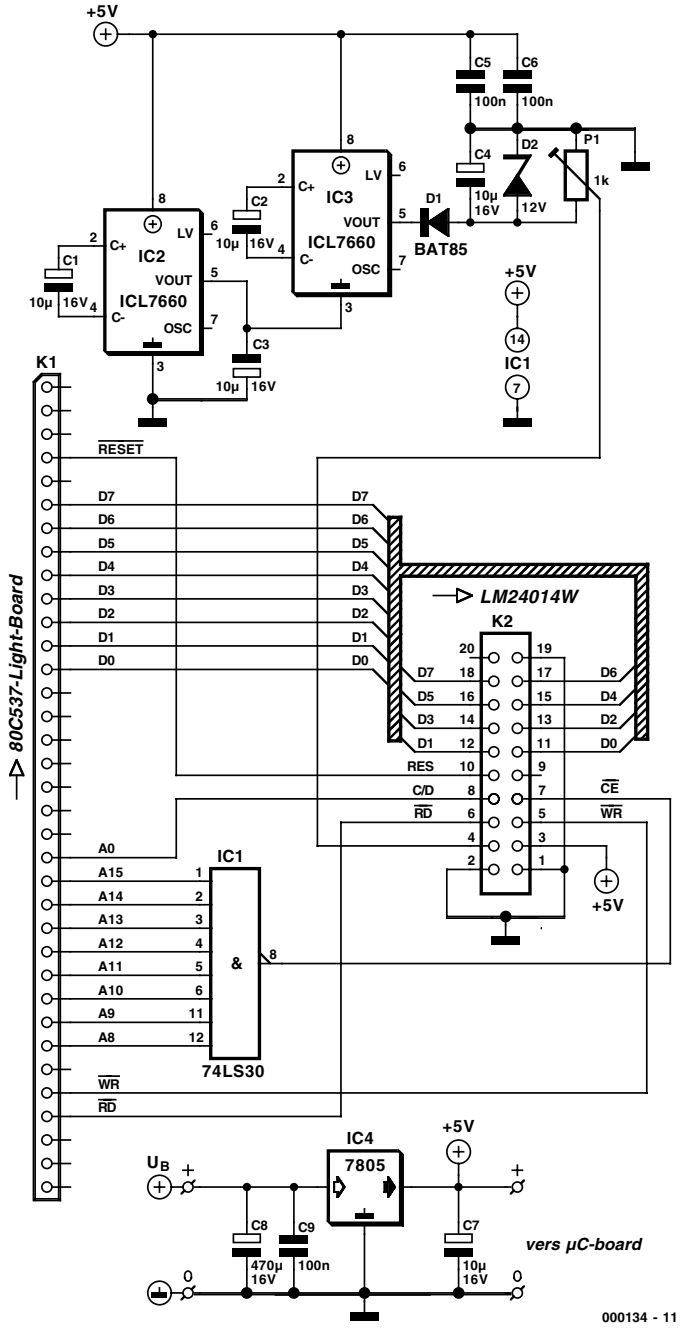


Figure 2. L'interface pour relier l'afficheur à la mono-carte à 80C537.

entre des signes de 6 x 8 ou de 8 x 8 pixels. L'affichage travaille sous +5 V, mais pour le réglage de contraste, il faut au LM24014 une tension auxiliaire négative de relativement grande amplitude, le feuillet de caractéristiques indique entre –12 V et –6 V. Les écrans modernes se satisfont souvent d'un seuil de diode sous le zéro, voire du zéro lui-même. Un point à vérifier dans les spécifications si vous faites appel à un autre modèle d'afficheur.

L'écran se relie à la mono-carte à 80C537 à l'aide du montage représenté à la **figure 2**. Le bus de données de l'afficheur se branche directement au bus correspondant de la carte à microcontrôleur. IC1 est un décodeur d'adresse qui s'arrange pour que l'on puisse joindre l'afficheur dans le domaine d'adresses $0FFXX_{HEX}$. La sortie de ce port est reliée au $\overline{CE}$ (*Chip Enable*, sélection de boîtier) du module LCD. La ligne d'adresse A0 se branche à C/$\overline{D}$, le signal de commande qui assure la sélection du registre interne de commande ou de données de l'afficheur. C'est grâce à cette sélection que l'on peut lire ou écrire dans le registre de données à l'adresse $0FF00_{HEX}$ et atteindre le registre de commande à l'adresse $0FF01_{HEX}$. La mise à zéro, les commandes de lecture ($\overline{RD}$) et d'écriture ($\overline{WR}$) de l'affichage sont reliées aux signaux de même nom de la platine du microcontrôleur. Pour ne rien laisser au hasard, la platine d'interface est aussi équipée d'un régulateur de tension 7805 pour assurer l'alimentation de l'affichage, mais également fournir, au besoin, l'alimentation de la platine du processeur, via les broches 5 (GND) et 9 (+5 V) de K1.

La **figure 3** présente la platine du montage d'interface. Comme elle est disponible auprès des adresses habituelles, tout un chacun aura donc la possibilité de réaliser le projet sans difficulté.

Gestion logicielle d'écran

Dans l'ordinogramme de la **figure 4**, on peut suivre l'ordre dans lequel les données doivent parvenir à l'écran, lequel doit, naturellement, être mis sous tension au préalable et subir une initialisation à chaud (*power on reset*). Nous verrons d'ailleurs que cette opération a lieu de concert avec le signal de RESET de la carte du microcontrôleur.

Avant que l'affichage ne soit en mesure de reproduire des images, il faut lui communiquer certains paramètres. Il s'agit, dans le désordre, de définir la largeur du domaine graphique sur écran, l'adresse de départ de la zone graphique, le mode d'affichage et le passage commandé par logiciel de la reproduction graphique. Il n'est donc pas question d'une initialisation comme sur un affichage purement alphanumérique. Les paramètres que nous assignons ici, nous pourrons toujours les modifier à un stade ultérieur.

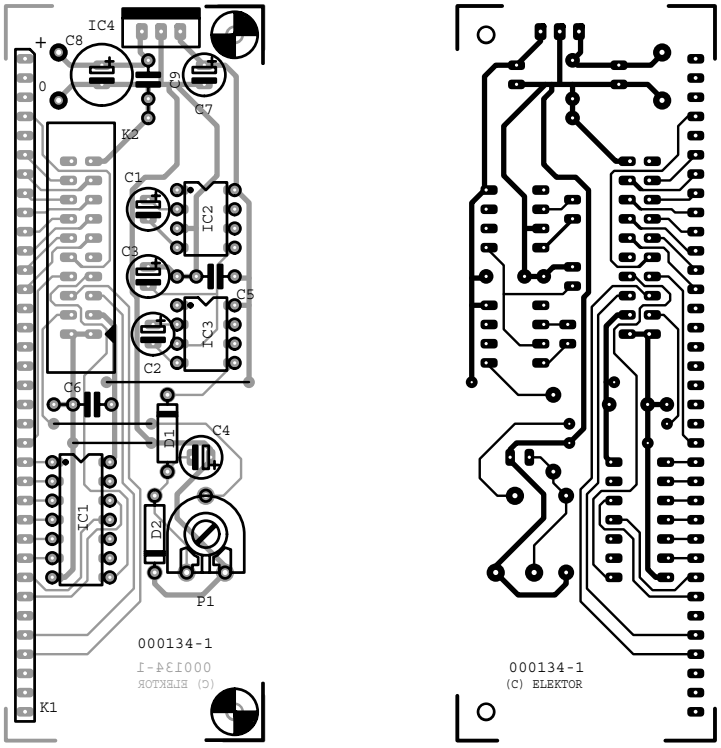


Figure 3. Les pistes et la répartition des composants du montage de la figure 2.

Zone graphique

La largeur de la zone graphique (GA, *Graphic Area*) est influencée par le niveau à la broche 19 de l'écran, celle qui permet de choisir entre des caractères de 6 x 8 ou de 8 x 8 pixels. Nous verrons également que l'écran se remplit du coin supérieur gauche vers le coin inférieur droit, par lignes horizontales d'un pixel de haut et (dans notre cas) de 8 pixels de large. Si nous choisissons le format 6 x 8 (+5 V sur la broche 19), les octets de données envoyées à l'affichage ne concernent que 6 pixels, les deux bits de poids fort sont simplement ignorés. En 8 x 8 (broche 19 à 0 V), en revanche, on prend en considération ces deux bits. Notre écran comporte 240 pixels en largeur, en format 8 x 8, la zone graphique compte donc 240 / 8 = 30 signes ($01E_{HEX}$ en hexadécimal) en largeur, alors qu'en 6 x 8, il y en a 240 / 6 = 40 signes, soit 028_{HEX} exprimé en hexadécimal.

Première adresse graphique

La mémoire d'un module d'affichage est communément plus vaste que l'ensemble des données à rendre sur écran. C'est pourquoi il nous faut indiquer l'adresse mémoire du bloc situé en haut, à gauche de l'écran : la première adresse graphique (GHA, *Graphic Home Address*). Son contenu représente donc la valeur des huit premiers pixels, un segment horizontal sur écran. L'adresse suivante concernera évidemment les huit pixels situés immédiatement à la droite du premier segment. La deuxième ligne d'écran commence, dans notre cas, trente adresses au-delà de celle de départ et chacune des suivantes démarre à l'adresse GHA + n x GA. Après avoir introduit une image dans la mémoire graphique, nous pouvons la faire défiler verticalement sur écran, il suffit d'augmenter ou de diminuer GHA par multiples (n) de GA.

Mode d'affichage

Outre des actions comme la présentation ou l'extinction du curseur, nous pouvons indiquer, grâce au mode d'affichage (*display mode*), le chevauchement nécessaire entre texte et graphe. En mode OR, il est possible de déterminer pour chaque pixel s'il est destiné au texte ou au dessin. Les autres modes se nomment AND (le pixel apparaît s'il appartient à la fois au texte et au dessin) et EXOR (le pixel n'est visible que s'il est affecté à un seul de ces domaines).

Liste des composants

Résistances :
P1 = 1 kΩ ajustable

Condensateurs :
C1 à C4,C7,C12 = 10 µF/16 V
C5,C6,C10,C11 = 100 nF
C8,C9 = 470 µF/16 V

Semi-conducteurs :
D1 = BAT85
IC1 = 74LS30
IC2,IC3 = ICL7660
IC4 = 7805

Semi-conducteurs :
K1 = embase auto-sécable SIL à 1 rangée de 35 contacts
K2 = embase mâle à 2 rangées de 10 contacts

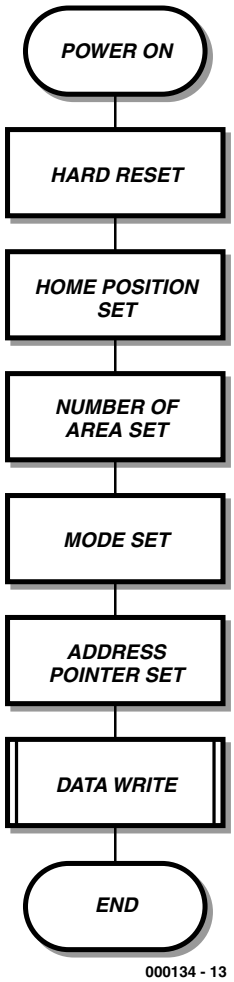


Figure 4. Dans cet ordinogramme, la séquence à respecter pour l'envoi de données à l'écran.

Des données pour l'écran graphique

Ces préparatifs une fois terminés, on peut inscrire des données dans la mémoire graphique du module LCD. La GHA fixée, plus de doute sur l'emplacement, en mémoire, des données relatives au premier segment (les huit pixels en haut à gauche de la première ligne balayée). Comment pointer vers le suivant, c'est l'affaire du pointeur d'adresse (*Address Pointer*), qu'il faut mettre à jour avant de transférer les données correspondantes vers le module. Mais le procédé est particulièrement pénalisant, puisqu'il demande de fournir chaque fois deux octets d'adresse pour pouvoir expédier les données de huit pixels seulement. Aussi, l'afficheur connaît-il un mode de remplissage automatisé, *Auto Write*, par lequel on donne une adresse de départ, après quoi les octets successifs sont considérés comme des données. L'écran s'occupe d'initiative d'incrémenter (ou de décrémenter !) le pointeur d'adresse après traitement de chaque octet. Après expédition du dernier octet de données, il faut bien entendu désactiver le mode *Auto Write*.
Remarquons au passage qu'il est possible de pointer vers un seul pixel et même d'aller lire sa valeur dans la mémoire graphique du module. Nous pouvons alors lire l'octet dans lequel se situe le pixel considéré, puis à l'aide d'une instruction AND ou OR modifier la valeur de ce bit, voire de plusieurs bits dans le même octet, et le récrire à la même adresse dans le module d'écran.
La communication avec l'affichage n'a rien de compliqué, mais comment s'y prendre pour transcrire une image en octets ? Dans un passé pas très lointain, on se servait d'une feuille de papier quadrillé : on coloriait les carrés pour tracer l'image en grand format, puis on comptait huit carreaux contigus pour les convertir en un octet (carré noir à 1, carré blanc à 0). C'est bien la méthode à suivre, mais avec des outils plus évolués !

Du graphe à l'octet

C'est bien sûr le PC qui va nous offrir son aide pour opérer la transcription pour écran graphique. Nous reproduisons l'image en Paint, un logiciel courant de dessin fourni avec Windows, ou bien, si nous disposons déjà de l'image sous forme de fichier, nous en adaptons les dimensions pour la mettre aux mesures de l'écran et l'enregistrons sous forme de grille de points au format bitmap (.BMP). Nous faisons alors appel à une application écrite en Delphi (version 2.0) pour traduire les pixels du graphe en octets de données pour l'afficheur. Ces octets viennent s'ajouter automatiquement à un listage en assembleur qu'il faudra alors traduire en assembleur pour 8051 et sauvegarder en fichier de format Intel HEX. Un logiciel de terminal permet ensuite de le transférer, par port sériel, dans la mémoire de programme de la mono-carte à 80C537 et de lancer l'application. Finalement, nous allons pouvoir admirer, sur l'écran graphique, le résultat de nos efforts et le fruit de notre inspiration artistique.

Composer des images en Paint

Pour réaliser un dessin (convenable) à reproduire sur l'afficheur, lançons tout d'abord Paint (dans le menu Démarrer, section **Programmes, Accessoires**). Cliquons sur **Image, Attributs**... et précisons les valeurs suivantes (cf. **figure 5**) :

Unités :	Pixels
Palette de couleurs :	Noir et blanc
Largeur :	celle de l'afficheur (pour nous : 240)
Hauteur :	celle de l'afficheur (chez nous : 64)

Nous pouvons à présent mettre à l'œuvre les outils de dessin pour créer une nouvelle image. Mais nous pourrions aussi bien aller en chercher une enregistrée dans un format graphique accepté par Paint (BMP, GIF, etc.) et l'agrandir ou la rétrécir pour qu'elle se cadre au mieux dans la fenêtre. Au moment de sauvegarder le fichier, pensez à préciser que vous en voulez un bitmap monochrome !

Conversion du bitmap

Les fichiers bitmap contiennent, outre le dessin lui-même, un tas d'informations telles que

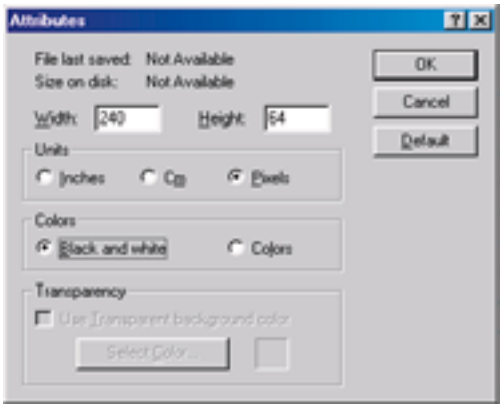


Figure 5. Au lancement de Paint, voici les choix à opérer.

le nombre de couleurs, les dimensions et ainsi de suite. Nous n'en avons pas besoin. Aussi allons-nous rappeler Delphi pour extraire du fichier bitmap l'information pure sur les pixels. En Delphi, nous nous servons d'un outil, une grille sous-jacente, appelée Canvas, une sorte de trame sur laquelle nous pouvons dessiner. Le logiciel se charge de la conversion en fichier BMP du graphe et le canevas contient alors les pixels dans un tableau à deux dimensions, l'avatar actuel de notre bon vieux papier quadrillé. Le programme ne tient pas compte des dimensions apparentes du dessin, Canvas s'adapte d'initiative à sa grandeur. Il n'est pas nécessaire de le modifier si l'on a choisi un autre modèle d'écran. Durant l'exécution d'une double boucle FOR, chaque pixel sera examiné à tour de rôle pour savoir s'il doit être blanc ou noir. Puis les pixels seront groupés par huit consécutifs, puisque c'est cette présentation dont nous aurons besoin dans le module graphique.
Le logiciel Delphi ouvre immédiatement le fichier assembleur TEST.A51 et y ajoute les informations à l'aide d'instructions DB (*Define Byte*). L'image se loge en mémoire de programme du système cible sous forme de tableau, d'où elle peut instantanément être reproduite à l'écran.

Reproduction d'image

À l'issue des étapes que nous venons de parcourir, nous disposons du programme assembleur dont la mission sera de présenter l'image sur l'afficheur. C'est un logiciel assez simple. En premier lieu, on appelle par leur nom logique les adresses de trois registres du 8051 ainsi que les registres de données et de commande de l'afficheur. Puis on déclare les paramètres de l'écran (GHA, GA, Mode Set, Address Pointer Set) et on le passe en mode Auto Data Write, de manière à ce que le pointeur d'adresse soit incrémenté automatiquement après réception de chaque octet de données. C'est donc la même structure qui va se répéter : transmettre d'abord les données (éventuelles) au registre de données de l'afficheur et ensuite l'instruction vers le registre de commande, de façon telle que le module LCD sache

Listage du fichier assembleur

```
ORG 0100H
; Definition of special function registers in the 8051
; data pointer DPTR, 16 bit register represented by DPL and DPH
DPL EQU 082H
DPH EQU 083H
; accumulator
ACC EQU 0E0H

; two registers of the graphic module: data and command register
data EQU 0FF00H
command EQU data+1



   ACALL chk01 ; check if the display is ready to receive data

; the Graphic Home Address will be set. First we'll send the LSB of
; this address (0H)
   MOV DPTR,#data ; DPTR points to LCD data register
   MOV A,#0H
   MOVX @DPTR,A

   ACALL chk01 ; check if the display is ready to receive data
; then the MSB (0H)
   MOV DPTR,#data
   MOV A,#0H
   MOVX @DPTR,A
   ACALL chk01 ; check if the display is ready to receive command
; and instruction Graphic Home Address Set (42H)
   MOV DPTR,#command
   MOV A,#42H
   MOVX @DPTR,A
   ACALL chk01 ; check if the display is ready to receive data

; The following instructions will set the Graphic Area (GA). The display is
; 240 pixels wide, 240/8 = 30 = 001EH
   MOV DPTR,#data
   MOV A,#1EH ; LSB of GA
   MOVX @DPTR,A
   ACALL chk01


   MOV DPTR,#data
   MOV A,#0 ; MSB of GA
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#43H ; Graphic Area Set = instruction 43H
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#80H ; Mode Set: logical OR
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#98H ; Display Mode Set: text off, graphics on
   MOVX @DPTR,A
   ACALL chk01

; The following instructions will set the LCD graphic Address Pointer to
; address 00H.
   MOV DPTR,#data
   MOV A,#00H ; LSB of the Address Pointer
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#data
   MOV A,#0 ; MSB of the Address Pointer
   MOVX @DPTR,A
   ACALL chk01

   MOV DPTR,#command
   MOV A,#24H ; Address Pointer Set = instruction 24H
   MOVX @DPTR,A
   ACALL chk01
```

```
; The display is put into Auto Increment mode, i.e. the module itself will
; increment the address pointer after receiving a data byte
   MOV DPTR,#command
   MOV A,#0B0H ; Data Auto Write Set = instruction 0B0H
   MOVX @DPTR,A
   ACALL chk01


; Write data to LCD module

; First read the end of the data block, LSB is stored in R0, MSB in R1
   MOV DPTR,#enddata
   MOV R0,DPL
   MOV R1,DPH
   MOV DPTR,#block                  ; start of data block
   MOV ACC,#0
loop MOVC A, @A+DPTR                ; read data byte from program memory
   PUSH ACC
; check if the last byte has been sent
   MOV A,R1
   CJNE A,DPH,next
   MOV A,R0
   CJNE A,DPL,next
; end of data block, jump out, first turn Auto Write off
   MOV DPTR,#command
   MOV ACC,#0B2H           ; Auto Mode Reset = instruction 0B2H
   LJMP stop
next POP ACC
   PUSH DPH
   PUSH DPL
   PUSH ACC
   ACALL check2            ; check display status, ready to receive next byte?
   POP ACC

   MOV DPTR,#data          ; address of LCD data register
   MOVX @DPTR,A            ; write byte to display module
   POP DPL
   POP DPH

   INC DPTR                ; points to next byte in data block
   MOV ACC,#0
   SJMP loop

; This subroutine check if the display is ready to receive new data or a new
instruction
chk01
MOV DPTR,#command
MOVX A,@DPTR
ANL A,#3
CJNE A,#3,chk01
RET
; This subroutine check if the display is ready to receive new data in Auto Write
mode
check2

   MOV DPTR,#command
   MOVX A,@DPTR
   ANL A,#8
   CJNE A,#8,check2

   RET
  ORG 200H
block
  DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
   .
;ceci continue ainsi sur de nombreuses lignes, le hasard voulant que le premier et
le dernier octet ne comportent que des 0.
   .
  DB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
enddata
stop              SJMP stop
  END
```

que faire des données, sans oublier de vérifier le registre d'état avant chaque opération d'écriture.
Enfin, nous touchons au but, nous allons réellement écrire les données dans la mémoire graphique de l'afficheur. Le groupe de données commence à l'étiquette « bloc » et le logiciel Delphi l'ajoute au listage. L'étiquette « enddata » indique l'adresse de l'octet qui suit immédiatement le bloc de données. Cette manière de procéder préserve l'indépendance du logiciel assembleur par rapport aux dimensions de l'affichage. Songez également que l'image peut parfaitement être plus grande que l'écran lui-même, puisque la mémoire est plus vaste, malgré que nous n'ayons trouvé nulle part, dans la documentation du LM24014, de combien elle le dépassait ! Si l'on veut ainsi profiter de l'effet d'agrandissement, il faut écrire davantage de lignes horizontales en mémoire. On peut alors faire défiler l'image sur écran en modifiant la *Graphic Home Address* d'un multiple de la *Graphic Area* GA. C'est précisément GA qu'il convient de modifier dans le listage assembleur pour l'adapter à un afficheur de dimensions différentes.
Une remarque, toutefois : il existe des modules d'affichage dont la suite d'adresses n'est pas linéaire. Nous avons ainsi trouvé un module de Toshiba sur lequel l'afficheur, en plein milieu du domaine d'adresses, fait un saut ! Il y a lieu, en pareil cas, d'en tenir compte dans le logiciel assembleur.

Pour finir

Il nous reste à présent à assembler le fichier source à l'aide d'un assembleur pour 8051, par exemple celui du « cours µC-80535 et assembleur » disponible sur la disquette **EPS1811**.
Après traduction en un fichier HEX pour Intel, n'importe quel logiciel de terminal (comme Hyperterminal qui accompagne Windows) permettra de l'envoyer par port sériel vers la monocarte à 80C537. Le port sériel sera configuré comme suit : 9 600 bauds, 8 bits, 1 bit d'arrêt, pas de contrôle de flux. Et finalement, nous allons pouvoir faire démarrer l'application et, dans l'instant, admirer sur écran LCD l'aboutissement de tous nos efforts.

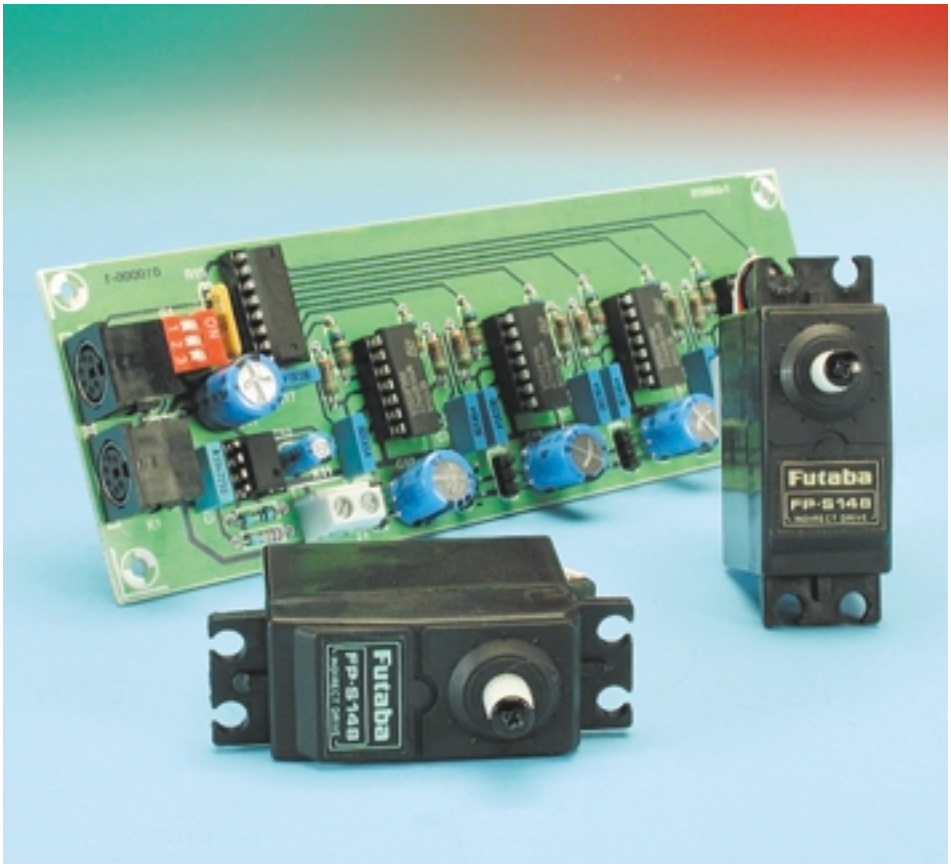
(000134)

Interface I²C pour servos

Pilotage de 8 servo-commandes via I²C

Christian Fuchs

De par leur couple élevé et leur force de blocage importante, les moteurs de servo-commande de modèles réduits télécommandés conviennent parfaitement pour nombre d'autres applications. L'interface décrite ici permet une commande facile de 8 servos par le biais du bus I²C. Un petit programme de démonstration tournant sous Win 95/98 illustre les possibilités du montage et souligne la simplicité de la programmation en Visual BASIC.



La commande simultanée de plusieurs servo-commandes par ordinateur, qu'elle se fasse de façon interactive ou par exécution d'un programme, requiert la totalité de la puissance de traitement de la machine. Il faut, pour lui éviter d'avoir à générer lui-même les impulsions de commande à la chronologie ô combien critique –tâche de plus en plus délicate au fur et à mesure que le nombre de servos concerné augmente– il faut impérativement intercaler une interface entre l'interface de sortie du PC en question et l'électronique des servo-commandes. Le présent montage travaille à l'aide des signaux de bus I²C fournis par une interface I²C tout ce qu'il y a de plus simple (telle **l'interface I²C pour port imprimante** décrite dans le numéro 249, mars 1999, page X10 et suivantes) et peut de son côté générer 8 impulsions de commande de servo indépendantes, ce qui partant revient à permettre la commande de 8 moteurs de servo. De

par l'existence d'une possibilité de paramétrage de l'adresse de circuit intégré (*chip address*), il est même possible de monter en chaîne un maximum de 8 interfaces pour servos, ce qui permet de commander jusqu'à... et oui vous avez bien compté.. 64 servos.

Le schéma de principe

Le synoptique de la **figure 1** illustre le principe de fonctionnement du montage. Un NE555, IC6, monté en multivibrateur astable, fournit, de par le dimensionnement du réseau RC dont il est doté, une impulsion en aiguille d'une longueur de quelques millisecondes toutes les 20 ms environ. Ces impulsions déclenchent 8 bascules monostables (*mono-flop*) intégrés dans 4 temporisateurs doubles du type 556, IC2 à IC5, qui à leur tour produisent des impulsions de 1 ms. Ces impulsions sont appliquées aux entrées de commande des moteurs de servo, ce qui se traduit par leur rotation vers une position en butée. Il est possible, par l'application à l'entrée de commande du temporisateur d'une tension de commande, d'allonger à 2 ms la largeur d'impulsion : les moteurs de servo tournent vers une autre position de butée, décalée dans la majorité des cas de 90 °.

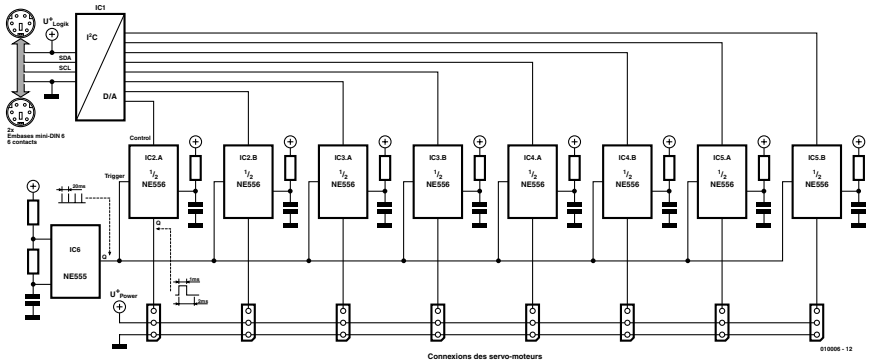


Figure 1. Synoptique de l'interface pour servos.

Cette tension de commande destinée aux 8 monostables est fournie par un TDA8444, un octuple convertisseur numérique/analo-

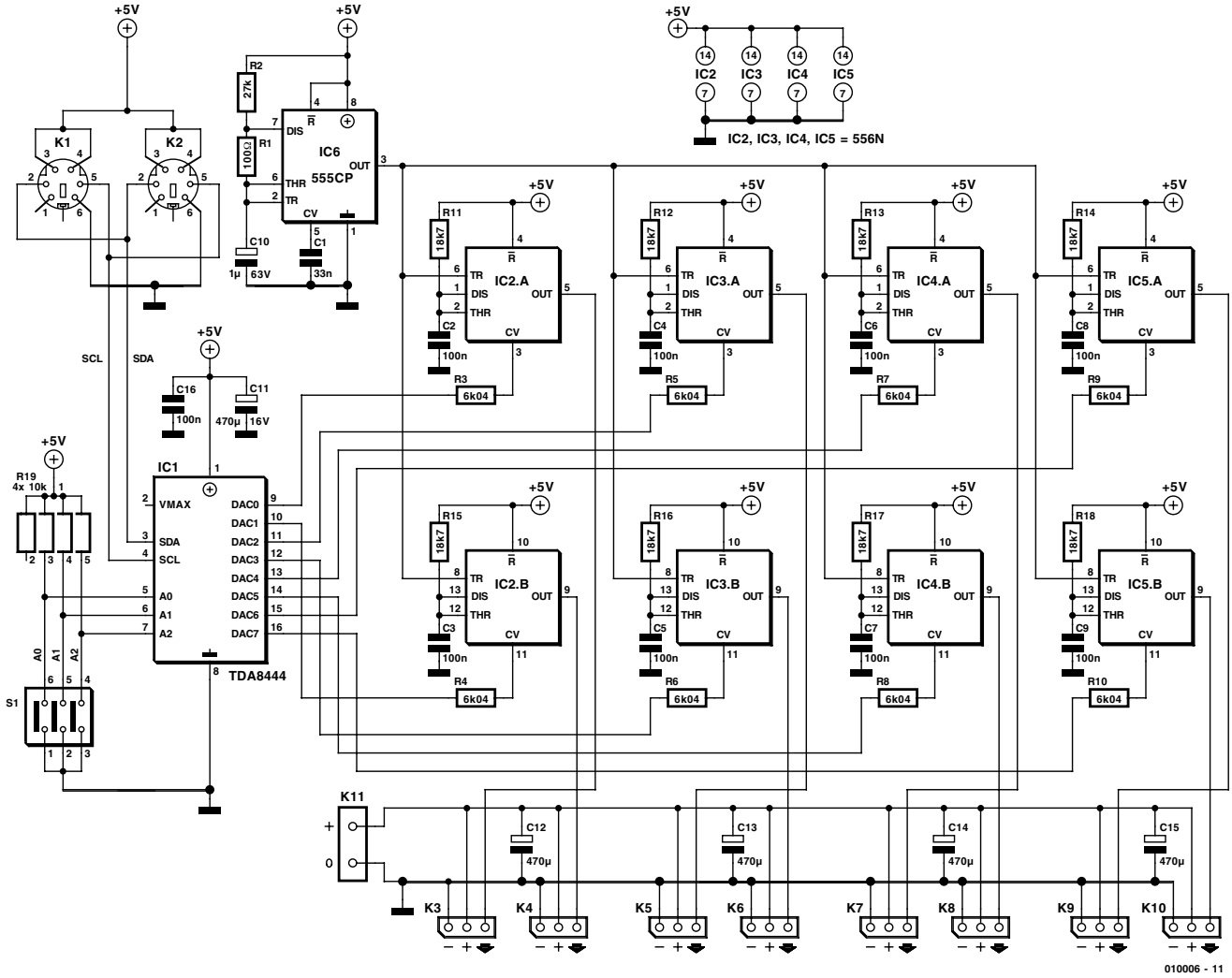


Figure 2. Le pilotage des temporisateurs se fait par le biais des CNA I²C.

gique (CNA) qui possède une interface I²C et se laisse programmer facilement à l'aide du logiciel idoine. Il a été prévu, sachant que les moteurs de servo consomment, en fonction de leur puissance, des courants différents mais importants, et qu'ils produisent, en dépit de la présence d'un filtrage efficace de la tension d'alimentation par le biais de gros condensateurs électrochimiques, un énorme spectre de parasites –ce qui peut, à une occasion ou une autre, se traduire par le plantage du TDA8444–, une alimentation (externe) indépendante pour les moteurs. Le reste de l'électronique (peu gourmande) dérive sa tension d'alimentation du +5 V du PC ou plus exactement de l'interface I²C.

Entrons dans les détails

Le coeur de cette réalisation, dont on retrouve le schéma de l'électronique en **figure 2**, est, on s'en serait douté après la lecture des paragraphes précédents, l'octuple CNA TDA8444 de Philips Semiconductors. Chacun des convertisseurs qu'il intègre donne au signal de sortie une résolution de 6 bits (ce qui revient à 64 pas). Les lignes de bus SCL (*Serial CLock*) et SDA (*Serial DAta*) arrivent, ainsi d'ailleurs que la tension d'alimentation, par le biais des embases mini-DIN K1 et K2 montées en parallèle, à l'interface servo et au TDA8444.

Les instructions I²C convenables permettent la sélection du CNA concerné (0 à 7) et la transmission de la valeur à convertir (pouvant aller de 0 à 63). Nous consacrons un paragraphe spécifique à cet aspect, à savoir la programmation du convertisseur.

Le positionnement des bits 2 à 4 de l'adresse du CNA (0100_A2_A1_A0_0) peut se faire de l'extérieur par la mise des 3 lignes A0 à A2 respectivement soit au niveau bas (masse) soit haut (+5 V ou en l'air). Il devient possible ainsi, grâce à ces différentes adresses, de commander 8 de ces circuits de conversion par le biais d'un seul et même bus. L'entrée VMAX (broche 12) offre la possibilité de réduire la tension de sortie maximale des 8 CNA (le circuit intégré accepte une tension d'alimentation pouvant aller jusqu'à 18 V). Si l'on a opté pour une tension d'alimentation de +5 V cette réduction n'est pas nécessaire de sorte que l'on pourra laisser la dite broche en l'air. Les tensions de sortie des 8 CNA étagées entre 0,26 et 3,12 V disponibles sur les broches 9 à 16 sont transmises, par le biais des résistances R3 à R10, aux entrées de commande (CV) du double temporisateur.

Les NE556 sont câblés en multivibrateurs monostables (*mono-flop*) classiques de sorte que le dimensionnement à 18kΩ7 de R et à 100 nF de C se traduit, dans le cas d'une tension de commande de 0,26 V, par une largeur d'impulsion de 1 ms. Le courant qu'induit une tension de 0,26 à 3,12 V appliquée aux résistances de limitation de 6kΩ04 pilote les monostables de façon telle que la largeur des impulsions de sortie augmente progressivement en 64 pas qui s'étagent entre 1 et 2 ms. Il peut s'avérer nécessaire, en raison de la tolérance des composants, d'avoir à adapter la valeur des résistances de manière à ce que l'augmentation de la largeur d'impulsion présente à chaque fois une rotation de 90 °

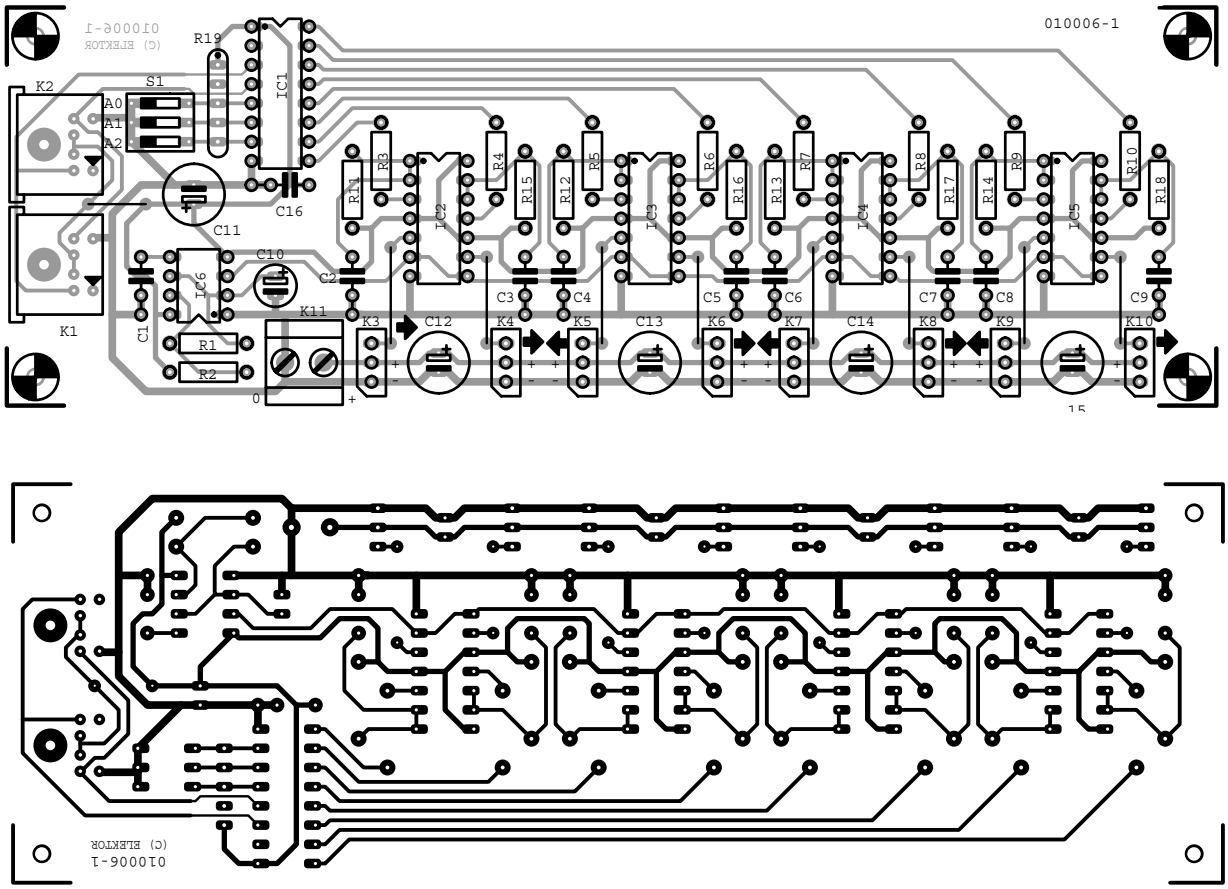


Figure 4. Cette platine simple face qu'il vous faudra réaliser vous-même simplifie très sensiblement la réalisation de ce montage.

très exactement. La génération de l'impulsion se fait par lors de l'arrivée, sur les entrées de déclenchement TR (broches 6 et 8), des impulsions en aiguille produites au rythme de 20 ms. Ces impulsions sont produites, nous le disions plus haut, par le NE555 câblé en multivibrateur astable à rapport cyclique asymétrique et tournant en roue libre (*freewheeling*).
Les 8 signaux de sortie des monostables arrivent (de même d'ailleurs que la tension d'alimentation des servos) chacun à une embase à 3 contacts auxquelles viennent se connecter les servo-commandes. Le brochage des connexions de servo diffère malheureusement d'un fabricant à l'autre, raison pour laquelle nous vous les proposons en **figure 3**. L'utilisation de pistes de bonne largeur et de condensateurs de découplage aident à éliminer les nombreux parasites potentiels.

La réalisation

La réalisation du montage à l'aide de la platine représenté en **figure 4** débutera par la mise en place des composants les moins sensibles tels que les (9) ponts de câblage, les embases et les connecteurs ainsi que les supports pour circuit intégré; on passera ensuite à l'implantation des résistances et des condensateurs. Attention à l'implantation du réseau de résistance R19 : il s'agit, souvenez-vous en, d'un composant comportant une polarité. Il faudra, avant de mettre les différents circuits intégrés dans leur support respectif, s'assurer de la présence, après branchement, par le biais du connecteur K1 ou K2, au montage d'une alimentation externe (!) fournissant 5 V, de la tension d'alimentation aux différentes broches des supports et des embases où elle devrait se trouver. Ce n'est qu'après s'être assuré que tout est OK à ce niveau, que l'on pourra implanter les circuits intégrés dans leurs supports en veillant au respect de leur polarité et connecter l'interface I^2C du PC au système. Il faudra prévoir, si l'on envisage d'utiliser un nombre important de servos, une alimentation externe, sachant que le PC n'est pas en mesure de fournir des courants élevés requis.
Après réapplication de la tension d'alimentation les 8 sorties des CNA devraient présenter une tension de 0 V (ou jusqu'à 0,26 V au maximum); on s'assurera ensuite à l'aide d'un oscilloscope de la présence, sur la sortie de IC6 (sur sa broche 3), des impulsions en aiguille de 1 ms de large. Un moteur de servo au neutre connecté au montage va jusqu'à l'une des butées. On a vérifié ainsi le bon fonctionnement du montage, hormis celui du TDA8444.

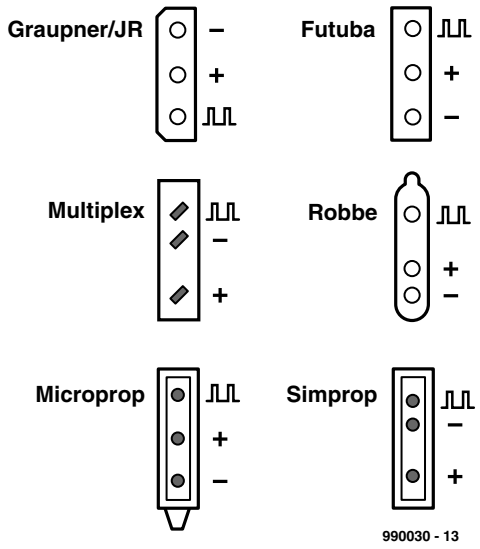


Figure 3. Brochages des connecteurs de différents types de servo-commandes.

Le logiciel

Il est temps maintenant de lancer le programme de test *i2cservo.exe* (un programme 16 bits qui tourne sous Win95/98 mais pas sous Windows NT). Une dll, *Iic.dll*, à placer soit dans le même répertoire que l'application soit dans le répertoire système, assure l'interfaçage indispensable avec le matériel côté ordinateur. Ce programme, qui a également

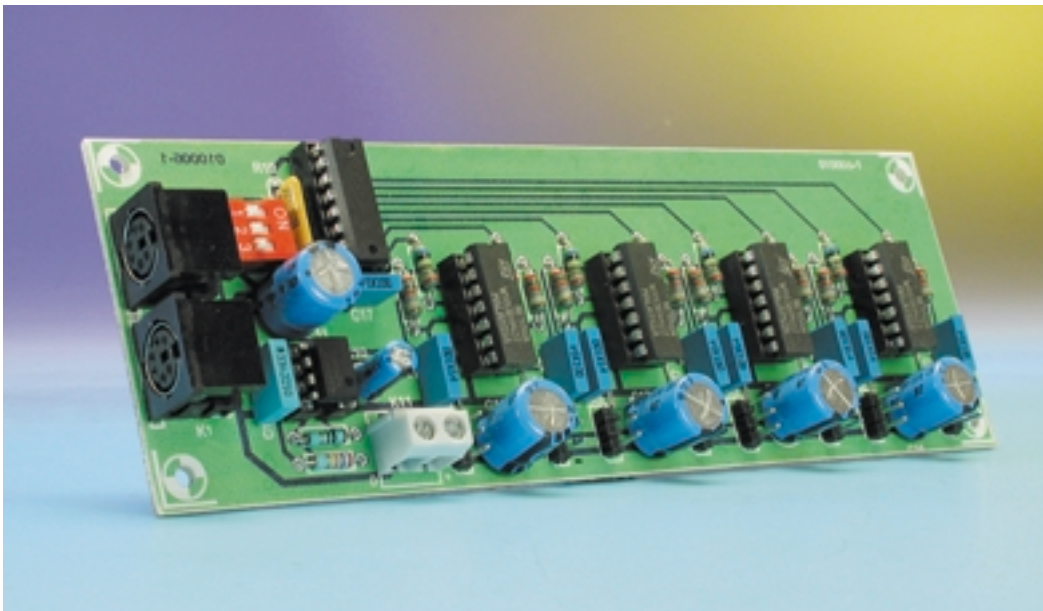
Liste des composants

Résistances :
R1 = 100 Ω
R2 = 27 kΩ
R3 à R10 = 6kΩ04/1%
R11 à R18 = 18kΩ7/1%
R19 = réseau SIL de 4 résistances de 10 kΩ

Condensateurs :
C1 = 33 nF
C2 à C9,C16 = 100 nF
C10 = 1 µF/16 V vertical
C11 à C15 = 470 µF/16 V vertical

Semi-conducteurs :
IC1 = TDA8444 (Philips)
IC2 à IC5 = NE556
IC6 = NE555

Divers
K1,K2 = embase mini-DIN à 6 contacts
K3 à K10 = embase autosécable mâle à 1 rangée de 3 contacts
K11 = bornier encartable à 2 contacts au pas de 5 mm (RM5)



été utilisé pour le montage « **interface I²C pour port imprimante** », est écrit en Visual BASIC 3. Il permet tout d'abord, par le biais d'un point de menu, le paramétrage des interfaces parallèles LTP1 (378_{HEX}) ou LPT2 (278_{HEX}). On peut ensuite définir la sous-adresse du composant I²C (l'adresse 7 se traduisant par la mise à+5 V des lignes A0 à A7, l'adresse 0 étant obtenue par la mise à la masse de ces mêmes lignes, A0 à A7). Les 8 potentiomètres à glissière correspondent aux 8 canaux CNA et partant aux moteurs des servo-commandes. Un déplacement à l'aide de la souris de l'un des organes de commande devrait se traduire par une réaction logique du moteur de servo-commande correspondant.
On trouvera sur le site Internet d'Elektor non seulement le dessin de la platine (au format .pdf) mais également ce programme de démonstration en VB3 sous la forme et du fichier exécutable et du fichier-source, téléchargements gratuits. Il vous sera possible ainsi de modifier le programme en fonction de vos propres besoins. Lors du lancement du programme on a lecture de la sous-adresse 7 (toutes les broches d'adrresse forcées au +5 V) préparamétrée

```
ChipAdress = Val(Text1.Text)
```

et positionnement de l'interface parallèle sur LPT2

```
Lpt = Lpt2.
```

LPT1 et LPT2 sont paramétrés respectivement, par le biais du programme *Modul1.bas*, à &H378 et &H278. L'utilisateur peut ensuite modifier ces 2 paramètres (sous-adresse 7 et port LPT2).

```
SubText1_Change,
```

Messages d'erreur

0	Error 0000	Absence d'erreur
1	Error 0001	Le bus I²C n'est pas libre
2	Error 0002	Non-réception de signal d'acquittement (Acknowledge)
3	Error 0003	Impossibilité de génération de condition d'arrêt
4	Error 0004	Absence de la tension d'alimentation au niveau du bus I²C
5	Error 0005	Interface I²C non connectée
6	Error 0006	Apparition d'une erreur ou problème non défini

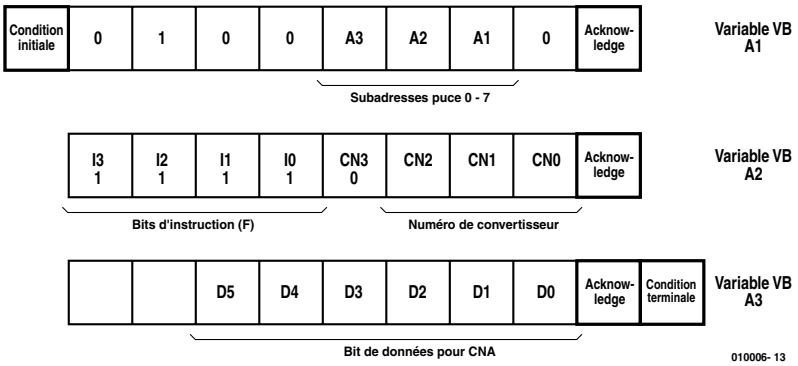


Figure 5. Modèle de programmation du TDA8444.

```
SubParallel1/2_Click).
```

Le programme vérifie ensuite que l'interface I²C est connectée correctement

```
ErrorMsg=TestConnected(LPT)
```

et fournit un message d'erreur décrivant la situation rencontrée. En cours de fonctionnement, on a, après chaque mouvement de l'un des organes de commande, lecture, par l'instruction

```
SubVScrollX_Change()
```

où X vaut de 0 à 7 pour désigner le potentiomètre 1 à 8, en fonction de sa position, d'une valeur allant de 0 à 63 qui est ensuite attribuée à la variable A3.

```
DacWert = VScrollX.Value.
```

La valeur MAX du paramètre VCcroll-Bars est fixée à 63, ce qui élimine toute nécessité de conversion. A2 reçoit, pour ses 3 bits de poids faible, le numéro d'ordre du CNA (0 à 7) qui correspond à l'organe de commande concerné. Le bit 4 contient toujours un 0 comme bit de numéro de CNA additionnel, les 4 bits de poids fort étant eux tous forcés à « 1 » (= 240 = instruction F: toujours écrire dans le même CNA, dans le cas de plusieurs opérations d'écriture successives sans indication de numéro de convertisseur). A1, pour finir, contient, dans les bits 2 à 4, la sous-adresse du circuit, les 4 bits de poids fort contenant eux, l'adresse de circuit fixe de 0100 (64_D).
Les 3 bits A1 à A3 de la **figure 5** constituent le télégramme de données qui sera, suite à un mouvement au niveau de l'organe de commande –qui se traduit lui par la génération des octets constituant ces données–, envoyé au TDA8444 par le biais de l'interface I²C.
Les routines requises à cet effet sont mises à disposition par le fichier *Iic.dll*, les dites fonctions étant définies sous le label *GLOBAL*.
Lors de l'émission on commence par établir, par le biais de l'instruction

```
ErrorMsg = StartCon( LPT)
```

une condition de début I²C, les 3 octets de données étant ensuite transmis par le biais de l'instruction

```
ErrorMsg=Transmit(LPT,A1..A3).
```

Le bus I²C est ensuite libéré par l'envoi d'une condition d'arrêt I²C

```
ErrorMsg = StopCon(LPT).
```

Les différentes erreurs pouvant se produire au cours de ce processus sont attribuées aux 5 variables Er1 à Er5 avant de se traduire, le cas échéant, par l'apparition d'un message d'erreur dans la fenêtre d'état du bas prévue à cet effet. L'encadré donne la correspondance entre les numéros d'erreur et leur source potentielle.
On pourra, pour de plus amples informations concernant le TDA8444, faire un tour sur le site Internet de Philips et entrer l'adresse suivante :

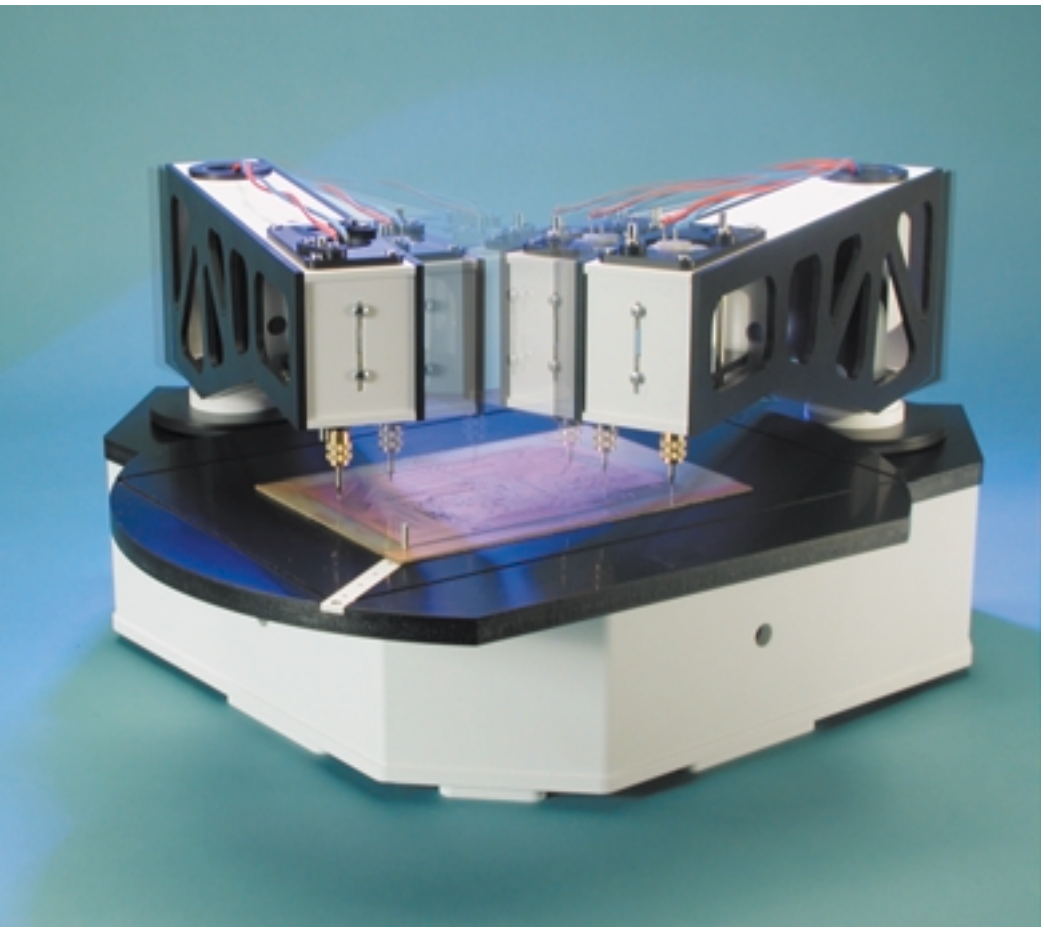
http://www.semiconductors.com/acrobat/datheets/TDA8444 3.pdf).

(010006)

ForTan

Partie 4 : maintenant allons-y !

La dernière partie des instructions de montage permettra de tester toutes les fonctions de ForTan et de mettre la machine au point. Ce n'est qu'alors que la foreuse tangentielle de platine CNC sera prête à faire ses premières armes.



Entre-temps, les premiers ForTan ont été fournis et peut-être même montés. Il est donc plus que temps de terminer leur description. Il ne manque plus, en effet, que le calibrage de la foreuse. Car seul un calibrage permet au logiciel de ForTan de déterminer le zéro et de calculer correctement toutes les coordonnées de forage, de positionner le ou les bras et le plateau et de déplacer les mandrins de haut en bas. Une série de commutateurs de position joue un rôle décisif lors de l'initialisation. Dans cette dernière partie consacrée à ForTan, nous montrerons comment effectuer le calibrage complet des bras et du plateau au moyen du logiciel TanBoTest et comment se servir du logiciel de commande TanBoDrive pour percer des platines.

Commutateurs d'orientation

Le boîtier de la machine comporte un interrupteur de fin de course optique fonctionnant par réflexion placé exactement entre les deux broches tournantes des bras porte-outil. Cet interrupteur de fin de course permet de déterminer la position du plateau tournant et des bras porte-outil 1 et 2. Le détecteur se compose d'une diode luminescente qui projette verticalement un faisceau infrarouge vers le haut et d'un phototransistor tourné vers le haut qui se déclenche lorsqu'un objet réfléchissant renvoie le rayon de la LED (DEL pour Diode Electro-Luminescente en français). Une feuille de chrome faisant office de réflecteur est située sur l'index de la broche du bras de l'outil et sur la face inférieure du plateau tournant.

Cette barrière optique à réflexion fonctionne à la lumière infrarouge non modulée et est donc très vulnérable aux perturbations causées par d'autres sources IR (lumière du jour et lumière artificielle ainsi que, par exemple, un briquet ou le rougeoiement d'une cigarette !!). L'interrupteur de fin de course est certes complètement couvert dans les domaines « circulaires » du plateau tournant, mais la lumière parasite peut s'infiltrer par les 2 côtés « tronqués » en position opposée. Alors pourquoi le plateau tournant n'est-il pas circulaire ? Il y a une autre raison importante que l'économie de maté-

riel : c'est la seule façon d'avoir accès au détecteur pour l'examiner et le nettoyer.
Au cas où la lumière parasite affecterait le détecteur, il suffit que le plateau tournant pivote jusqu'à ce que l'obscurité se fasse, puis effectue encore *un petit angle de rotation* jusqu'à ce que la lumière apparaisse de nouveau. La position de la feuille réfléchissante est déterminée de cette façon. Lors de la rotation suivante en direction opposée, la transition de la clarté à l'obscurité doit avoir lieu exactement au même endroit que la transition contraire. Une fois que le système connaît la position du plateau tournant, il saura aussi désormais où se trouvent les méplats du plateau pouvant laisser passer la lumière parasite. Le logiciel en tient compte.
Il est moins complexe d'initialiser les bras d'outil car le plateau tournant couvre le détecteur dans chaque position, ce qui le protège de la lumière parasite. Le système cherche (et trouve) la position réfléchissante de l'index du bras.
Un problème non trivial se manifeste lors de l'utilisation simultanée de 2 bras. Il existe un danger de collision lorsque les 2 bras tentent simultanément d'atteindre la position de l'interrupteur de fin de course ! Mais ce problème est résolu très élégamment. Le logiciel Windows enregistre la position des 2 bras dans un fichier temporaire chaque fois que le programme se termine. Ce fichier est rechargé lors de l'activation suivante du programme, ce qui évite une collision des 2 bras. Si le programme « se plante » (ce qui est parfois la règle sous Windows), ce fichier ne sera pas disponible lors du prochain lancement. Le programme passe alors en mode de guidage manuel. Une représentation graphique de ForTan apparaît sur l'écran. La souris permet de faire pivoter les bras d'outil sur l'écran dans la position qu'ils ont réellement sur la machine. Pas besoin d'une très haute précision, mais ForTan se fie naturellement à vos indications.
Deux autres micro-interrupteurs de fin de course dans les bras d'outil sont chargés de déterminer les 3 états différents de l'avance de la tête.
État 1 : avance de la tête tout en haut en position de repos. Si la bobine de levage n'est pas sous tension, la force du ressort installé au bas du bras doit pousser le chariot de la tête jusqu'en haut, ce qui ouvre le commutateur BO (BO possède un contact de repos). Si un bras n'est pas disponible, il ne possède pas non plus de BO, l'entrée correspondante du commutateur de fin de course (BO1 à BO4) de la platine est ouverte. Le logiciel peut tester séparément la position supérieure de l'avance de la tête de chaque bras.
L'état 2 décrit la position inférieure du chariot de la tête et est obtenu en interrogeant BU. Tous les BU des différents bras possèdent des contacts de travail et sont montés en parallèle. Cela fonctionne pour la bonne raison qu'une seule tête à la fois peut se trouver en position inférieure. La logique de sécurité GAL empêche de commander simultanément plusieurs chariots de tête. Et si, malgré tous, plusieurs chariots de tête se trouvaient en position inférieure (par suite d'un défaut mécanique), les BO s'en apercevraient.
L'état 3 est situé entre les deux (ni en haut ni en bas). En l'absence de défaut, cela ne peut se produire que lors d'une avance. Si ce cas se produit sans mouvement de levage, le chariot de la tête est vraisemblablement bloqué.

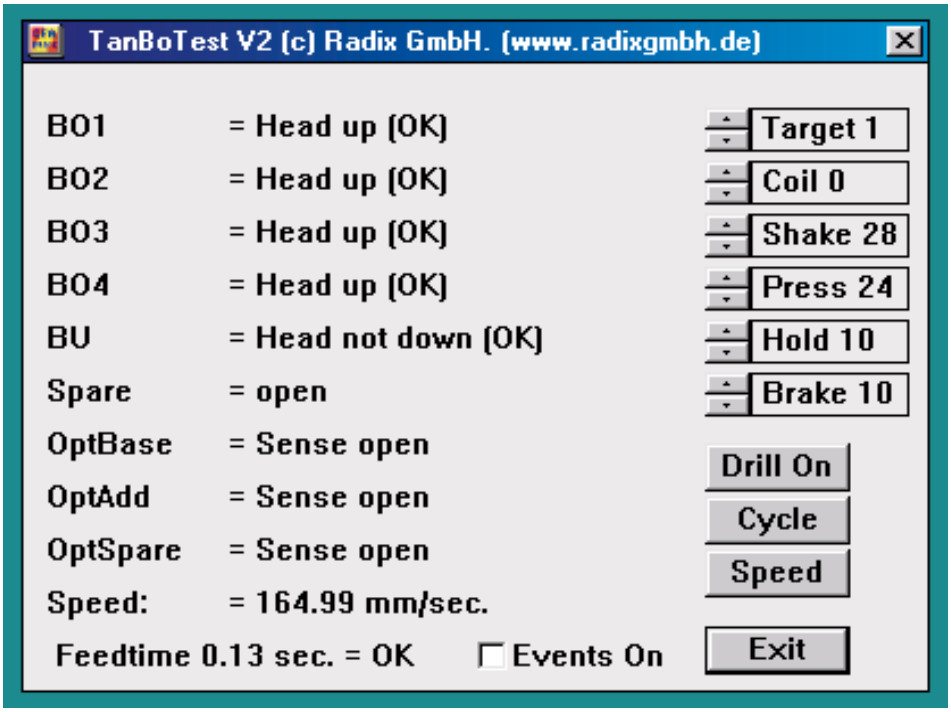


Figure 1. TanBoTest teste les divers micro-contacts et détermine les paramètres de fonctionnement qui sont stockés dans un fichier de configuration.

Le logiciel Windows peut mesurer le temps exact de chaque perçage en chronométrant l'intervalle entre le départ de l'interrupteur supérieur de fin de course et l'arrivée à l'interrupteur inférieur de fin de course. Un bris de foret est identifié par un intervalle trop court, un intervalle trop long peut indiquer l'usure excessive du foret. Si le perçage d'un trou dépasse 5 secondes, l'unité de commande coupe de son propre chef l'avance de la tête et le moteur de la broche.

Les essais d'abord...

Le logiciel de commande « assume » que ForTan est électriquement et mécaniquement en ordre de marche. Les données d'entrée – les données de perçage sont ici en format Excellon – sont préparées et transformées de façon à transmettre les instructions nécessaires au microcontrôleur sous forme d'avances et d'instructions de perçage. On a alors atteint les limites de TanBoDrive.EXE. L'interpolation linéaire pour l'interprétation des fichiers HPGL est déjà incluse –logique, sinon ForTan ne pourrait pas effectuer de perçage– mais l'importation des données HPGL est encore bloquée et ne sera libérée que lorsque le bras à graver annoncé aura été exhaustivement testé. TanBoDrive.EXE peut être comparé au pilote d'une imprimante. Si quelque chose ne fonctionne pas dans la carte du contrôleur ou la mécanique, le programme refuse simplement ses services.
Il est nécessaire de disposer avant tout d'un

logiciel de service et de diagnostic qui teste toutes les fonctions de la carte du contrôleur et des unités connectées, en particulier celles des différents interrupteurs de fin de course qui doivent être câblés et connectés correctement. Et ce n'est que si tous les composants fonctionnent comme prévu et si tous les tests donnent des résultats positifs que l'on pourra commencer à percer.
Le logiciel de test *TanBoTest* répond exactement à ce but. Après le démarrage du logiciel, les états de tous les interrupteurs de fin de course sont affichés dans une boîte de dialogue et actualisés toutes les 300 ms. L'interrupteur optique peut être testé avec le doigt ou une pièce réfléchissante. Il est aussi possible de mettre en marche et d'arrêter toutes les broches de perçage, et d'alimenter les électroaimants de levage à une puissance à déterminer librement afin de contrôler la libre manœuvre des avances et de les optimiser si le besoin s'en faisait sentir.
Le programme, qui se trouve à l'adresse Internet :

www.radixgmbh.de/deutsch/menu_bestellung.html

est gratuit, auto-explicatif, et affiche les états de tous les interrupteurs de fin de course. Ne manquez pas de vérifier les messages de la boîte de dialogue, testez les 3 états du chariot de la tête. Testez l'interrupteur de fin de course optique avec une surface réfléchissante et une source de lumière parasite. Si *Target* se trouve sur 1 et que vous poussez le chariot de la tête vers le bas, BO1 et BU doivent aussi réagir (et pas aux alentours de BO2) si on pousse avec le doigt le chariot de la tête actuellement actif. Déterminez avec le « défileur » (*scroller*) de *Coil* le courant de la bobine nécessaire pour pousser vers le bas le chariot de la tête. Le numérotage des cibles (*targets*) est défini très clairement : Si on considère ForTan dans le sens de la longueur et depuis le haut, le bras gauche est le bras d'outil 1 et le bras droit le bras d'outil 2.
Étant donné que tous les commutateurs jouent un rôle si important, leurs entrées sur la carte du contrôleur devaient être soudées plutôt qu'insérées. Cela demande un peu plus d'efforts mais restera en place. Vous devriez alléger la traction exercée sur les câbles qui sortent des broches rotatives avec un peu de scotch collé sur le PVC noir pour que les barrettes de picots ne soient pas chargées mécaniquement. Il faut toutefois assurer la liberté d'action des câbles.
Récapitulons brièvement toutes les fonctions du logiciel de test :

Le *défileur Target* choisit l'étage final 0 à 3 pour la foreuse (*Drill*), le cycle (*Cycle*) et la bobine (*Coil*).

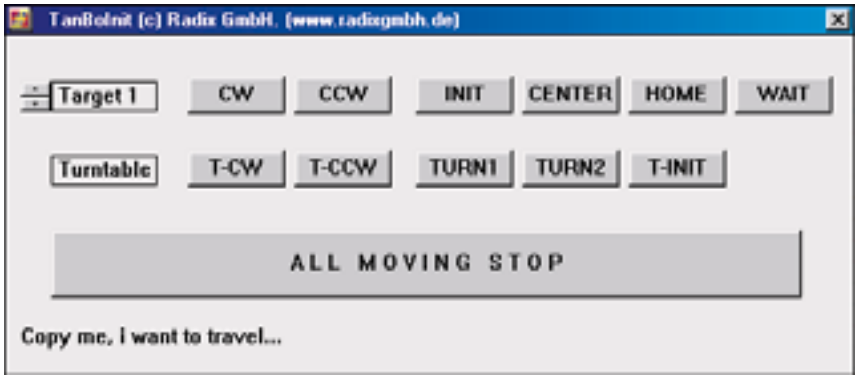


Figure 2. Programme d'initialisation pour le réglage des bras et du plateau rotatif.

Le *défileur Coil* peut être déplacé de 0 à 65 et permet de faire varier le courant de la bobine d'avance de 0 (arrêt) à 65 (100 %) pour l'étage final cible choisi au moyen de *Target*. N'oublions pas que 100 % représente une surcharge pour les bobines de levage d'origine et que celles-ci ne peuvent être utilisées ainsi qu'avec un facteur de service d'environ 30 %. Les bobines peuvent chargées constamment jusqu'à l'échelon 26. Le logiciel provoque l'arrêt si après 5 s le réglage des bobines n'a toujours pas été modifié.

Le bouton *Drill On* met en marche ou arrête l'étage final de la broche de perçage pour la cible choisie.

Les 4 « *défileurs* » (*scroller*) suivants concernent le cycle de perçage (*drill-cycle*). Chaque cible (target) possède ses propres valeurs de pression (*press*) et de freinage (*brake*). Les valeurs de vibration (*shake*) et de maintien (*hold*) sont communes aux 4 cibles. Chaque nouveau clic sur le bouton « Cycle » démarre une séquence de perçage avec le paramétrage défini.

Le *défileur Press* définit la force d'appui de la mèche sur le platine. Ce paramètre joue également, ne serait-ce que partiellement, sur la vitesse de déplacement de la tête –si l'on donne au défileur Press une valeur trop faible l'énergie de la bobine magnétique n'est pas suffisante pour obtenir le déplacement de la tête. Si la valeur choisie est trop importante on court le risque, dans le cas extrême, d'abîmer la mèche lors de son entrée en contact avec le circuit imprimé. La normale est une valeur comprise entre 24 et 40. Il faudra l'adapter au type de mèche utilisée. Si la valeur que vous obtenez avec la tête d'origine est inférieure à 24, cela signifie que vous avez réalisé l'ensemble de mouvement de la tête avec précision et soin. Nos félicitations !

Le *défileur Brake* définit la valeur de freinage qui permet à la tête, lors de sa remontée, d'arriver en douceur en butée haute. Si l'on a choisi une valeur trop élevée, on verra le chariot de tête redescendre brièvement avant qu'il ne reste définitivement en position haute. À l'inverse, en cas de choix d'une valeur trop faible, le micro-rupteur supérieur fait office de butée mécanique haute (ce qui n'est pas sa fonction), situation qu'il faudra impérativement éviter. Il vous faudra trouver la position dans laquelle la tête cesse d'osciller et augmenter la valeur obtenue d'une ou deux unités.

Le *défileur Hold* définit la durée (par pas de 52 ms) pendant laquelle la mèche reste en position basse après qu'elle ait traversé la platine. Un choix de durée trop faible se traduit par des orifices dotés de brames sur le dessous, une durée trop longue constitue une perte de temps pure et simple. La valeur par défaut de 10 (soit 0,52 s) constitue une excellente valeur moyenne qui donne de beaux orifices de tous diamètres.

Le *défileur Shake* est destiné à apporter un remède autonome en cas de problème. Si, par exemple, les roulements à rouleaux sont très encrassés et que le chariot de support de la tête se déplace plus difficilement, la pression induite par le ressort n'est plus suffisante pour

faire remonter le chariot et déclencher le micro-rupteur de butée haute. Il est possible de simuler un problème de ce genre en forçant le chariot légèrement vers le bas (quelques millimètres seulement), de sorte que le micro-rupteur de butée haute indique un BO (*Head not up*). Si, lors du relâchement prudent du chariot, la tête garde la position antérieure, on a simulé le type de panne en question. La valeur de « *Shake* » fait descendre légèrement la tête avant de la laisser remonter, sans freinage, vers le haut. Il faudra bien évidemment veiller à ce que cette opération de descente ne soit pas suffisamment violente au point de faire rebondir la tête non dotée de mèche sur le circuit imprimé.

Le *bouton Cycle* lance l'exécution du cycle de perçage sur la cible définie, la durée de déplacement (diminuée de la valeur de maintien, *Hold*) sert de durée d'alimentation (*feed time*). Si l'exécution du cycle s'est faite sans erreur, on verra apparaître à la suite de la durée le message *OK*, sinon un problème sera indiqué par l'apparition d'un *Error*. Une durée de 4,5 à 5 s associée à un message d'erreur signifie que la mèche n'est jamais arrivée en bas (à simuler par un paramétrage de Press=0). Une durée de l'ordre de 0,5 à 1 s avec un message Error signifie qu'il est impossible d'atteindre la butée haute (à simuler par la position de cas problématique et le paramétrage Shake=0).
Les valeurs déterminées pour le cycle de perçage sont à entrer dans le fichier de configuration TanBoDrive.

Le bouton *Speed* envoie au contrôleur des données à une vitesse croissante jusqu'à ce que l'ordinateur ne puisse plus gérer leur flux et que le FIFO du contrôleur tourne à vide.

La sortie de *Speed mm/s* est basée sur une interpolation de 2 axes, donc par exemple avance du plateau tournant et d'un bras d'outillage à cette vitesse. Les limites mécaniques de la vitesse d'avance sont de l'ordre de 80mm/s. Si le test de vitesse fournit un meilleur résultat, cela signifie qu'on dispose d'un excédent de puissance dont on peut tirer parti pour se déplacer simultanément le long d'axes supplémentaires. Aucun moteur pas à pas ne doit être raccordé lors du test de vitesse !

La case *EventsOn* attribue –lorsqu'elle est cochée – du temps de calcul au système d'exploitation et donc au reste des processus au cours du test de vitesse dans l'espoir d'accélérer les choses. Le programme de commande TanBoDrive.EXE fait appel à un modèle dynamique d'attribution du temps de calcul au système d'exploitation, mais n'hésite pas à bloquer totalement le système si le besoin s'en fait sentir.

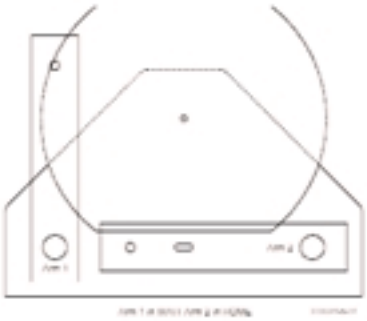
Le logiciel doit connaître au départ l'adresse du port LPT auquel le contrôleur de ForTan est raccordé. Ce réglage E/S se trouve sous Gestionnaire de périphériques/Ports (COM et LPT)/Port imprimante/Ressources et est le plus souvent égal à 0378-037F$_{HEX}$. Ouvrez maintenant une page vierge dans l'éditeur de texte ASCII et copiez le premier de ces nombres (0378 dans l'exemple) à la première ligne. Pressez ensuite 2 fois la touche d'entrée et enregistrez le fichier sous le nom lptaddr.txt dans le répertoire qui contient *TanBoTest*. Ce fichier est utilisé par tous les programmes ForTan et doit se trouver dans le même répertoire que tous les autres programmes.
Une fonction importante du programme de test est la mesure de la vitesse de transfert du flux de données par l'interface Centronics. L'encadré contient des explications supplémentaires.

Adjustment des bras

Après avoir testé le fonctionnement du matériel avec *TanBoTest* et l'avoir corrigé s'il y a lieu, il faut calibrer et ajuster la machine. Le calibrage, en l'occurrence la détermination du point zéro, ne pose aucun problème dans le cas d'une machine linéaire ; mais il est nécessaire ici d'ajuster quelque chose de rond sans commencement ni fin. En outre, le montage de nombreux éléments de ForTan varie considérablement ; par exemple, la broche de perçage peut être déplacée de plusieurs millimètres avant de la visser. Pour

a

b



c

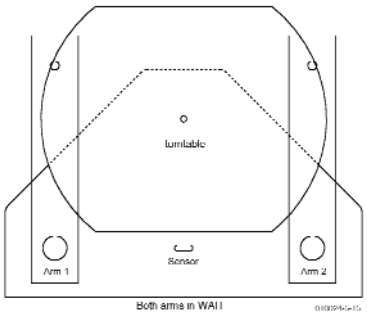


Figure 3. Les positions des bras lors du réglage du plateau rotatif : homewait(a), waithome (b) et waitwait (c).

convertir les coordonnées X/Y il faut connaître exactement la longueur du bras, faute de quoi le positionnement ne se fera jamais avec précision. C'est toutefois possible et, comme promis dans le premier article sur ForTan, sans recourir à aucune méthode de mesure de précision.
Tout ce dont on a besoin pour un ajustement exact est le logiciel *TanBoInit* qui fait aussi appel au fichier *lptaddr.txt*. D'ailleurs, au cas où quelque chose de désagréable se passerait lors de l'ajustement ou si vous pensez que le mouvement d'avance actuel causera une collision, cliquez sur l'arrêt d'urgence *ALL MOVING STOP* et la machine s'arrêtera

immédiatement.
On commence par les bras, puis on passe au plateau tournant. Cela ne fonctionne pas en sens contraire car le plateau tournant ne peut être réglé que lorsque les bras sont déjà calibrés. Vous pouvez faire avancer chaque bras à volonté avec les boutons CW (*clockwise*, dans le sens horaire) et CCW (*counterclockwise*, dans le sens anti-horaire). Choisissez le bras de 1 à 4 qui doit se mouvoir, comme dans *TanBoTest*, au moyen du défileur *Target*. Contrôlez d'abord si tous les bras d'outil pivotent correctement sous l'action des touches CW et CCW. Indiquez aussi les cibles pour lesquelles aucun bras n'est installé et pressez CW et CCW. Dans ce cas, rien ne doit bouger.
L'attribution des numéros des cibles a déjà été contrôlée lors du test de ForTan. Les 4 interrupteurs BO sont placés dans l'ordre 1 à 4. Poussez vers le bas le chariot de la tête ; n'effectuez l'opération que pour un seul bras. Seule la première ligne des BO doit changer. Ce test est extrêmement important car il permet de vérifier le fonctionnement de la détection de collision. Les entraînements possèdent une force incroyable et sont capables sans autre de s'endommager mutuellement !
Le plateau tournant possède ses propres boutons CW et CCW ; le *défileur Target* est toujours attribué à un bras porte-outil. Avant de calibrer un bras, faites pivoter le plateau tournant de façon à intercepter la lumière de la barrière optique à réflexion.
Commencez par calibrer le bras 1 (Target 1). Faites se déplacer le bras 2 (si disponible) jusqu'à l'emplace-

Buffer Underrun?

Le logiciel du contrôleur de ForTan fait passer Windows au temps réel. La sortie des données sous Windows est certes très rapide mais il est incapable de les coordonner exactement dans le temps; c'est pourquoi le contrôleur de ForTan est équipé d'un FiFo qui place les données transmises dans leur séquence temporelle exacte.
À condition que le flux de données de Windows ne s'arrête pas assez longtemps pour que le FiFo se vide complètement, cela fonctionne à la perfection. La taille du FiFo est de 35 positions ; donnons un petit exemple numérique.
Si nous supposons qu'il faille faire pivoter un moteur pas à pas d'un un nombre de tours moyen de 500 pas/s. Cela correspond, dans le cas des moteurs utilisés dans ForTan dont la résolution angulaire de pas est de 1,8°, à un nombre de tours de 2,5 tr/s. La démultiplication 200:1 de la transmission qui suit fait que l'arbre de sortie pivote de 4,5° par seconde. La circonférence de ForTan est de 1510,6 mm, la vitesse d'avance est donc de 18,88 mm.
Quel est le flux de données nécessaire dans ce cas ? Le cahier des charges du port Centronics permet d'envoyer au moins 20 000 octets/s. Un débit de 46 000 octets/s a été mesuré sur un ordinateur d'une génération précédente (Pentium 150 MHz).
Chaque pas du moteur a besoin d'exactement une transmission, si bien que les 500 transmissions nécessaires par seconde forment le pourcentage ridicule de 2,5 % du débit réalisable. La largeur de bande ne constitue donc en aucun cas un problème.
À 500 pas/s, un pas du moteur dure 2 ms. Le FiFo avec ses 35 positions peut donc contenir les données des 70 ms suivantes. Au cours de ces 70 millisecondes, l'ordinateur utilisé traite 10,5 millions de cycles du processeur. Au cas où Windows « perdrait la cadence » et ne remplirait pas le FiFo à temps, la continuité du flux de données sera interrompue.
C'est pourquoi les programmes qui chargent fortement le système ou qui accaparent même toute sa capacité à eux seuls ne sauraient tourner en parallèle avec le logiciel de ForTan.
Comment contrôler la régularité du flux de données ?
Une liaison de commande sur la platine de ForTan indique si le FiFo est vide.
Ce signal parvient à la broche 12 de l'interface Centronics (PaperEmpty) et devient actif quand tous les octets du FiFo ont été traités.
Mais pourquoi un flux continu de données est-il donc si important ? Les moteurs pas à pas avec un taux d'impulsions élevé doivent être amenés lentement au nombre de tours voulu par une fonction de rampe appropriée. Le moment d'inertie du rotor ne peut pas suivre un changement du nombre de tours trop rapide et se bloque. Cet effet est beaucoup plus marqué lors de la mise en marche que lors de l'arrêt mais dépend du moteur et de son étage final. Si le flux d'impulsions est subitement coupé pendant que le moteur tourne encore, le rotor ne s'arrête pas immédiatement mais parcourt une ou plusieurs positions sur sa lancée en raison de son inertie. On ne le remarque pas en l'absence de rétroaction, par exemple sous forme de codeur fixé à l'axe.
Quelle est la solution ? Le mieux est bien entendu de ne pas permettre au flux de données de s'interrompre. Il faut vérifier avant la sortie de chaque impulsion du moteur si la ligne du de FiFo a été activée par le contrôleur et donc si le FiFo a fonctionné à vide. On peut alors – si les caractéristiques de moteur sont connues – décider sur la base du flux de données spécifié initialement si le moteur est bloqué (cela n'arrive en fait qu'à partir d'un nombre de tours relativement élevé) et, si oui, effectuer un nouveau déplacement de référence vers les interrupteurs de fin de course. Si ce comportement – que l'on reconnaît au nombre élevé de déplacements de référence – est trop fréquent, il faut utiliser un ordinateur plus rapide, diminuer la charge de système ou abaisser le flux d'impulsions aux moteurs.
Tout cela n'est pas un grand problème, si ce n'est que les déplacements de référence inutiles sont effectués aux dépens de la vitesse totale moyenne et qu'ils freinent très sensiblement le système. À l'opposé, une réduction de la vitesse du moteur peut augmenter la vitesse moyenne du système en supprimant les déplacements de référence supplémentaires.
Mais il existe une solution encore meilleure : On accroît la taille du FiFo comme on le faisait jadis dans le cas de l'impression désynchronisée (spooler). Ce procédé peut être remis au goût du jour pour ForTan. Une petite RAM statique, de 8 Ko par exemple, permet d'augmenter la période intermédiaire du FiFo à 8,25 secondes lorsque la fréquence est, comme plus haut, 500 pas/s.
Ce FiFo supplémentaire constitue en fait un dispositif auxiliaire externe ; il ne suffit pas de changer de contrôleur. Les contrôleurs avec RAM de grande capacité intégrée sont en effet considérablement plus onéreux. Le module peut être alimenté par la broche OptSpare (DC 5V). Il serait aussi possible de concevoir un petit module sur carte enfichable dans l'embase du contrôleur et qui jouerait le rôle d'une sorte d'antémémoire.
Si quelque chose se produit dans cette direction, vous serez la première personne à l'apprendre par Elektor ou sur le site de ForTan.

ment approximatif de sa position d'attente (WAIT). Une courte cheville métallique brillante longue de 20 mm est restée de l'assortiment de matériel qui comportait la pince de serrage en laiton. C'est le seul instrument de mesure nécessaire. Enfoncez cette cheville aussi profondément que possible mais sans forcer dans le trou de l'arbre d'aluminium au centre du plateau tournant. Ce trou a été usiné avec la plus haute précision : on devrait entendre un beau « plop » lorsqu'on retire la cheville.

Placez le plus grand adaptateur dans la pince de serrage du bras 1 et serrez sans excès l'écrou par dessus. Desserrez les 8 vis des supports de palier de l'avance de la tête. Il s'agit de 8 pièces ovales dans lesquelles se trouvent les arbres de guidage de 4 mm du chariot de la tête. Ne dévissez pas les vis, mais desserrez-les jusqu'à ce que le chariot de la tête puisse être déplacé latéralement.

Placez le bras 1 au centre du plateau tournant en vous servant des touches CW et CCW jusqu'à ce que vous puissiez appuyer la pince de serrage exactement sur la cheville. Le jeu du chariot de la tête libre devrait être suffisant pour cela, les paliers du bras ne doivent en aucun cas être pressés de force dans la position correcte. La précision est améliorée si la dernière commande donnée est toujours CCW.

Poussez maintenant tout en bas le chariot de la tête et serrez l'écrou de la broche. L'ensemble du chariot de la tête est fixé verticalement par le tourillon d'acier au centre de la table ; cela oblige les paliers à prendre une position déterminée. Tirer alors avec une prudence particulière les paliers inférieurs, puis les paliers supérieurs, d'abord légèrement puis fermement, ce qui permet d'ouvrir tout à fait l'écrou de la broche. Le traîneau glisse maintenant (on l'espère) avec douceur vers le haut. Moins il y a de friction et plus la vitesse de perçage de ForTan sera élevée !

On a fait d'une pierre deux coups : le chariot de la tête est parfaitement vertical et la longueur du bras est réglée exactement sur les cotes du but. Nous avons donc déjà la moitié des coordonnées polaires : la longueur. Le logiciel calcule l'angle manquant. Pressez le bouton *INIT* pour le bras qui vient d'être ajusté. Le bras se dirige alors automatiquement vers l'extérieur jusqu'à ce que son masque de l'index passe sur le détecteur optique dans la table. Le logiciel connaît à présent exactement le nombre d'unités d'angle séparant l'interrupteur de fin de course du centre. Le centre représente un angle de 45°.

Contrôlez encore une fois la position centrale, retirez le tourillon du trou central et placez-le directement dans la pince de serrage. Cliquez sur le bouton *CENTER* du logiciel pour que le bras se positionne au centre. Si on appuie sur la broche, la cheville doit pénétrer exactement dans le trou (il faut ensuite absolument desserrer la pince de serrage, laisser remonter la tête, déplacer le bras et ensuite retirer la cheville du trou !!).

Si le positionnement n'est pas parfait en visant directement le but, on peut ajuster l'angle du bras avec les touches CW et CCW ; la dernière commande devait être ici aussi toujours CCW. Presser le bouton *INIT* lors de chaque ajustement supplémentaire pour sauvegarder les nouvelles valeurs. Le bras retourne immédiatement au détecteur lorsqu'on relâche le bouton. Appuyez sur *HOME* si la position centrale est enfin atteinte à votre satisfaction. Target 1 se place alors en position d'origine (*Home*). Procéder de même avec un deuxième bras, mais retirer d'abord le bras 1 de la zone dangereuse et cliquer sur Wait. Conservez soigneusement le tourillon après cet ajustage pour le cas où vous effectueriez de nouveau un ajustage après une modification ou une extension.

Ajustement du plateau tournant

Si vous avez ajusté deux bras, le bras 2 se trouve maintenant au-dessus du détecteur et le bras 1 en position d'attente (*Wait*). Pressez aussi la position *WAIT* pour le bras 2 de sorte que les deux bras se trouvent côte à côte verticalement et parallèlement. Cliquez aussi sur *WAIT* si vous n'avez ajusté qu'un bras et qu'il se trouve en position *HOME*. Pour ajuster le plateau tournant, placez en position 1 de la barre de repérage – dans le trou situé le plus à l'extérieur au bord du plateau – la cheville de repérage d'environ 8 mm contenue dans le sac de la pince de serrage, placet Target à 1 et pressez *TURN1*. Le bras 1 se déplace alors légèrement de sa position d'attente en direction du plateau tournant.

Faites alors pivoter le plateau tournant avec *TCCW* (PAS *TCW*!) de façon à ce que la cheville parvienne à la barre de repérage sous la pince de serrage. Effectuez un ajustement fin à cet endroit avec *TCCW* ainsi qu'avec *TCW* de telle sorte que la cheville pénètre dans la pince de serrage quand on appuie sur le chariot de la tête. La dernière commande d'ajustement devrait toujours être TCCW. Lorsque vous êtes satisfait du positionnement, appuyez sur *T-INIT*, et puis sur le bouton *TURN 2*. Le plateau tournant pivote d'environ 140 ° dans le sens inverse des aiguilles d'une montre. Le bras de l'outil se déplace en synchronisme avec le plateau jusqu'à ce que tous deux soient arrivés à la deuxième position possible de l'intersection des 2 cercles. Les bases mathématiques du positionnement de ForTan font l'objet d'une animation Flash sous

www.radixgmbh.de/deutsch/menu_link.html.

Lorsque le plateau tournant et le bras porte-outil s'immobilisent, la goupille cylindrique devrait glisser de nouveau facilement dans la pince de serrage et cela sans écraser d'autre pièce lorsqu'on appuie sur le chariot de la tête. Une fois que le calibrage des deux points du plateau tournant est terminé, appuyez de nouveau sur *T-INIT* ; le plateau tournant se déplace jusqu'à ce que l'interrogation des interrupteurs optiques de fin de course ait lieu et se place sur *HOME*, puis le bras de l'outil se place sur *WAIT*.

Répéter le processus de calibrage avec le second bras s'il est aussi installé. Ce n'est certes pas nécessaire du point de vue mathématique, mais les tolérances seront ainsi mieux compensées.

Le fichier *TanBo.def* écrit sur le disque dur lorsqu'on quitte le programme en cliquant sur la croix en haut à droite dans la barre du titre du programme contient tous les paramètres mesurés et est utilisé par tous les autres programmes de commande. *TanBoDrive* ne peut démarrer en l'absence du fichier *TanBo.def*.

Une percee !

Maintenant que la machine est calibrée, il est possible de percer la première carte de circuits imprimés. Il faut (hormis bien entendu la platine gravée mais pas encore percée) un fichier de perçage en format Excellon. Presque tous les programmes de conception de platines sont en mesure de livrer un fichier de perçage dans ce format. Le fichier Excellon contient (entre autre) les coordonnées cartésiennes X et Y de tous les trous à percer

figurant dans le tracé. Le fichier contient encore quelques instructions de commande et (éventuellement dans un deuxième fichier) les informations sur le diamètre de perçage en fonction des coordonnées de chaque trou. Ce fichier peut être ouvert, visualisé et modifié avec un éditeur ASCII normal.
Le bouton *IMPORT* de *TanBoDrive* ouvre une boîte standard de sélection de fichier Windows et attend que le fichier Excellon soit choisi. Les 2 points du système de référence sont ensuite cherchés dans le fichier. Leur diamètre ne doit être utilisé pour aucun perçage ! TanBoDrive cherche ces 2 coordonnées qui sont utilisées comme points de référence, mais ne sont naturellement pas percées. Une fois que ces points sont identifiés, il faut indiquer dans quel trou de référence placer la cheville fixe. À cet effet, l'image de perçage est affichée en réduction dans la boîte de dialogue.
Le contrôle de la planéité du plateau tournant fait aussi partie des réglages de la machine. Car plus le plateau tournant est plan et plus faible est la quantité de « matériau sacrifié » à placer entre la platine et le plateau tournant. Le plateau tournant est fixé par une flasque intermédiaire généreusement dimensionnée, un disque d'acier, à l'arbre tournant au moyen de 3 vis M6 bien accessibles du haut. Si vous faites pivoter le plateau tournant, vous pouvez vérifier s'il présente des variations de hauteur. Marquez l'emplacement le plus bas du plateau tournant et démontez-le. Placez à l'endroit marqué un petit bout de papier imprégné d'huile de table sur la flasque. La raison de cet huilage : le papier devient imperméable, de sorte que l'humidité ambiante ne peut pas en modifier l'épaisseur. Revissez le plateau sur la flasque en serrant bien les vis et effectuez un nouvel essai. Après 2 ou 3 essais au maximum, le plateau tournant ne présente plus aucune variation de hauteur mesurable.
Le matériau sacrifié – un carton mince ou une feuille de papier – est nécessaire car les extrémités des forets sont en général coniques et non pas cylindriques. Le carton doit donc être au moins aussi épais que l'extrémité du foret. L'avance maxima, donc la position la plus basse du foret, peut être ajustée au moyen d'une vis moletée qui se trouve à l'avant du bras de l'outil.
Placez maintenant la carte de circuits imprimés gravée sur le plateau tournant pendant que vous choisissez un des 12 trous du système de repérage. Placez la platine sur la courte cheville métallique et déplacez la cheville mobile pour qu'elle entre dans le deuxième trou de la platine. Poussez légèrement la partie mobile vers l'extérieur pour encore mieux centrer la carte. Si la platine est très striée, il est préférable d'en fixer les coins avec un peu de scotch.

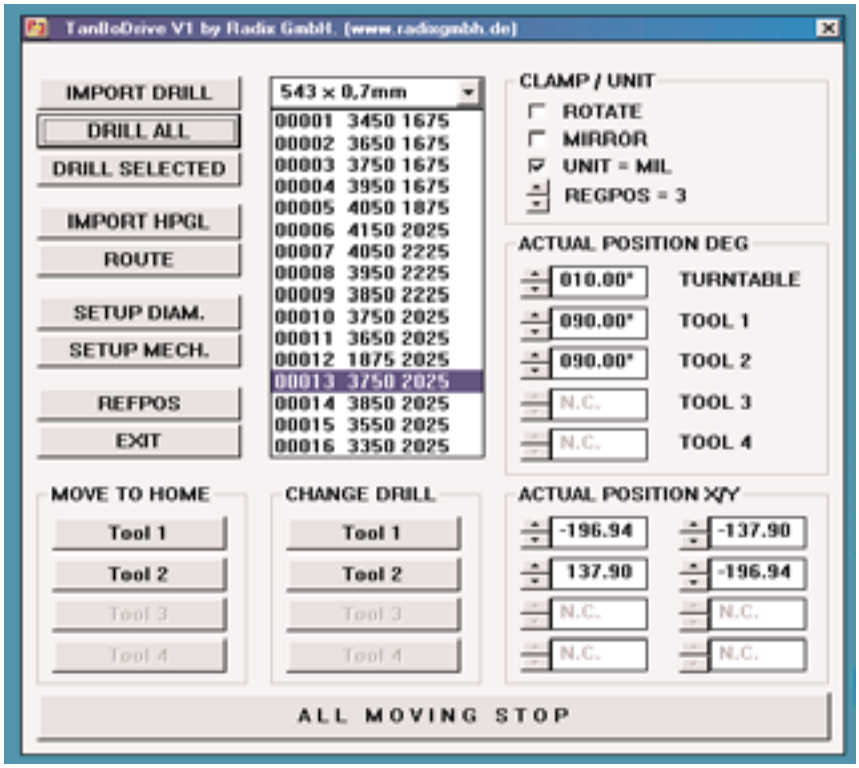


Figure 4. Après avoir importé un fichier Excellon dans le programme d'exécution TanBoDrive, et positionné la platine correctement on peut y aller !

Choisissez maintenant par logiciel le numéro de la cheville de repérage à laquelle la platine est accrochée en vous servant du *défileur FixPoint*. Les chevilles sont numérotées consécutivement de 1 à 12 de l'extérieur (rayon maximum) à l'intérieur. Vous n'avez toutefois pas besoin de les compter car chaque clic sur le *défileur* fait apparaître la représentation du perçage appropriée pour le trou de la cheville affiché. On peut donc voir d'un coup d'œil si la position est correcte.
Il ne reste en principe que 2 sources d'erreur : la carte de circuits imprimés est inversée à 180°. Le trou au point fixe représenté dans la boîte de dialogue se trouve dans le dispositif de déplacement et réciproquement. Cliquer dans la case à cocher *ROTATED* pour positionner correctement la platine sur l'écran.
Un nombre restreint de programmes inversent l'image de perçage (comme les pistes conductrices dans les sérigraphies de montage des platines Elektor). Mais comme la perceuse ne s'y retrouve pas sans les pistes de cuivre, il faut cliquer sur la case à cocher *REFLECTED*. Et la situation est déjà corrigée.
Il ne reste plus qu'à presser sur Start. ForTan place le premier bras de l'outil en position de changement de foret et indique le diamètre désiré dans une boîte d'informations. Placez le foret approprié dans la pince de serrage et confirmez par OK. ForTan se met alors au travail et perce toutes les coordonnées avec la première taille de foret. Le deuxième bras, s'il existe, se place pendant ce temps en position de changement de foret.
Un certain nombre d'interrogations apparaissent ; elles dépendent du nombre de dimensions différentes de perçage ou si plusieurs cartes de circuits imprimés identiques doivent être percées. Il est en fait inutile d'entrer dans les détails : le logiciel est très convivial et va pour ainsi dire presque de soi. En cas de pépin ou de problème, on éprouve le besoin de comparer ses expériences avec d'autres ForTanistes ou simplement d'insulter l'inventeur de ForTan. La page d'accueil de la société Radix est parfaitement conçue pour cela.

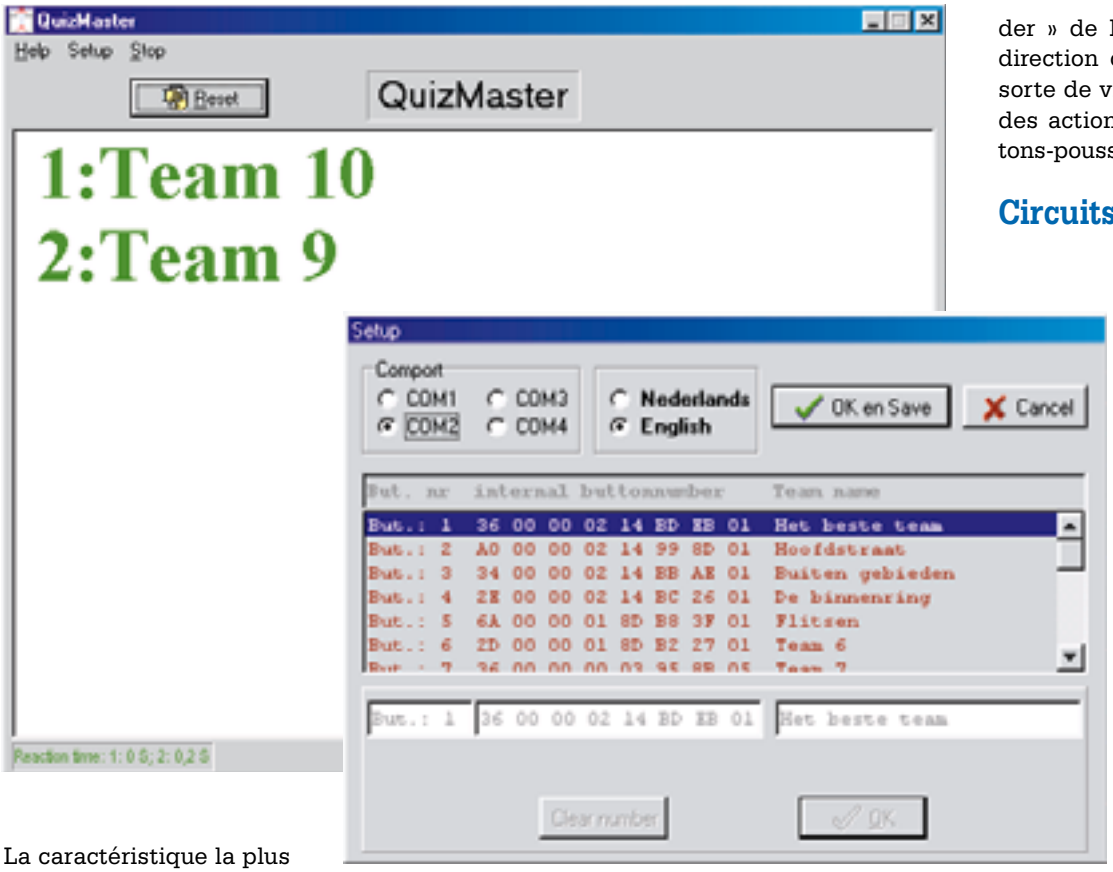
(010024-5)

QuizMaster

En version mono-filaire

projet : Thomas Rudolphi

Un rien de matériel, un programme à 3 sous tournant sous Windows et un PC, il n'en faut pas plus pour déterminer quel bouton, parmi un maximum de 20, a été actionné en premier. Le PC visualise sur son écran l'ordre exact des actions des différents participants. Un arbitre idéal pour les quiz et autres jeux interactifs !



La caractéristique la plus frappante de ce QuizMaster est sa simplicité. En effet, l'ensemble se résume en fait à une minuscule interface, quelques boutons-poussoirs et un programme Windows. Cette interface de 7 composants seulement est reliée au port COM (sériel) du PC.

Les boutons-poussoirs prévus pour les participants au quiz (20 au maximum) sont dotés d'un composant dit « *1-Wire multidrop* » et sont connectés en parallèle à l'interface. Le programme écrit à l'aide du « C++ Builder » de Borland se charge de la direction de l'ensemble et fait en sorte de visualiser à l'écran l'ordre des actions sur les différents boutons-poussoirs.

Circuits « 1-Wire »

L'âme du fonctionnement de ce montage est le circuit « 1 Wire » dont est doté chacun des boutons-poussoirs. Nous avons eu récemment l'occasion de publier plusieurs montages utilisant ce composant, dont le « **OneWireSpy** » du numéro 264 (juin 2000) et le « e-**Key** » du numéro 268 (novembre 2000). Donnons-en, à l'intention des lecteurs occasionnels de ce magazine une description succincte.

Cela fait déjà quelques années que Dallas Semiconductor propose une série de composants qu'il est possible, au travers d'un bus mono-filaire (d'où le « 1-Wire » que l'on retrouve partout), de commander, lire et programmer par

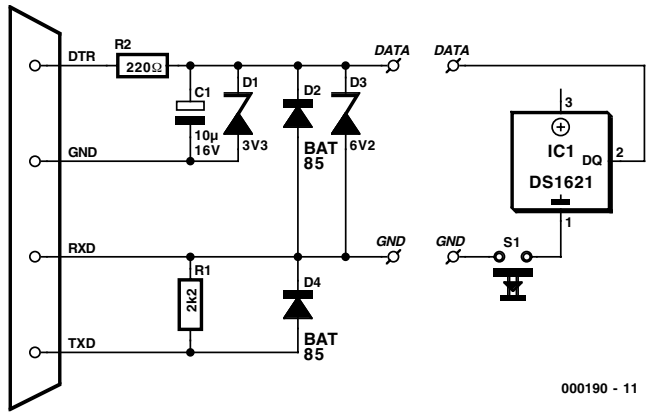


Figure 1. Le matériel se résume en fait à une interface pour le port RS-232 et à un bouton-poussoir avec composant 1-Wire.

le biais d'un maître, un microcontrôleur par exemple. Ces composants « 1-Wire » sont destinés principalement aux tâches de mesure de température, de gestion d'accumulateurs et de capture de données. Dans le cadre du protocole monofilaire un multiplexeur intégré « démêle » les signaux entrants et fait en sorte que le circuit intégré concerné reçoive les signaux qui lui sont destinés. De par le circuit de commande (driver) de sortie en drain ouvert dont sont dotés les composants « 1-Wire » il est possible non seulement de connecter plusieurs de ces composants en fonction ET-câblée (*wired AND*) au même bus, mais encore de les relier à une électronique « classique ».

Bien que les possibilités offertes par les composants « 1-Wire » soient, à strictement parler, bien plus étoffées que ce que requiert la présente application, le DS2401 au prix (relativement) abordable convient parfaitement à l'utilisation que nous en envisageons ici. La meilleure façon de décrire ce type de composant est de le « traiter » de circuit intégré d'identification sans fonction spéciale additionnelle servant uniquement de « numéro d'identification électronique ». Le circuit intégré intègre une ROM 64 bits gravée au laser comportant un numéro de série unique à 48 bits, un contrôle CRC (*Code Redundancy Check*) à 8 bits et un code de famille lui aussi sur 8 bits. Il est possible partant, lors d'une lecture, de le reconnaître et de l'identifier immédiatement, ce qui est très exactement la fonction dont nous avons besoin ici.

Tous les composants « 1-Wire » offrent cette possibilité d'identification et conviennent de ce fait tous à une utilisation dans le cadre de cette application. La disponibilité du DS1820, un capteur de température, est meilleure que celle du DS2401, mais coûte 2 fois plus cher que ce dernier, sans même mentionner qu'il serait dommage de ne pas utiliser sa fonction spécifique (de mesure de température).

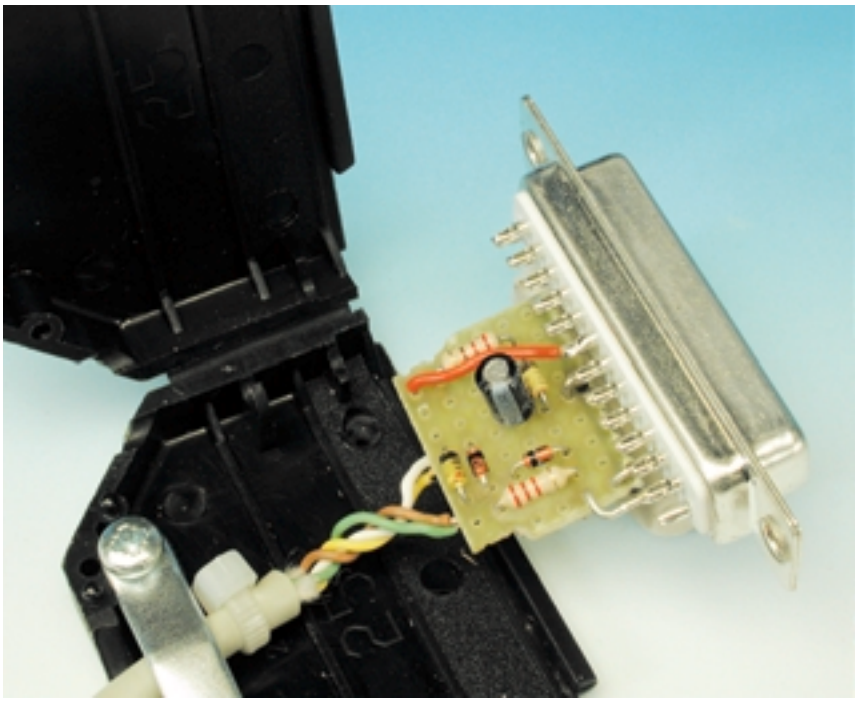


Figure 2. L'ensemble de l'interface pourra être intégrée dans un connecteur RS-232.

L'électronique

La **figure 1** donne le schéma de l'ensemble de l'électronique mise en oeuvre ici. Nous découvrons, sur la gauche, l'interface PC, les boutons-poussoirs dotés des composants « 1-Wire » se trouvant sur la droite du schéma. Comme nous le disions, le QuizMaster accepte un maximum de 20 de ces composants pris en parallèle et connectés au bus « 1-Wire » par le biais des lignes DATA et GND. Dès que l'on a une action sur le bouton-poussoir S1 le composant « 1-Wire » correspondant, IC1 dans le cas présent, est relié au bus de sorte que le PC peut l'identifier.

En électronique, la simplicité est un atout important. Notre interface se résume en fait à quelques diodes (zener ou standard) associées à une paire de résistances et à un condensateur, sachant que l'unité de commande ne comporte pas d'autre composant que le bouton-poussoir et le circuit intégré.

L'alimentation de l'interface se fait depuis le PC par le biais du connecteur RS-232. La résistance R2 et le condensateur électrochimique C1 assurent une limitation de la tension d'alimentation et son filtrage. Comme, d'autre part, les composants « 1-Wire » ne supportent que quelques milliampères, la résistance de forçage au niveau haut (*pull up*) R1 limite le courant à de l'ordre de 6 mA.

La longueur maximale de câble entre l'interface et les contacts du bouton-poussoir ne doit pas dépasser 30 mètres.

Quelques soudures

La simplicité de l'interface est telle qu'il ne vaut pas la peine de la doter de son propre boîtier. La solution la plus pratique consiste à effectuer un montage « en l'air » voir d'utiliser

un petit morceau de platine d'expérimentation à pastilles que l'on glissera dans le capuchon d'un connecteur sub-D. Comme le montre, en **figure 2**, la photo de l'un de nos prototypes, ce type de connecteur offre suffisamment de place à cet effet. Dans le cas d'un connecteur sub-D à 25 contacts, les lignes DTR, GND, RXD et TXD se trouvent, respectivement, aux contacts 20, 7, 3 et 2. Dans le cas d'un connecteur à 9 contacts, ces lignes sont les broches 4, 5, 2 et 3 respectivement.
Comme nous pensions qu'il serait agréable, pour le participant (fébrile) à un quiz, de disposer d'un bouton-poussoir tombant parfaitement sous la main, c'est bien le cas de le dire, nous avons opté pour un bouton de sonnette de porte royalement dimensionné que nous avons, pour cette application spécifique, vissé sur un morceau de bois. Le DS2401 pourra y être fixé à l'aide de quelques gouttes de colle. La photo de la **figure 3** montre comment nous nous y sommes pris dans le cas de ce prototype particulier. La ligne de masse (GND) du bus « 1-Wire » pourra être relié directement à l'un des pôles du bouton-poussoir. Le DS2410 est pris en série dans la ligne DATA.
Les boutons-poussoirs des différents candidats ainsi préparés sont tous montés en parallèle sur le bus « 1-Wire ». La solution la plus pratique pour ce faire est d'utiliser des connecteurs pour lignes de téléphone de type RJ-11 (6 pôles dont 4 contacts connectés), composants que l'on trouve dans n'importe quel magasin de bricolage tant soit peu correctement achalandé. Une pince coupante spéciale permet de monter ces connecteurs sur du câble de téléphone plat. Il est possible, par le biais d'adaptateurs à 3 embases femelles, d'interconnecter les boutons-poussoirs pour les brancher. Le brochage des contacts est le suivant : broche 1 = NC (**N**on **C**onnecté), broche 2 = masse, broche 3 = data, broche 4 = data, broche 5 = masse et broche 6 = NC. Ce doublement de certains des contacts peut paraître superflu, mais il évite de mauvaises surprises au niveau d'un croisement à l'intérieur des adaptateurs triples.

Le programme

Comme nous le disions plus haut, le logiciel écrit avec le « C++ Builder » de Borland tourne sous Windows 95/98. Le programme ainsi que son code-source sont disponibles sur disquette (**EPS000190-11**) auprès des adresses habituelles mais peut également être télédéchargé gratuitement sur le site Internet d'Elektor. En 3 mots, une description succincte du fonctionnement du programme :

Figure 3. Il y a suffisamment de place dans ce bouton-poussoir pour y placer le DS2401.

On procède, dans un *thread*, à une remise à zéro (reset) du bus « 1-Wire » (un signal au niveau bas pendant 480 µs suivi d'une période d'écoute au niveau haut de 480 µs elle aussi). En cas d'action sur l'un des boutons-poussoirs le circuit intégré dont il est doté va réagir. On prend ensuite en compte l'identificateur 64 bits interne pour reconnaître le bouton-poussoir ayant été actionné. Dans le cadre d'un fichier *init*, ce numéro est couplé à un candidat (ou à une équipe de candidats), le logiciel affichant ensuite à l'écran la liste de la correspondance qui en résulte.

Initialisation

Il va falloir, avant de pouvoir utiliser le QuizMaster, définir un certain nombre de paramètres. Il faut, tout d'abord, définir le port COM sériel utilisé. Ce paramétrage pourra se faire dès l'écran de bienvenue.
On pourra, une fois que l'interface est connectée au PC et que le port est paramétré correctement, choisir l'option « Setup » du menu. Un tableau représente les 20 boutons-poussoirs. Il faudra, pour chacun de ces boutons-poussoirs, connaître l'identificateur 64 bits interne et le nom de l'équipe (ou celui du candidat).

– La saisie du nom de l'équipe se fait par un double clic sur le bouton-poussoir de la liste correspondant. Cette action active l'écran d'édition par le biais duquel on pourra entrer, dans le dernier champ, le nom requis.
– L'identificateur 64 bits requiert un apprentissage. Lors d'une action sur le bouton-poussoir requis on voit apparaître le dit numéro dans le champ correspondant. Il ne reste plus ensuite qu'à cliquer sur le bouton OK.

Il faudra reprendre la procédure décrite ci-dessus pour chacun des boutons-poussoirs concernés. Une fois que cette opération générale est terminée on clique sur les boutons « OK et Save ». Les données ainsi saisies sont stockées dans le fichier de configuration, *config.txt*, un fichier ASCII. Lors de son lancement, le programme prendra en compte les données qui s'y trouvent. Le QuizMaster est fin prêt à départager les candidats du quiz et ce à la fraction de seconde près !

(000190)

Accumulateurs Lithium-ion

Techniques et circuits de charge

Gregor Kleine, ingénieur diplômé

Malgré ses prix encore élevés, la technologie la plus récente, celle des accumulateurs Lithium-ion, s'est imposée assez rapidement dans les applications qui requièrent la densité d'énergie la plus élevée. L'amélioration de l'intensité maximale admissible et la technique de charge moderne décrites dans cet article y ont contribué.

Il n'existe toujours pas d'accumulateur idéal. L'accumulateur Cadmium-Nickel (CdNi, NiCd dans les pays anglo-saxons et germaniques) bien connu reste le meilleur pour ce qui est de l'intensité maximale admissible la plus élevée, et l'effet de mémoire si exaspérant fait indubitablement partie du passé (cf. texte encadré). Cet accumulateur pour appareils est en outre le moins coûteux, mais le caractère problématique des substances nocives associées au cadmium qui fait partie des métaux lourds le poussent lentement mais sûrement sur une voie de garage. Les accumulateurs nickel hydrure métallique (NiMH), d'une conception similaire, mais plus riches en énergie et dépourvus de métaux lourds, remplacent de plus en plus l'accumulateur CdNi. Leur intensité maximale admissible s'est certes sensiblement améliorée, mais n'égale cependant pas encore tout à fait celle des accumulateurs CdNi. La capacité des accumulateurs NiMH a pratiquement doublé depuis leur introduction sur le marché – des accumulateurs R6 à 2 Ah sont en voie de développement – mais, malgré ces énormes progrès, les accumulateurs Lithium-ion modernes n'ont pas leur pareil pour produire le maximum d'énergie à volume et poids minimum. Ils sont certes encore coûteux et d'un emploi délicat, mais ils restent la meilleure solution pour assurer la maniabilité et la portabilité des véritables « dévoreurs » de courant que sont les ordinateurs portables et les caméscopes. Les accumulateurs spéciaux à courant élevé basés sur la technique Lithium-ion qui sont apparus récemment peuvent être utilisés dans les véhicules et même dans les ULM. Les accumulateurs NiMH du planeur à moteur électrique « Antares » développé par la firme allemande Lange Flugzeugbau à Zweibrücken seront remplacés avant même la fabrication en série par des accumulateurs Lithium-ion permettant à l'appareil qui pèse quand même 500 kg pour une puissance du moteur de 42 kW (57 ch) d'atteindre une altitude d'au moins 3 000 mètres.

Structure et propriétés

La tension nominale de 3,6 V ou 3,7 V des accumulateurs Lithium-ion utilisés dans les ordinateurs bloc-notes, les caméscopes, les portables, etc. est assez élevée pour de nombreuses applications, ce qui permet

par exemple de se passer des 2 à 3 éléments NiMH. Un seul élément suffit en fait dans les accumulateurs Lithium-ion.
Un élément d'accumulateur Lithium-ion est constitué d'une anode de graphite et d'une cathode en oxyde de cobalt de lithium ou oxyde de manganèse de lithium avec un électrolyte organique liquide dans lequel est dissous un sel de lithium qui fournit les ions lithium. La tension nominale est de 3,6 V (oxyde de cobalt) ou de 3,7 V (oxyde de manganèse) selon la substance de la cathode. Les tensions finales de charge correspondantes sont de 4,1 V (oxyde de cobalt) ou de 4,2 V (oxyde de manganèse). Il faut absolument respecter ces tensions finales de charge à ±50 mV près sous peine d'endommager irrémédiablement l'élément de l'accumulateur. Il faut aussi éviter à tout prix de décharger des éléments Lithium-ion en-deçà de 2,4 V ou 2,5 V sous peine de raccourcir irrémédiablement la vie de l'accumulateur.
Dans le cas des accumulateurs Lithium-ion, les ions lithium (Li+) quittent la cathode d'oxyde de cobalt de lithium en libérant un électron. Les ions lithium qui se trouvent dans l'électrolyte se fixent à l'anode de graphite en capturant des électrons.

Problèmes en cas de manipulation erronée

La surcharge d'un accumulateur Lithium-ion est très dangereuse pour lui. Elle peut provoquer un dégagement de gaz, une surchauffe et même faire exploser l'élément. Si la tension finale de charge de 4,1 V ou 4,2 V est dépassée de plus de 1 %, les ions lithium de l'élément commencent à se transformer en lithium métallique qui réagit très fortement au contact de l'eau de l'électrolyte, ce qui cause l'explosion de l'élément. D'autre part, la tension finale de charge de l'accumulateur Lithium-ion ne doit pas être trop basse de plus de 1 % sous peine de réduire sensiblement la capacité de l'élément. Un écart négatif de 100 mV par rapport au seuil entraîne déjà une perte de capacité de 7 %.
La décharge complète d'un accumulateur Lithium-ion entraîne rapidement une perte de capacité. Comme cet effet est irréversible, il faut éviter à tout prix de parvenir à la tension de décharge finale que l'on pourrait même, sans jeu de mot, qualifier de « terminale ».
Cette sensibilité aux erreurs de manipulation est aussi la raison pour laquelle les éléments Lithium-ion ne sont pas disponibles dans le commerce des composants. Les fabricants de systèmes Lithium-ion ne fournissent au contraire les éléments qu'à l'industrie ; c'est aussi pourquoi ils s'écartent des formes de construction usuelles (R6, R14, R20, etc.). Une grande partie des éléments fabriqués est destinée à des assembleurs d'articles pour réaliser des blocs d'accumulateurs ne pouvant être utilisés que dans un appareil prévu à cet effet (par exemple un ordinateur bloc-notes ou un caméscope). On s'assure ainsi que seul le chargeur idoine sera utilisé.

Blocs d'accumulateurs

La **figure 1** donne des exemples de mise en circuit de blocs d'accumulateurs. Dans cas le plus simple, un capteur de température NTC vient encore s'ajouter aux éléments de l'accumulateur (**figure 1a**). On obtient une protection plus efficace contre la surcharge ou la décharge complète du bloc d'accumulateur avec un circuit intégré de protection spécial (**figure 1b**, cf. dans ce cadre le paragraphe « Circuits intégrés actuels »). Ce composant surveille la tension des éléments et coupe s'il y a lieu le courant de charge ou de décharge à l'aide de deux FETMOS. Les FETMOS T1 et T2 sont commutés en opposition, sinon la diode intrinsèque, la « *body diode* », laisserait encore passer le courant en sens inverse bien que le FETMOS soit bloqué. Si par exemple T2 est en circuit, le courant de charge qui provient de la connexion « + », passe par les éléments d'accumulateur et parvient à la connexion de source de T1, s'écoule par la diode intrinsèque de T1 et par T2, qui est commuté, pour parvenir enfin à la connexion « – ». Inversement, le courant de décharge s'écoule par la diode de T2 si le FETMOS T1 est commuté. Si les deux FETMOS sont coupés, les diodes intrinsèques reliées en tête-bêche empêchent le courant de passer dans l'une ou l'autre des 2 directions.
Le circuit intégré de protection empêche la décharge complète en séparant simplement

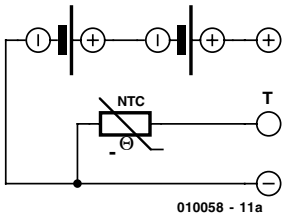


Figure 1a. Circuit d'un bloc d'accumulateur à capteur thermique.

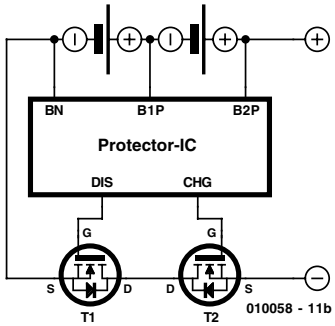


Figure 1b. Circuit d'un bloc d'accumulateur à circuit intégré de protection.

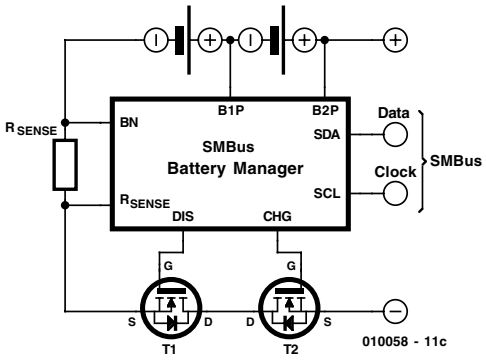


Figure 1c. Circuit d'un bloc d'accumulateur à circuit intégré de gestion de la batterie basé sur le SMBus.

Chimie de l'accumulateur (charge):

pôle positif: $LiCoO2 \rightarrow Li1\text{-}n\ CoO2 + n\ Li+ + n\ e-$
pôle negatif: $C + x\ Li+ + x\ e- \rightarrow CLin$

La décharge s'effectue par le processus exactement inverse.

l'accumulateur du consommateur. Il agit exactement de même en cas de surtension. Finalement, les FETMOS sont aussi commutés à impédance élevée lorsque le courant de charge ou de décharge devient trop élevé. Le circuit intégré lui-même consomme moins de 1 µA en mode d'attente.

la figure 1c montre un bloc d'accumulateur avec un circuit intégré de gestion de la batterie « *SMBus Battery Manager* » (SMBus = *System Management Bus*). On trouve ici par rapport à la solution précédente une résistance de mesure de courant (Rsense, le plus souvent de l'ordre de 10 milliohms) qui permet au circuit intégré de gestion d'observer exactement l'état de la charge. Les 2 fils du bus SCL (fréquence) et SDA (données) permettent au bloc d'accumulateur de communiquer avec le contrôleur de charge de l'appareil dans lequel le bloc d'accumulateur est utilisé. Le bus SMBus est disséqué plus bas.

Pour des raisons de sécurité, les accumulateurs Lithium-ion possèdent une soupape de sûreté comme les autres accumulateurs pour permettre aux gaz se formant en cas de surcharge de s'échapper. L'élément PTC installé aussi dans les cellules sert à la protection contre les courts-circuits : sa température qui augmente avec le courant accroît aussi sa résistance, ce qui a vite raison du courant de court-circuit.

Techniques de charge

Le processus de charge d'un accumulateur Lithium-ion s'effectue à courant/tension constants et exige une surveillance exacte de la tension des éléments car ces accumulateurs perdent vite leur capacité lorsque la méthode de charge est incorrecte. La première étape du processus de charge consiste à vérifier la tension de l'élément à vide. Si celle-ci n'atteint pas 2,5 V, cela signifie que l'élément est complètement déchargé. On doit alors tirer prudemment l'élément de cet état de décharge profonde (ce que l'on nomme « *prequalification* ») en limitant le courant de charge à 5 mA au moyen d'une charge de maintien (*trickle charge*). Une fois que l'élément a atteint 2,5 V, on peut commencer la charge rapide par une charge à courant constant de 1 C à 2 C. Cette alimentation en courant est maintenue jusqu'à ce que la tension de l'élément soit montée à 4,1 V (oxyde de cobalt) ou 4,2 V (oxyde de manganèse). La phase de tension constante commence à partir de ce point : le contrôleur de charge maintient la tension de l'élément à 4,1 V ou 4,2 V (±5 mV). Plus la charge progresse, plus le courant de charge diminue. Cette phase de complément (*top-off phase*) est terminée lorsque le courant tombe au-

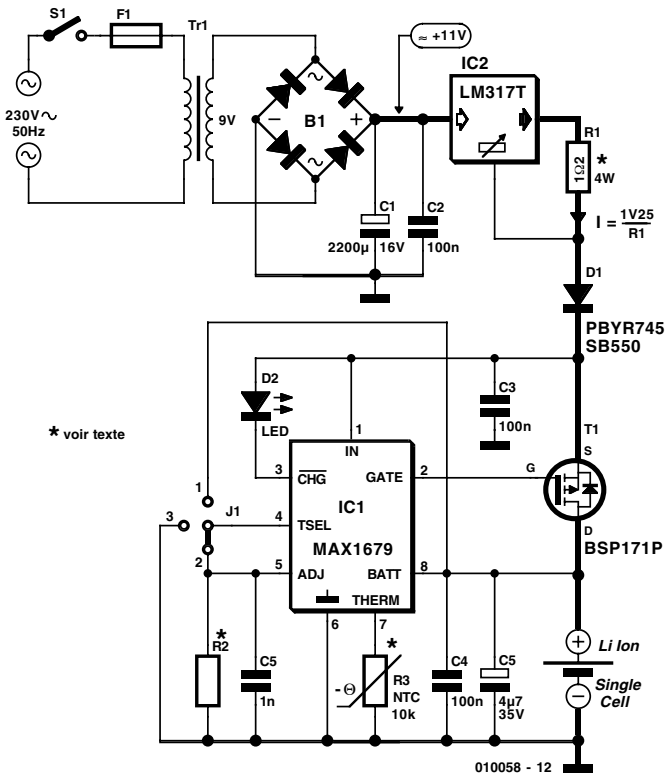


Figure 2a. Chargeur simple d'accumulateurs Lithium-ion basé sur le MAX 1679.

Tableau 1

Informations fournies par les LED avec le MAX 1679

État de la LED	Signification
LED clignote	– Qualification (Vbatt < 2,5 V)
LED allumée	– Charge (Charge rapide voire charge d'entretien en cours
LED clignote	– Écoulement du temporisateur de charge rapide
LED clignote toutes les 3,5 s	– Fin du processus de charge

dessous d'une valeur déterminée. Elle est généralement de 5 % de la valeur de la phase de courant constant, donc entre 0,05 C et 0,1 C. L'élément d'accumulateur Lithium-ion est alors complètement chargé.

Tant que les éléments ne sont pas encore complètement chargés, ils peuvent supporter plus de 4,2 V, par exemple une charge par impulsions. Pour des raisons de sécurité, les deux phases de charge sont surveillées au moyen de temporisateurs : il existe un temporisateur pour la charge rapide et un temporisateur pour la durée totale de charge. Si le temporisateur pour la charge rapide expire avant que l'élément n'ait atteint la tension de 4,2 V, le contrôleur de charge s'arrête et transmet une erreur. Le temporisateur pour la durée totale de charge interrompt la phase de complément au cas où l'interruption finale de celle-ci ne se produirait pas auparavant de la façon normale.

Dans le cas des accumulateurs Lithium-ion, aucune charge de maintien n'est nécessaire en raison de la valeur négligeable de l'autodécharge. Le passage constant d'un faible courant conduirait toujours à une surcharge de l'élément.

De plus, un capteur NTC contrôle généralement la température de l'élément d'accumulateur. Si l'élément s'échauffe trop au cours de la charge, le circuit intégré de charge interrompt celle-ci jusqu'à ce que l'élément se soit refroidi. La plage admissible se trouve par exemple entre +2°C et +45°C.

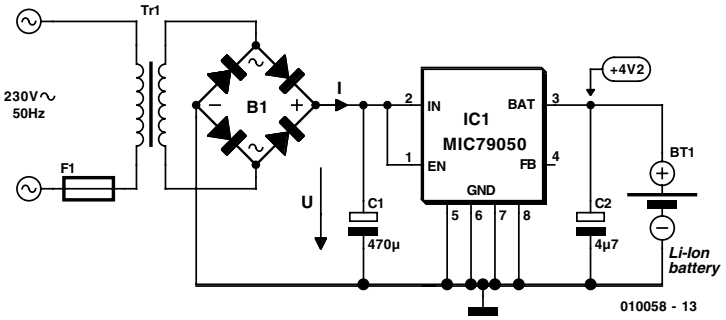


Figure 3a. Circuit de charge Lithium-ion extrêmement simple à bloc d'alimentation secteur et MIC 79050.

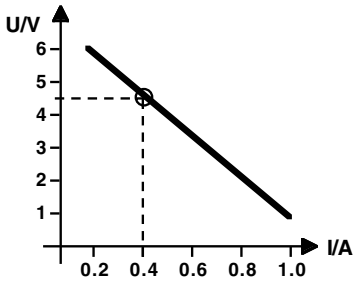


Figure 3b. Caractéristique en charge du bloc d'alimentation secteur.

Circuits intégrés simples de charge d'accumulateurs Lithium-ion

Le circuit MAX 1679 de MAXIM et quelques autres composants permettent de réaliser sans peine un chargeur d'accumulateur Lithium-ion (**figure 2**). Ce circuit intégré surveille très exactement la tension de l'accumulateur par la connexion BATT et commute par le FETMOS T1 un courant de charge d'une source de courant constant vers l'élément de l'accumulateur. Cette source est formée du régulateur de tension IC2 et de R1. Le régime pulsé à taux d'impulsions variable permet de positionner à volonté le courant de charge moyen au moyen du FETMOS entre zéro et le maximum. Le cycle de charge d'un accumulateur Ion-Lithium décrit plus haut peut donc être rigoureusement respecté. Le MAX 1679 active sans interruption le FETMOS pendant la phase de charge rapide et l'active au rythme du taux d'impulsions variable pendant la phase de complément. Le MAX 1679 permet de mesurer la tension maximale de 4,2 V de l'accumulateur avec la tolérance requise de moins de 1 %. Une diode luminescente indique l'état du chargeur (cf. le **tableau 1**).

La diode de Schottky D1 empêche l'élément d'accumulateur de se décharger par la diode intrinsèque du FETMOS une fois le processus de charge hors circuit. La résistance NTC raccordée à THERM permet de surveiller la température de l'élément. ADJ donne accès à une tension de référence interne qui fixe la tension supérieure de 4,2 V de l'élément et qu'on peut adapter s'il y a lieu à des accumulateurs Lithium-ion d'une tension de charge finale de 4,1 V. Le pont raccordé à TSEL permet de choisir une temporisation de plus ou moins longue durée (charge totale entre 2,8 heures et 6,25 heures). Pour que l'élément de l'accumulateur ne se décharge pas inutilement, le MAX 1679 se met hors circuit à la fin de la charge de sorte que le courant de l'accumulateur passant par BATT ou T1 n'atteint pas 1 µA.

On peut réaliser des circuits de charge d'accumulateurs Lithium-ion extrêmement simples avec des blocs d'alimentation secteur possédant une caractéristique appropriée en fonction de la charge appliquée. La **figure 3a** montre un circuit équipé du MIC 79050 de Micrel (*www.micrel.com*)**.** Ce circuit requiert que la limitation de courant par la résistance interne du transformateur –la caractéristique en charge– se comporte de telle sorte qu'un courant de 0,5 à 1,0 C s'établisse à +4,5 V (ce sont 0,4 A dans la **figure 3b**). Dans la phase de charge à courant constant, ce courant est fourni directement à l'élément d'accumulateur tant que la tension n'atteint pas 4,2 V. Lorsque la cellule atteint la tensions de 4,2 V définie dans le MIC 79050, la charge se poursuit à tension constante. Le courant de charge diminue et la tension d'entrée du MIC 79050 augmente.

La **figure 4** montre un circuit équipé de la puce LM 3622 de National Semiconductor. Ce circuit intégré forme avec le transistor PNP T1

L'effet de perte de mémoire

La fondation allemande « Stiftung Warentest » a testé l'an passé des éléments d'accumulateurs de taille R6 qui est la plus courante et ce, avec des résultats étonnants publiés dans la revue **test** (fascicule 7/2000) :

Effet de mémoire

Non-sens – écrit **test**. Les accumulateurs (aussi bien les CdNi que les NiMH) ont été 50 fois à moitié déchargés puis rechargés à fond. Résultat : perte de capacité nulle – des accumulateurs CdNi comme des NiMH ! Tous les accumulateurs testés ont obtenu « très bien » pour ce critère.

Autodécharge

Contrairement à la doctrine anciennement en vigueur, l'autodécharge des accumulateurs NiMH est plus faible que celle des accumulateurs CdNi. Le meilleur accumulateur NiMH a obtenu la mention « bien » pour ce critère (n'a subi qu'une faible perte de capacité après 80 jours à 20 °C), trois autres ont obtenu « satisfaisant » mais par contre, hormis un seul « suffisant », tous les accumulateurs CdNi testé étaient « insuffisants » pour ce qui est de l'autodécharge.

Conclusion : cessons de nous préoccuper de l'effet de mémoire – et oublions les accumulateurs cadmium-nickel (sauf pour l'outillage électrique et le modélisme – bien que cela soit aussi en train de changer lentement). Les cellules RAM (« Rechargeable Alkaline Manganese » ou « Rechargeable Au Manganèse » dans la terminologie canadienne) ne constituent pas vraiment, selon **test**, une alternative valable – trop chères et nombre de charges limité. Dans le test de « test », on s'est aussi servi de batteries alcali-manganèse « normales » (cellules primaires) au lieu de cellules RAM – qui se sont comportés exactement comme les « accumulateurs alcali-manganèse » coûteux. Mais cela, les lecteurs d'Elektor le savent depuis la parution de l'article « régénérateur pour piles alcalines » (n°222, décembre 1996)…

Tableau 2. Circuits intégrés de protection pour packs d'accus Li-Ion (similaires à celui de la figure 1b)			
Fabricant	**Type**	**Fonction**	**Nombre de cellules**
Maxim	MAX 1665	Lithium-Ion Battery Pack Protector	2, 3 ou 4 *
Maxim	MAX 1666	Advanced Lithium-Ion Battery Pack Protector	2, 3 ou 4 *
ON Semiconductor	MC33348	Lithium Battery Protection Circuit	1
ON Semiconductor	MC33349	Lithium Battery Protection Circuit	1
ON Semiconductor	MC33351A	Lithium Battery Protection Circuit	3
Philips Semiconductors	SAA1502	Safety IC for Li-ion	1
TI / Unitrode	UCC3952	Li-ion Battery Protection IC	1
** Plusieurs versions de circuits intégrés pour les différents nombres de cellules.*			

Tableau 3. Circuits intégrés de charge simples pour accus Li-Ion (similaires à celui de la figure 2 ou 3a)			
Fabricant	**Type**	**Fonction**	**Nombre de cellules**
Maxim	MAX1679	Single Cell Li+ Battery Charger	1
Maxim	MAX1736	Single Cell Li+ Battery Charger for Current-Limited Supply	1
Micrel	MIC 79050	Simple Lithium-Ion Battery Charger	1
Linear Technology	LTC1730	Lithium-Ion Battery Puls Charger	1
Linear Technology	LTC1731	Lithium-Ion Linear Battery Charger Controller	1
Linear Technology	LTC1732	Linear Lithium Battery Charger Controller	1
Linear Technology	LTC1734	Lithium-Ion Linear Battery Charger	1
National Semiconductor	LM3420	Lithium-Ion Battery Charge Controller	1 ... 4 *
National Semiconductor	LM3620	Lithium-Ion Battery Charge Controller	1 ... 2 *
National Semiconductor	LM3622	Lithium-Ion Battery Charge Controller	1 ... 2 *
TI / benchmarq	bq2400	Linear Li-Ion/Li-Polymer Charger	1 ... 2
TI / benchmarq	bq2057	Li-Ion Charge Management IC	1 ... 2
** Plusieurs versions de circuits intégrés pour les différents nombres de cellules.*			

et la diode de protection contre la décharge D1 un régulateur linéaire qui maintient exactement la tension de l'accumulateur B1 à la valeur prescrite de 4,1 V/élément ou de 4,2 V/élément. Rsense permet au réalisateur du circuit de paramétrer le courant de charge désiré au moyen de la formule :
Courant de charge = 0,1 V/Rsense.
Il faut que le transistor T1 soit en mesure de convertir l'excédent de la puissance d'entrée en chaleur.

La batterie intelligente

Avec l'arrivée des ordinateurs bloc-notes et des portables alimentés par accumulateur, il est devenu nécessaire de savoir assez précisément quand l'accumulateur sera vide. Rien n'est plus irritant que de constater, après un avertissement « Low Bat » lors de la sauvegarde, que l'accumulateur est en réalité vide. C'est pourquoi Intel et Duracell ont entrepris de développer un bloc d'accumulateur intelligent qui fournit au processeur de l'appareil par bus bifilaire (le SMBus, *System Management Bus*) des informations sur la chimie de l'accumulateur (NiCd, NiMH, Lithium-ion, etc.) et sur l'état momentané de la charge. Les informations mémorisées dans le bloc d'accumulateur sur le processus de charge (rapide) optimum permettent de prolonger la durée de la vie des éléments d'accumulateur utilisés.

Tableau 4. Circuits intégrés de charge universels					
Fabricant	**Type**	**Nombre de cellules NiMH/NiCd**	**Nombre de cellules Li-Ion**	**Type de cellules**	**Détection de fin de charge**
Maxim	MAX1772	2 à 4	2 à 4	Accu au Plomb, Li-Ion, CdNi, NiMH,universel	Paramétrage du courant et de la tension par le biais du contrôleur
National	LM3647	2 à 7	1 à 4	Li-Ion, CdNi, NiMH, universel	-ΔU, tension, température, temps
Linear Technology	LTC1325	1 à 8	1 à 3	Système de gestion d'accu piloté par microprocesseur pour accus au Plomb, Li-Ion, CdNi et NiMH	Tension, courant, température, temps

Tableau 5. Contrôleurs d'alimentation en courant pour accus Li-ion				
Fabricant	**Type**	**Nombre de cellules Li-Ion**	**Fonction**	**Détection de fin de charge**
Maxim	MAX1737 MAX1757 MAX1758	1 à 4 1 à 3 1 à 4	Stand-Alone Li-Ion Charger Controller	Limite de tension et de courant, thermistance, durée maximum

Oui, le bloc d'accumulateur peut même commander le Power Management de l'appareil à l'aide des diverses séquences de commande définies dans la norme SBS. SBS est l'abréviation de Smart Battery System et peut être utilisé sans licence

Adresses Internet

Infos concernant les accus:
www.ebatts.com/tips.asp (anglais)
www.batteryweb.com (anglais)
www.duracell.com/Fun_Learning/index.html (anglais)
http://www.nec-me.co.jp/english/index.htm
www.energizer.com/products/alkaline/ (anglais)
http://www.hobbyseek.net/regio-f/ (en construction)

Fabricants de cellules pour accumulateurs :
http://www.accucell-usa.com/start.htm (anglais)
http://www.sonnenschein-lithium.de/ (anglais)
http://www.ansmann.de/eansman.html (anglais)
www.duracell.com
www.duracellusa.com
www.energizer.com,
www.energizer-eu.com
www.hitachi.com/products/material/batteries/index.html
www.panasonic.com/industrial_oem/battery/battery_home.htm
www.philipsbatteries.com
www.power-sonic.com/
www.renata.com
www.saft.alcatel.com
www.sanyo-energy-europe.com
www.sonnenschein-lithium.de
www.tdk-europe.com/
www.toshiba.com/taec
www.varta.de

Fabricants de circuits intégrés :
www.maxim-ic.com
www.national.com
www.onsemi.com
www.linear-tech.com
www.micrel.com
www.dalsemi.com
www.semiconductors.philips.com

par toutes les autres sociétés intéressées (www.SBS-forum.org). La surveillance du courant de charge et de décharge permet de calculer très exactement l'état de charge momentané et donc la durée résiduelle de fonctionnement d'un appareil. Des contrôleurs de charge en technique Smart Battery sont fournis par exemple par MAXIM (MAX1645, MAX1647/1648 et MAX1667) et par Linear Technology (LTC1759).

Dallas Semiconductor propose aussi un composant pour la surveillance de blocs d'accumulateurs Lithium-ion, le DS2760, qui utilise toutefois le bus Dallas à 1 fil (*one wire bus*).

Circuits intégrés actuels

De nombreux circuits intégrés pour la surveillance et la charge d'accumulateurs Lithium-ion sont apparus l'an passé sur le marché. Les principaux fabricants, Maxim, Linear Technology, ON-Semiconductor (société scindée de Motorola), National Semiconductor et Philips Semiconductors offrent une vaste gamme de produits qui ne se limite pas aux circuits intégrés de protection et de charge des accumulateurs Lithium-ion. Micrel offre le très simple MIC 79050 évoqué plus haut. Les tableaux 2 à 5 donnent un aperçu. On trouvera des renseignements détaillés et des fiches de données sur le site Internet de chaque fabricant. Le texte encadré « Adresses Web » ne contient pas que les adresses Internet des fabricants de circuits intégrés, mais aussi celles des fabricants connus d'éléments d'accumulateurs et de sites contenant des informations intéressants sur les accumulateurs.

(010058)

Bibliographie :
Ça bouge chez les accus et batteries, Elektor n°260, février 2000, page 18 et suivantes
Panorama des piles, Elektor n°273, mars 2001, pages 50 et 51
Chargeur d'accus HP, Elektor n°256 et 257, octobre et novembre 1999, pages 20 et 26 respectivement, et suivantes
Note d'application 64, Using the LTC1325 Battery Management IC, Linear Technology, août 1996

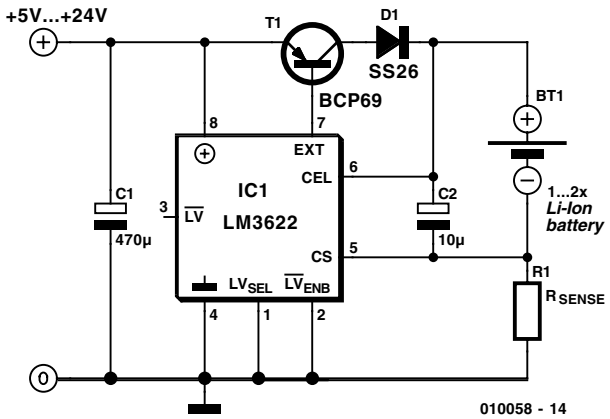


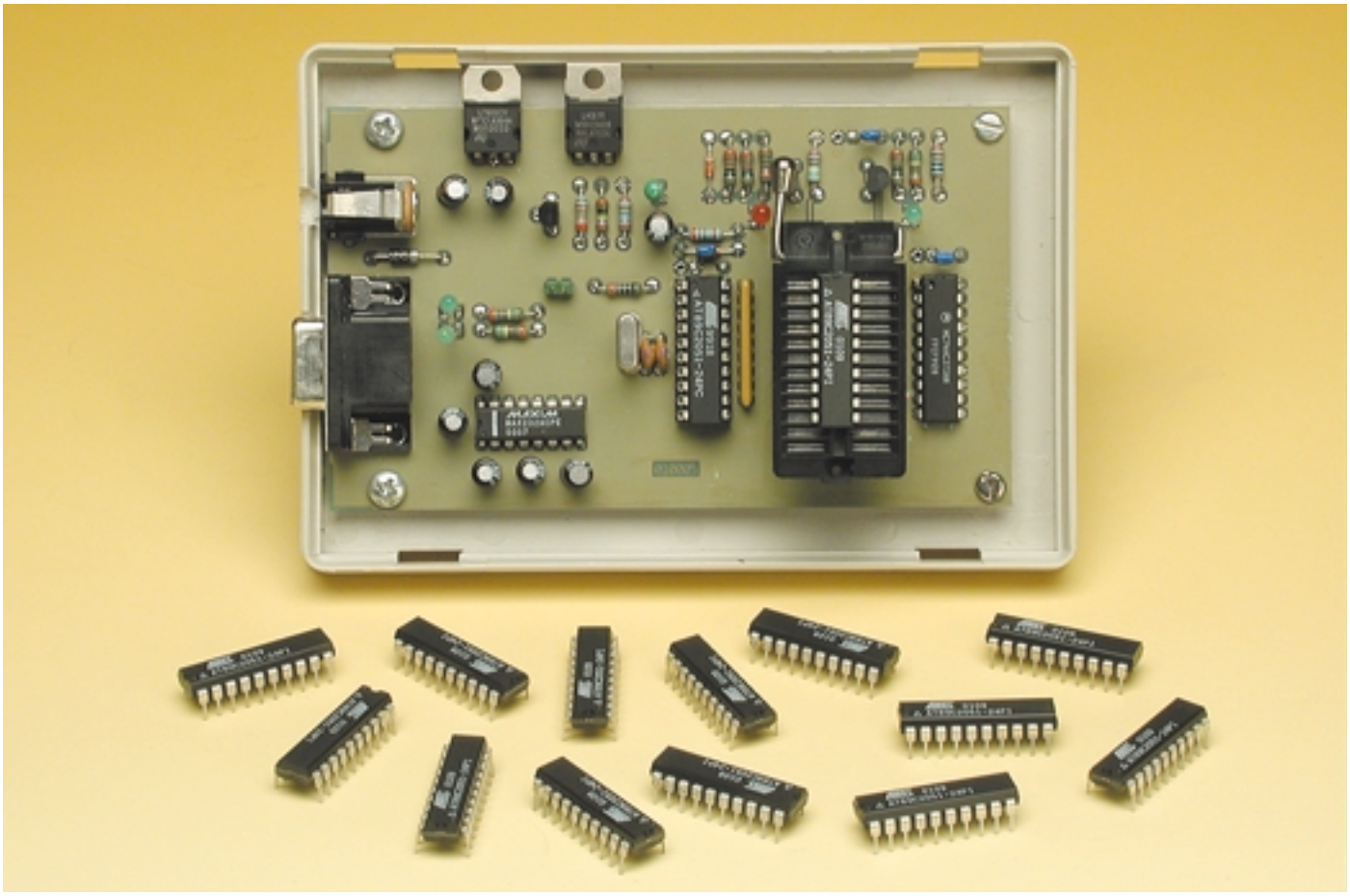
Figure 4. Chargeur Lithium-ion à fonctionnement linéaire basé sur le LM 3622.

Programmateur Atmel

pour les 89Cx051 jusqu'à 4 K

Andreas Oyrer

Le programmateur présenté ici permet de programmer les 3 processeurs à 20 broches de la famille 89CXXX d'Atmel les plus connus à savoir les 89C1051, 89C2051 et le tout récent 89C4051 de 4 Koctets doté lui de mémoire de programme.



La famille des microcontrôleurs 8 bits compatibles MCS-51 d'Atmel à mémoire Flash (EEPROM) à la puissance remarquable et au prix resté très abordable s'est agrandie d'un nouveau membre, le 89C4051, doté d'une mémoire de programme de 4 Koctets. Si l'on fait abstraction de la taille de la mémoire de programme, les différents membres de cette famille ont des caractéristiques identiques :

- 128 octets de RAM
- 15 lignes d'E/S
- 2 compteurs/décompteurs 16 bits
- Une architecture d'interruption à 2 niveaux et 5 vecteurs

– Port sériel full duplex programmable
– Comparateur analogique de précision
– Circuit d'horloge/oscillateur intégré.

La famille AT89Cx051 peut travailler à n'importe quelle fréquence d'horloge allant de 0 Hz (donc en statique) à 24 MHz et supporte 2 modes économiques pilotables par logiciel à savoir *Idle* (la CPU est arrêtée) et *Power-down* (la RAM est sauvegardée, le reste des fonctions étant mises à l'arrêt).

Programming the Flash

L'important, pour n'importe quel programmateur, est bien évidemment les caractéristiques de programmation de la mémoire Flash. Lorsque le microcontrôleur quitte les chaînes de fabrication de chez Atmel la matrice de mémoire de programme ne contient que des FF_{HEX}, qu'il est prêt à l'écriture et cela octet par octet (*bytewise*). Une fois la matrice programmée, il est possible de (re)programmer un octet qui n'est plus vierge uniquement par réécriture, après effacement de la totalité de la mémoire, ce qui implique la reprogrammation de l'ensemble de la mémoire de programme. Un AT89Cx051 possède un compteur d'adresse EEPROM qui est remis à zéro (00_{HEX}) par application d'un flanc montant sur l'entrée de RAZ (RST = *Reset*). Chaque impulsion d'horloge appliqué à l'entrée XTAL1 en provoque l'incrémentation.
Les états logiques des lignes de port P3.1 à P3.7 paramètrent le mode de programmation; le **tableau 1** récapitule cette information.
Atmel recommande, pour le transfert d'un programme dams la EEPROM, l'utilisation de **l'algorithme de programmation** suivant :

1. On applique, pour la mise en fonction, la tension d'alimentation entre les broches VCC (+ de l'alimentation) et GND (Masse) et on force les lignes RST/VPP et XTAL1 à la masse.

2. Mettre RST/VPP et P3.2 au niveau logique haut (H).

Modes de programmation Flash

Mode	RST/VPP	P3.2/PROG	P3.3	P3.4	P3.5	P3.7
Écriture des données de programme	12 V	impulsion négative 1,2 ms	L	H	H	H
Lecture des données de programme	H	H	L	L	H	H
Écriture du Lockbit 1	12 V	impulsion négative 1,2 ms	H	H	H	H
Écriture du Lockbit 2	12 V	impulsion négative 1,2 ms	H	H	L	L
Effacement de la mémoire	12 V	impulsion négative 1,0 ms	H	L	L	L
Lecture de l'octet de de signature	H	H	L	L	L	L

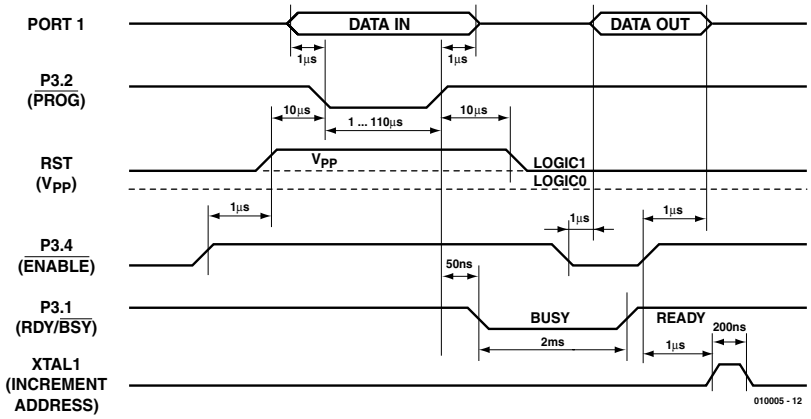


Figure 1. Chronodiagramme d'un cycle d'écriture/lecture.

3. Paramétrer les lignes de port P3.3 à P3.7 en respectant les informations du tableau 1.

L'écriture et la lecture de données de programme requièrent les étapes suivantes :

4. Le premier octet de programme pour l'adresse de mémoire 000_{HEX} est placé sur les lignes P1.0 à P1.7.

5. Mettre la ligne RST/VPP à 12 V et lancer le processus de programmation.

6. Il faut appliquer à la ligne P3.2 une impulsion négative d'une longueur comprise entre 1 et 110 µs pour obtenir l'écriture de l'octet dans la matrice d'EEPROM ou celle des bits de verrouillage (*lock bits*) servant à assurer la sécurité du contenu du processeur. Le cycle d'écriture se termine automatiquement après de l'ordre de 1,2 ms.

7. La lecture des données requiert de faire passer la tension appliquée sur la ligne RST de 12 V à une valeur correspondant à un niveau logique haut et de mettre les lignes de port P3.3 à P3.7 aux niveaux requis. L'octet se trouvant à l'adresse actuelle apparaît sur le port P1.

8. Une fois l'opération de lecture/d'écriture de l'octet faite, le compteur d'adresse est incrémenté par l'application d'un flanc montant sur la ligne XTAL1. Le nouvel octet est placé sur le port P1.

9. Les pas 5 à 8 sont répétés jusqu'à ce que l'ensemble de l'espace de mémoire soit occupé ou que soit atteinte la fin du code de programme.

10. Il faudra, pour obtenir la fin du processus

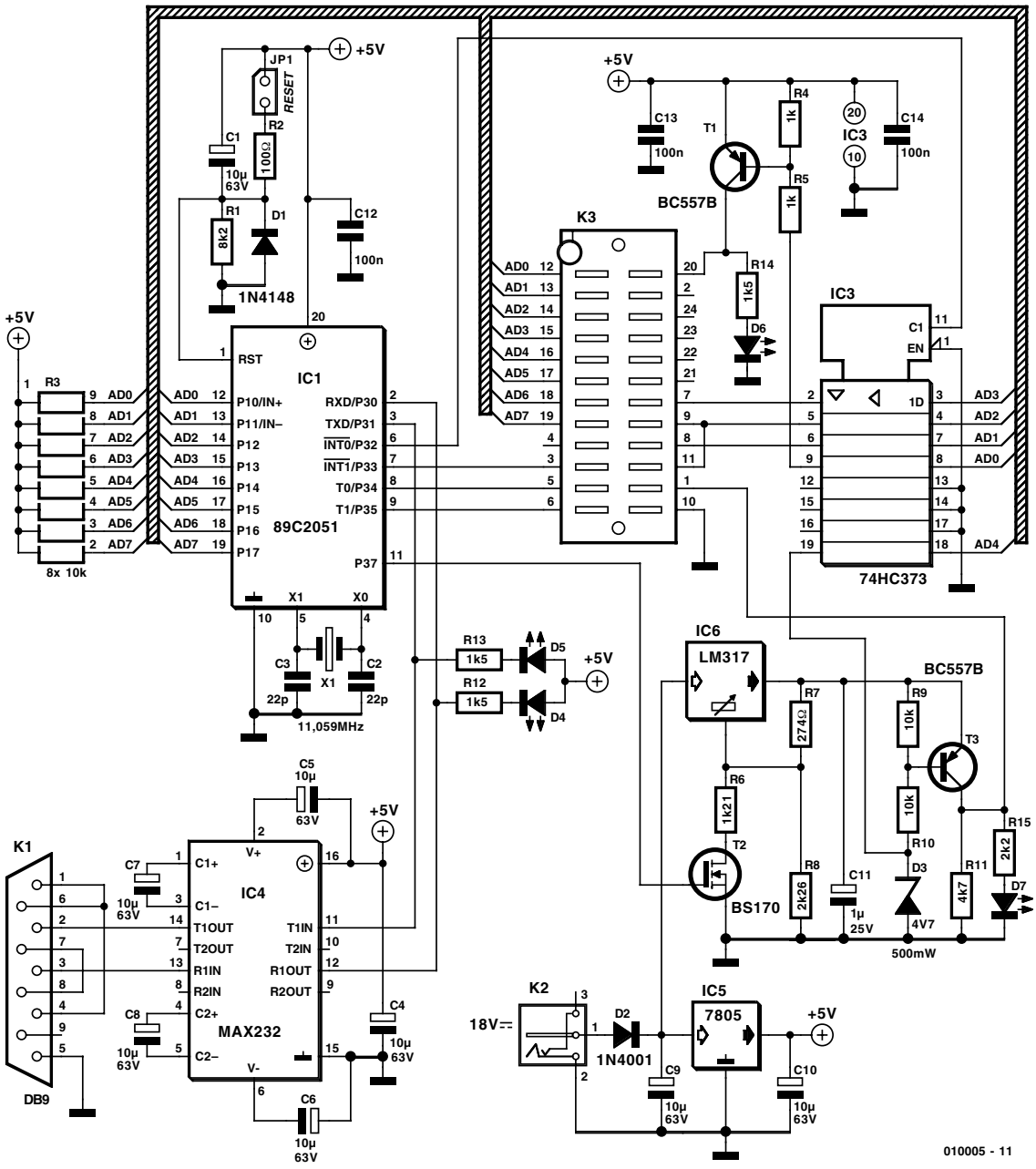


Figure 2. L'électronique du programmateur Atmel comporte 2 modules « encombrants », le microcontrôleur-maître et le support FIN destiné à recevoir le composant à programmer.

de programmation, mettre les lignes XTAL1 et RST à la masse et couper la tension d'alimentation VCC/GND.

Le chronodiagramme de la **figure 1** reproduit un cycle d'écriture/lecture complet. La ligne de port P3.1 qui signale READY/$\overline{\text{BUSY}}$ attire l'attention. La fiche de caractéristiques indique, pour la longueur de l'impulsion de programmation, une durée comprise entre 1 et 110 µs. Sachant cependant qu'un cycle de programmation dure lui de l'ordre de 1,2 ms et qu'il est recommandé de ne pas lancer de nouveau cycle, la ligne P3.1 bascule au niveau bas (BUSY) pendant quelque 2 ms lors de l'écriture. Ce n'est qu'après la remontée de P3.1 au niveau haut (READY) que l'on peut avoir lancement d'un nouveau cycle d'écriture.

Il est possible, comme le montre le chronodiagramme, de **vérifier** le processus d'écriture, octet par octet au cours de la durée du BUSY lors de l'écriture (vérification qui est en fait une comparaison) ou en une seule fois, si tant est que les bits de protection (*lockbits*) ne soient pas positionnés, (lecture) :

1. Pour cela on fait passer RST du niveau bas au niveau haut, mettant du même coup le compteur

d'adresse à 000_{HEX}.

2. Les signaux de commande correspondant au mode choisi sont appliqués aux lignes P3.3 à P3.7 et l'octet de données lu sur le port P1.

3. L'application d'une impulsion sur XTAL1 incrémente le compteur d'adresse (le fait passer à l'adresse suivante) de sorte que l'on aura apparition sur le port P1 de l'octet de donnée suivant.

4. On a répétition de l'étape 3 jusqu'à ce que l'on en ait terminé avec la lecture de l'ensemble de la mémoire.

Il n'est pas possible de vérifier les bits de protection, seul leur comportement permet de les identifier.
On aura effacement électrique de l'ensemble de la mémoire, celui des bits de protection compris, à la suite de l'application de la combinaison requise aux lignes P3.3 à P3.7 et de la mise de la ligne P3.2 au niveau bas pendant une durée minimum de 10 ms. La totalité des emplacements de mémoire se trouve alors remplie de « 1 ». Ce processus d'effacement de puce est une condition impérative si l'on veut reprogrammer une mémoire qui n'est plus vierge ou effacer les bits de protection.
Les octets de signature identifient le fabricant et le type de composant. Il est possible de les lire à l'image des emplacements de la mémoire de programme, mais il faut pour cela que les lignes P3.5 et P3.7 soient mises au niveau bas. On verra s'afficher les valeurs suivantes :

000H	1EH: (source) Atmel
001H	11H: 89C1051
	21H: 89C2051
	41H: 89C4051

Le programmateur sous l'aspect matériel

Vu que toutes les fonctions et toutes les caractéristiques des 3 microcontrôleurs sont identiques, exception faite de la taille de la mémoire de programme, un programmateur n'a guère de problème pour remplir sa tâche. Il lui suffit en effet, au début de tout processus, de lire l'octet de signature pour identifier le type de contrôleur auquel il a affaire.
Le programmateur décrit dans le présent article supporte tous les modes de programme décrits :

– *write* (écriture)
– *read* (lecture)
– *verify* (vérification)
– *program lockbit* (programmation des bits de verrouillage)
– *erase* (effacement)
– *read signature byte* (lecture de l'octet de signature).

Lors de l'écriture on vérifie, par le biais de l'information fournie par l'octet de signature, que la taille du fichier choisi ne dépasse pas la capacité de mémoire du type de processeur utilisé. Il est possible, lors d'une lecture, de mémoriser le code sous 2 formats différents : en Intel-Hex ou en binaire. La lecture de l'octet de signature est, au demeurant, facultative et l'on pourra, si nécessaire (lorsque l'on a affaire à un contrôleur défectueux par exemple) désactiver cette fonction dans le menu. Dans ce cas-là le programmateur suppose qu'il s'agit d'un 89C4051.
Comme le montre le schéma de la **figure 2**, l'électronique du programmateur est relativement simple, se réduisant en fait à un microcontrôleur-maître qui ne comporte en fait que 865 octets de code de programme, code indispensable au fonctionnement du programmateur. IC1, un 89C2051 bien évidemment, se charge de toutes les tâches qu'implique la programmation : il paramètre le mode de programmation en fonction des souhaits de l'utilisateur, convertit au format parallèle les données du code de programme arrivant par le biais de l'interface RS-232 du PC, transmet le contenu de la mémoire venant tout juste d'être programmée de même que toutes sortes d'autres messages vers le PC et pilote l'application et la coupure des différentes tensions (0 V, 5 V et 12 V) en respect des exigences du chronodiagramme. En dépit de toute la bonne volonté que l'on peut y mettre, les 15 lignes d'E/S dont dispose le 89C2051 sont loin de suffire et leur petit nombre complique sensiblement les choses.
La communication avec le PC se fait par le biais des lignes de port P3.0 et P3.1. Le clignotement des 2 LED D4 et D5 trahit l'existence d'un transfert de données. La ligne de port P3.3 (Interrupt 1) sert d'entrée pour la surveillance du signal $\overline{BUSY}$/READY du microcontrôleur à programmer (identifié sur le schéma par l'acronyme *PUP* = *Processor Under Programming*). L'impulsion de programmation s'est vu réserver elle aussi une ligne (à savoir P3.5); il en est de même pour le flanc attaquant XTAL, l'impulsion d'incrémentation du compteur d'adresse du PUP qui elle occupe la ligne P3.4.
Le reste des lignes disponibles, 10 en tout et pour tout, doivent partant fournir 5 signaux de commande et permettre le transfert de 8 bits de données, combinaison qui requiert impérativement de trouver une astuce et d'y faire appel.
La totalité du port P1 de IC1 est reliée à son homologue du *PUP*, 5 de ses lignes, à savoir P1.0 à P1.4, vont également à IC3, un verrou de type D (*D-Latch*) à 8 bits du type 74HC373. Le microcontrôleur commence par placer les signaux de commande sur les 5 verrous utilisés. L'entrée d'horloge (broche 11) commence par se trouver au niveau haut de sorte que les verrous sont transparents : les 3 signaux de commande AD1 à AD3 (comme le montre le tableau 1, les lignes P3.5 et P3.7 gardent toujours le même niveau) arrivent directement au *PUP*, AD0 pilote, par le biais du transistor T1, la ligne d'alimentation +5 V du contrôleur, la présence de la tension d'alimentation étant visualisée par la LED D6, et AD4 assure la présence de la tension de programmation sur la broche RST/VPP.
Une fois que ces différentes opérations ont eu lieu, l'entrée d'horloge du verrou repasse au niveau bas, ce qui se traduit par une mémorisation des signaux de commande.
Il faut, pour la génération des tensions requises sur la ligne RST/VPP, un régulateur de tension variable au programmateur Atmel, composant qui prend ici la forme de IC6. Si l'on a besoin, pour une lecture, de 5 V sur cette ligne, le transistor FET petits signaux T2 est passant et les 2 résistances R6 et R8 se trouvent prises en parallèle. Si l'on a besoin de 12 V, le FET est bloqué de sorte que seule R8 se trouve prise en circuit. La commutation du BS170 se fait par le biais d'une ligne de port propre (P3.7) de IC1.
De façon à ce qu'en fin de processus de programmation/lecture la tension présente sur la ligne VPP/RST soit au potentiel de masse (et qu'ainsi le PUP puisse être extrait de son support sans risque) le LM317 est basculé vers une tension de 5 V.
En outre, la ligne AD4 produit, par le biais du verrou, une tension de 5 V aux bornes de la combinaison R1/D3. Dans ces conditions le blocage du transistor T3 est garanti et la

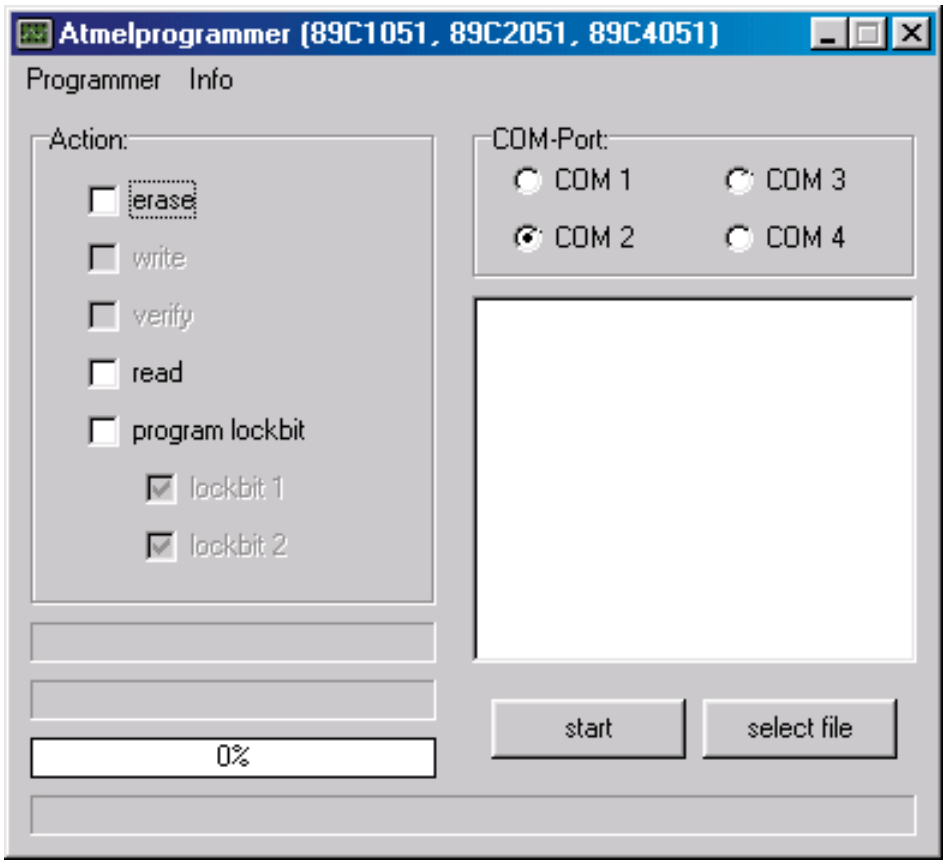


Figure 3. L'interface utilisateur du programme pour PC Programmer_eng.exe.

ligne RST/VPP se présente inévitablement, au travers de R11, le potentiel de masse.
Ainsi, toutes les lignes de port du 89Cx051 sont utilisées.

Le logiciel

La disquette (**EPS010005-11**) correspondant à ce projet et disponible auprès des adresses habituelles comporte un programme en assembleur baptisé *Atmelpr.hex* à mettre dans le microcontrôleur-maître; sachez qu'il existe, pour ceux d'entre nos lecteurs qui n'auraient pas la possibilité de programmer eux-mêmes –pour le moment encore car une fois ce projet terminé ils l'auront– ce type de composant, un microcontrôleur programmé (**EPS010005-41**) disponible auprès de ces mêmes adresses. Ce logiciel d'exécution est épaulé par un programme écrit en Delphi baptisé *Programmer_eng.exe* qui se charge, au niveau du PC, du pilotage de toutes les activités du programmateur Atmel telles que lecture, écriture et effacement. La disquette comporte également, outre la version en anglais (avec certains commentaires en allemand) une version dans le langue de Goethe (l'allemand) de ce programme ainsi que son code-source écrit en Delphi. Le programme tournant sur le PC (ainsi d'ailleurs que le dessin de platine, mais pas le programme à griller dans le processeur présent lui sur la disquette évoquée plus haut) est disponible gratuitement au téléchargement sur notre site Internet dont l'adresse ne vous est sans doute pas inconnue (*www.elektor.presse.fr*).
Le port série paramétré par défaut est COM2. Lors de l'exécution du programme on voit apparaître une fenêtre à 2 « volets » (cf. **figure 3**). Dans la partie gauche on coche tout simplement les actions à effectuer, la partie droite visualisant les actions ayant été effectuées. On aura ainsi par exemple, apparition d'une série de messages telle que :
reading signature byte ok
programming lockbit 1 ok
error: file to large !
Et ainsi de suite.
Si l'on coche *read*, le reste des fonctions est mis hors-fonction et l'on voit apparaître une fenêtre permettant de choisir le fichier dans lequel devra être écrit contenu du processeur. Si l'on donne l'extension .hex on aura automatiquement lecture d'un ficher de format Intel-Hex, si l'extension est .bin, on aura lecture d'un fichier binaire.
Le bouton « *select file* » permet de choisir le fichier que l'on veut écrire. On a à nouveau le choix entre le format .hex et .bin. La taille du fichier est indiquée sous le type de contrôleur. Un menu déroulant visualise l'état d'avancement des différentes actions. Dans le cadre du bas on voit s'afficher le nom du fichier et son cheminement (*path*).
La fonction *read signature byte* permet de choisir s'il faut, lors du lancement de l'exécution de l'opération, lire ou non l'octet en question. Il faudra toujours activer cette fonction sauf si le programmateur se trouve dans l'impossibilité de lire l'octet de signature. L'information concernant l'octet de signature est affiché sous la fenêtre de sélection de type de contrôleur. Les fonctions d'écriture (*write*) et de lecture (*read*) requièrent elles aussi cette information, sachant que sinon elles utilisent comme paramétrage par défaut les caractéristiques du 89C4051.
Si l'on a oublié de connecter le programmateur au PC on verra s'afficher le message « *timeout, check programmer or COM-Port !* ». Il faudra dans ce cas-là procéder à la vérification de la liaison sérielle et s'assurer que les lignes RxD, TxD et GND (Masse) sont interconnectées une à une et non pas croisées et vérifier en outre que le programmateur est bien mis en fonction. S'il devait se faire que la taille du fichier dépasse la capacité de mémoire de programme du contrôleur enfiché dans le support, un message le signale. On aura également émission d'un message (« *error in file* » si le fichier comporte une erreur.

Réalisation et étalonnage

L'électronique du programmateur Atmel prend place sur une petite platine double face à trous métallisés dont on retrouve le dessin des pistes et la sérigraphie de l'implantation des composants en **figure 4**. Il faudra faire en sorte que le plan du support de programmation soit légèrement rehaussé par rapport au reste des composants de manière à ce qu'il soit dans le même plan que le

Liste des composants

Résistances :
R1 = 8kΩ2
R2 = 100 Ω
R3 = réseau SIL de 8 résistances de 10 kΩ
R4,R5 = 1 kΩ
R6 = 1 210 Ω
R7 = 274 Ω
R8 = 2 260 Ω
R9,R10 = 10 kΩ
R11 = 4kΩ7
R12 à R14 = 1kΩ25
R15 = 2kΩ2

Condensateurs :
C1,C4 à C10 = 10 µF/63 V vertical
C2,C3 = 22 pF
C11 = 1 µF/25 V vertical
C12 à C14 = 100 nF

Semi-conducteurs :
D1 = 1N4148
D2 = 1N4001
D3 = diode zener 4V7/500 mW
D4 à D6 = LED verte à haut rendement
T1,T3 = BC557B
T2 = BS170
IC1 = AT89C2051-12PC (programmé EPS010005-41)*
IC2 (K3) = support FIN 24 contacts
IC3 = 74HC373
IC4 = MAX232
IC5 = 7805
IC6 = LM317T

Divers :
JP1 = embase à 2 contacts cavalier
K1 = embase sub D encartable en équerre à 9 contacts
K2 = embase jack d'alimentation
X1 = quartz 11,059 MHz
Boîtier 137 x 95 x 25 mm, tel que, par exemple, Pactec WM46

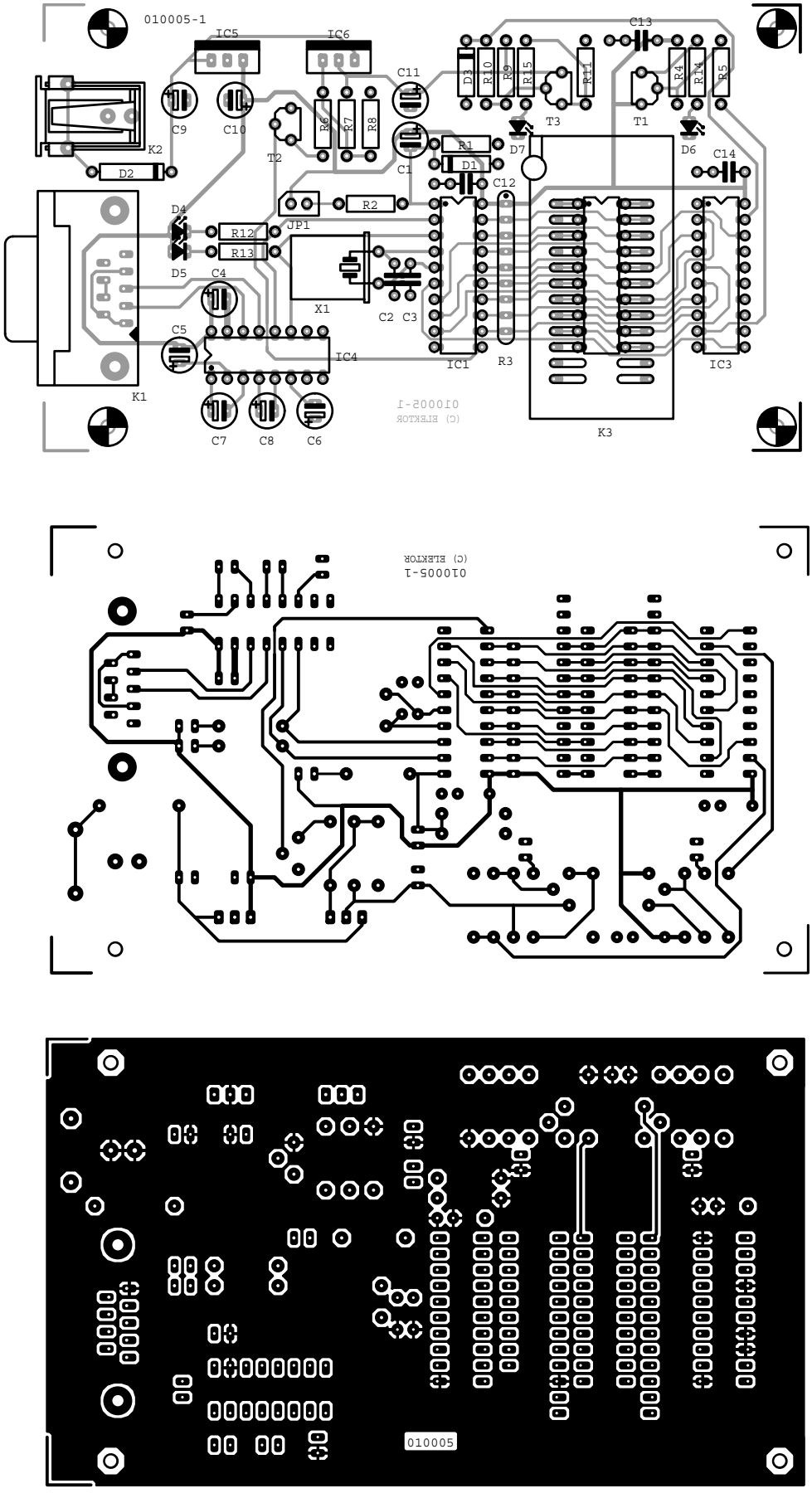


Figure 4. Dessin des pistes et sérigraphie de l'implantation des composants de la platine double face à trous métallisés conçue à l'intention de cette réalisation.

dessus du couvercle et qu'il soit facile de mettre le *PUP* en place dans le support et de l'en sortir. L'approche la plus simple et la moins chère consiste à utiliser un support à wrapper. Mais la solution la plus professionnelle (au niveau du coût aussi d'ailleurs) prendra la forme d'un support FIN (à **F**orce d'**I**nsertion **N**ulle) qui n'existe malheureusement dans le commerce qu'avec un minimum de 24 broches. Nous avons prévu suffisamment de place

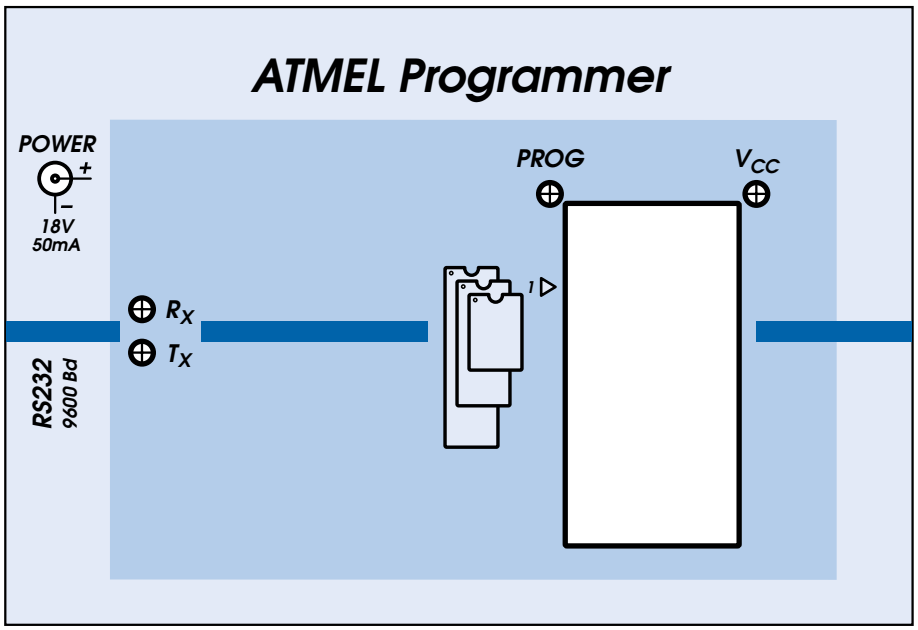
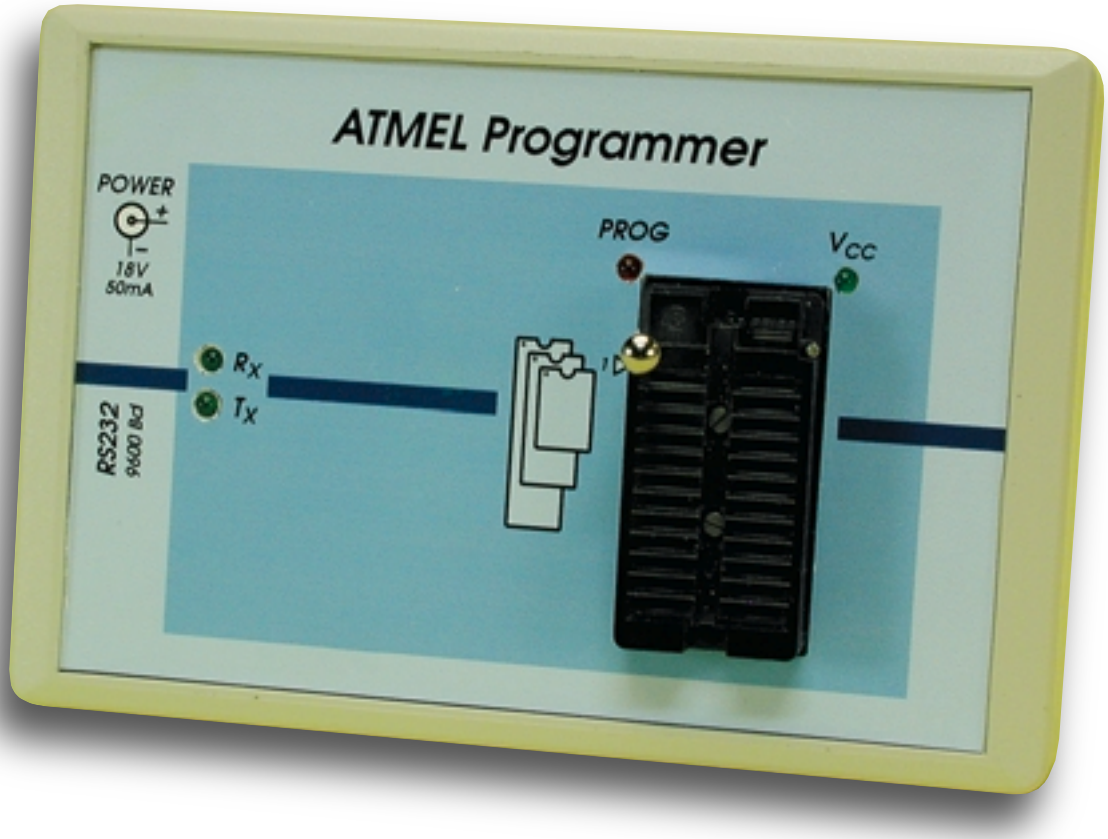


Figure 5. Rien de tel qu'un joli dessin de face avant pour donner un fini professionnel à une réalisation.

sur la platine pour l'utilisation d'un composant de ce type.
L'embase Sub-D à 9 contacts prend aisément place dans le boîtier proposé, le jack d'alimentation est relativement serré lui de sorte qu'il faudra le limer légèrement.
Une fois que l'on en aura terminé avec la construction de la platine, que l'on en aura contrôlé la réalisation et qu'elle aura trouvé place dans le boîtier proposé dans la liste des composants (et pour lequel est dimensionné le dessin de face avant proposé en **figure 5**) il ne reste plus qu'une seule opération à effectuer avant de pouvoir procéder à sa première programmation : vérifier les tensions de programmation.
Pour ce faire, on connecte le programmateur au PC et on choisit l'écriture d'un fichier quelconque. On désactivera l'option « *Read Signature Byte* » vu que sinon on aurait apparition du message « *error : wrong or no processor* ». Pendant le processus d'écriture la ligne RST est mise à 12 V. On pourra, si la dite tension n'atteint pas le niveau requis, augmenter légèrement la valeur de R8 (ce qui se traduit par une augmentation de la tension) ou en réduire quelque peu la valeur (la tension, logiquement, diminue).
On procède de la même façon au niveau de R3 pour disposer de la tension de 5 V. Pour cela on choisit ensuite l'option « *read* » et on joue, si nécessaire, sur la valeur de la résistance prise en parallèle sur R3. Ceci termine, comme le dit souvent le porte-parole du jury d'un pays quelconque lors du Grand Prix Européen de la Chanson, le processus d'étalonnage des tensions; vous pouvez maintenant vous lancer dans la programmation de votre premier microcontrôleur de la famille prolifique d'Atmel.

(010005)



Projets MP3 à foison

Portables à faire soi-même

Harry Baggen

Il y a quelques mois nous avons, dans le cadre de cette rubrique, donné un certain nombre d'adresse Internet où étaient présenté des projets MP3. Dans la majorité des cas ceux-ci utilisaient le matériel d'un PC mis au rancard. Mais il existe d'autres approches. Il y a même des projets qui vous permettent de réaliser un lecteur MP3 réellement portable.

L'approche basée sur une carte-mère de PC, un disque dur voire un lecteur de CD-ROM associés à quelques touches de commande, un affichage LC et un logiciel spécifique est loin d'être une voie sans issue lorsque l'on envisage de réaliser son propre lecteur de fichiers MP3, mais cette combinaison est relativement encombrante. MP3 est devenu aujourd'hui un sujet tellement populaire que nombre d'amateurs d'électronique se sont penchés sur le sujet ce qui explique que nous puissions découvrir des sites Internet consacrés à la réalisation de lecteurs MP3 réellement portables (et non pas uniquement « transportables »). Le support de stockage prend la forme d'un (mini-)disque dur, d'un lecteur de CD-ROM ou d'une carte de mémoire Flash.

Le site **MP3Projects** [1] constitue un bon point de départ à la recherche de projets de ce genre. Il donne un panorama quasi-complet et actualisé des projets MP3 à faire soi-même proposés sur la Toile au niveau mondial.

En partant de ce site, nous avons visité quelques adresses MP3 travaillant sur de petits lecteurs portables. Nous avons, à dessein, cette fois, passé à côté des usines à gaz (à base de cartes-mères pour PC).

MP3 PuBliC de **MP3ar** [2] est un bel exemple de lecteur MP3. On se trouve ici en présence d'une platine à peine plus grande qu'une souris. On peut y connecter un affichage LCD par le dessus et utiliser, dans la version la plus récente du projet, tant un lecteur de CD-ROM qu'un disque dur. Un programme pour PC écrit pour cette réalisation permet le transfert de fichiers d'un PC vers le disque dur. Tous les dessins de circuits imprimés, les logiciels et les codes-sources sont mis à disposition gratuitement. Ce sont 3 Argentins qui sont les pères de ce projet. L'enveloppe financière de la réalisation est estimée à quelque 130 $.

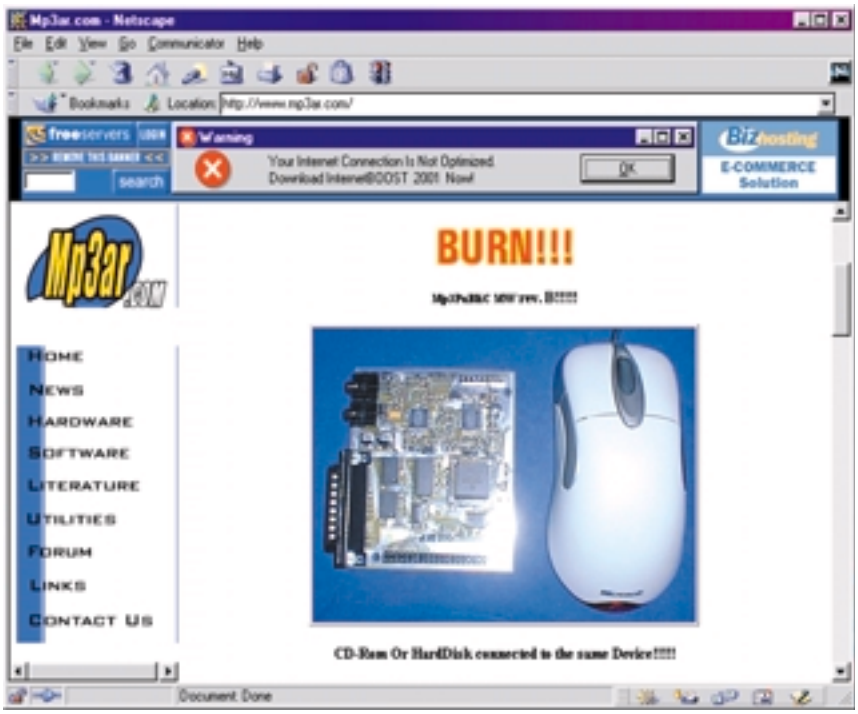
Le **High Capacity MP3 player** de **MPRJ** [3] est un projet qui mérite indiscutablement lui aussi le qualificatif de « portable ».

Ce lecteur utilise un (mini-)disque dur IDE standard d'une capacité de plusieurs centaines d'heures de musique. De par l'utilisation d'un CAN 24 bits, cet appareil peut se targuer d'une qualité sonore excellente. Le lecteur peut travailler à différents taux de compression MP3, de sorte que l'utilisateur peut choisir la qualité sonore qu'il désire. MPRJ fournit lui-même tous les composants de ce projet. On peut acheter une platine montée et testée (150 $) voire la platine seule ou les différents composants. Le code-source du progiciel sont mis à la disposition de ceux qui voudraient modifier l'une ou l'autre chose sur le lecteur.

Le **MoP3-player** de Frank Hickl & Mario Becker [4] est un projet en plein développement. Le coeur du système est un microcontrôleur enfoui bon marché, un MC68HC11. Comme sur nombre d'autres lec-

teurs, le décodage MP3 est confié à un MAS3507D ou à un STA013. Un affichage LCD de dimensions relativement importantes (128 x 64 pixels) facilite la communication avec l'appareil. Les photos présentent un circuit imprimé de taille relativement compacte de sorte que nous pouvons le considérer comme un lecteur portable. Pour le moment, l'appareil utilise un lecteur de CD-ROM, mais il est prévu, dans un avenir plus ou moins proche, de pouvoir également utiliser un disque dur.

YAMPP [5] est un site où l'on travaille dur. YAMPP est l'acronyme de Yet Another MP3 Player. Nous découvrons sur ce site plusieurs projets dont certains sont encore en cours de développement. Tous les lecteurs comportent une connexion vers un disque dur IDE et un affichage. Il est très intéressant d'examiner de près les différents projets et de les comparer vu qu'ils reposent sur plusieurs types de processeurs. Ici aussi, les informations concernant le matériel et le logiciel sont, comme dans la plupart des projets de ce type, librement disponibles.



Audio Affinity 2000 [6], oeuvre d'une équipe d'électroniciens de Floride, est un projet très particulier, qui se trouve cependant encore à ses premiers balbutiements. La spécificité de cet appareil est qu'il doit permettre d'enregistrer et de reproduire en format MP3. Il y a encore peu de choses concrètes sur ce site, mais il n'est pas mauvais de le suivre du coin de l'oeil et de lui rendre visite de temps à autre.

Le **ELM MP3 Player** [7] est un projet qui ressemble quasiment à un produit commercialisé (ou sable), en ce qui concerne le prototype mis au point par son concepteur japonais du moins. La mémoire de ce lecteur prend la forme de cartes de mémoire Smartmedia utilisés couramment sur les lecteurs MP3 ou les appareils photo numériques. Le montage repose sur un microcontrôleur de ATmel et utilise un chipset DSP/DAC de Micronas.

Nous terminons cette revue par un lecteur qui semble être un projet de fin d'études de plusieurs étudiants (Peter D'Antonio et Daniel Riiff). Ce projet a été baptisé **EE 476, Final Project: Portable** MP3 Player [8]; s'il s'agit pour le moment d'un prototype, sa compacité n'en reste pas moins très acceptable. Il est certain que ce projet peut devenir extrêmement intéressant si ses auteurs arrivent à mettre l'ensemble de l'électronique sur une platine de dimensions acceptables. L'une des spécificités marquantes de ce projet est qu'il utilise des cartes de mémoire Flash.

(015065)

Adresses Internet :

[1] MP3Projects: www.mp3projects.com/
[2] MP3 PuBliC: www.mp3ar.com/
[3] MPRJ: www.pjrc.com/tech/mp3/
[4] MoP3: www.geocities.com/SiliconValley/Circuit/3150/index.html
[5] YAMPP: www.myplace.nu/mp3/
[6] Audio Affinity 2000: www.audioaffinity.com/
[7] ELM MP3 Player: http://elm-chan.org/reports/mpc/report e.html
[8] Portable MP3 Player: http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2000/peterdan/final.htm

78K – Test it!

Tout un programme que ce cri du coeur

Nous n'avons pas pu nous empêcher de demander à NEC, son fabricant, de nous faire parvenir un exemplaire de leur nouveau kit de démonstration pour les microcontrôleurs de la famille 78K.

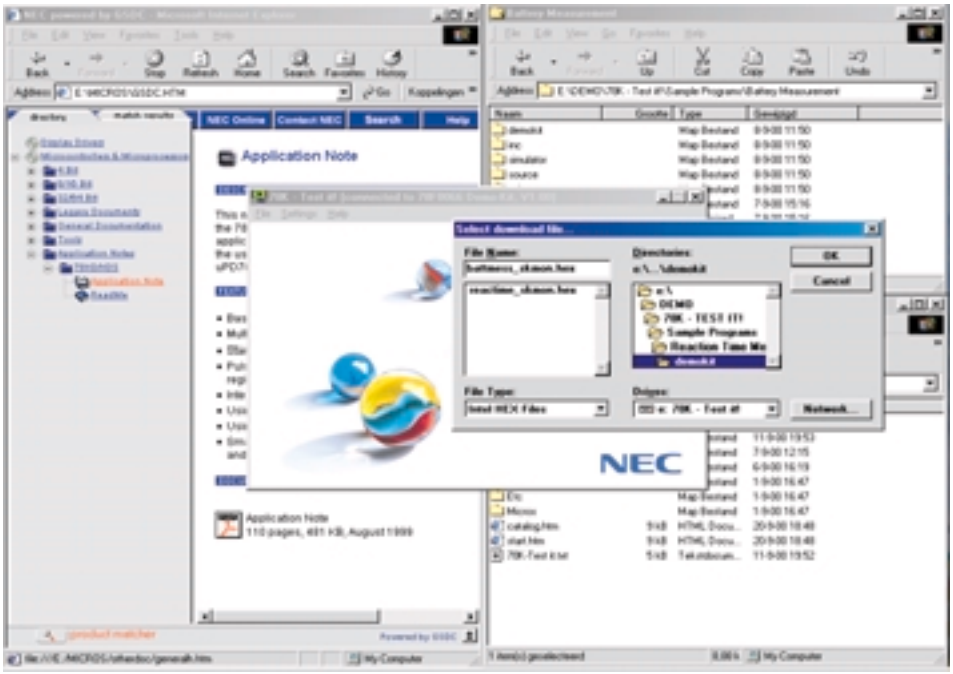
Le kit arrive dans une jolie boîte agrémentée quelques billes, le nouveau logo de ralliement de NEC. Une surprise à son ouverture : elle contient tout, mais vraiment tout le matériel nécessaire, à savoir une platine compacte alimentée par 3 piles montées sur le dessous, un CD-ROM comportant la documentation et l'environnement de développement d'IAR, y compris un compilateur C (limité à 4 Koctets de code), un câble RS-232 pour la connexion au PC ou au portable, sans même parler d'un petit tournevis et de 3 billes...

Un coup d'oeil à la carte permet de constater la puissance de ce nouveau membre de la famille des microcontrôleurs 8 et 16 bits de NEC. La platine ne comporte en effet, outre le microcontrôleur, qu'un unique MAX3222, une paire de quartz (4,194 304 MHz et, nous supposons qu'il s'agit d'un 32 kHz), le tout épaulé par quelques composants passifs en CMS. 3 boutons-poussoirs servent au lancement et à la commande, une huitaine de LED à visualiser l'une ou l'autre expérience. De par la présence des 3 piles, la carte est autonome en fait. Après avoir enlevé le petit morceau de plastique coincé dans le porte-pile qui y fait office d'interrupteur pour voir les LED s' allumer en une sorte de chenillard. Il suffit de connecter la petite carte au port série du PC pour être, quelques secondes plus tard, en mesure de faire tourner l'un des 8 programmes fournis « prêt-à-tourner » avec le kit. Aspect remarquable, il n'est pas nécessaire d'installer quelque logiciel que ce soit.

Après ces premiers balbutiements on pourra passer aux choses sérieuses et se faire la main sur l'environnement de développement d'IAR, l'un des grands spécialistes dans ce domaine, et son compilateur C. Vous pourrez ensuite, si le coeur vous en dit et que vous l'avez bien accroché, tenter de modifier le code source, le compiler et le lier pour disposer d'un code exécutable.

Le coeur de la carte de démonstration est l'un des microcontrôleurs les plus récents de la famille 78K0, le μPD78F0066, composant disposant de 5 Koctets de RAM dont 4 utili-

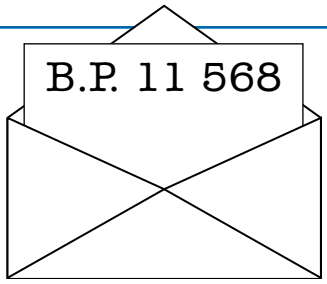
sables pour stocker et exécuter du code utilisateur téléchargé. Il s'agit d'un composants CMS carré à 80 broches qu'il sera sans doute très délicat de souder soi-même, mais ce n'est pas non plus là le but de la manoeuvre. Une faible partie des 48 Koctets de mémoire Flash seulement est requise par le *boot-loader* qui a pour fonction de suivre le téléchargement et l'exécution du code utilisateur. Comme il s'agit de mémoire Flash la reprogrammation de la carte ne pose pas le moindre problème et pourra avoir lieu aussi souvent que nécessaire.
La carte comporte également 2 ports à 10 broches chacun en permettant la connexion de composants analogiques ou numériques.
Le kit est accompagné d'un CD-ROM qui fournit toutes les informations les plus récentes sur les différentes familles de microprocesseurs de NEC. Om y trouve également un programme baptisé 78K - Test it!. Après avoir connecté le kit à l'interface RS-232 du PC et avoir procédé au paramétrage adéquat, on pourra télécharger l'un des 8 fichiers de démonstration, dont un chenillard, un chrono-décompteur (8 mn max), un générateur de mélodies (avec adjonction de résonateur piézo-électrique), etc.
Le processus de téléchargement est illustré par le clignotement « intelligent » des LED (alternance des 4 LED « cardinales » et des 4 LED des coins);



un clignotement de l'ensemble des LED signale la fin du téléchargement. Une action sur START réinitialise la carte; il faut ensuite reconnecter la carte au PC par le point de menu *Settings / Connect*. On peut ensuite passer au téléchargement suivant.
Ce kit ne prétend bien évidemment pas remplacer quelque outil de développement professionnel de NEC que ce soit, ces derniers étant bien plus complets avec leur ICE (*In Circuit Emulator*), l'environnement de développement intégré (IDE - *Integrated Developpment Environment*) d'IAR, compilateur C, débogueur, carte d'interface PCI/PCMCIA, alimentation et documentation technique sur papier.
NEC propose cette carte de démonstration au prix de 49 ¤ qui couvre en fait les frais d'envoi et de manutention.

Pour en savoir plus au sujet des composants de la famille 78K0 et 78K0S, un tour sur le site Internet de NEC à l'adresse : *www.nec.de/78K* s'impose.
Une dernière remarque qui pourrait presque aller de soi. On se trouve ici en présence d'une carte de démonstration et non pas d'un outil de développement qui pourrait servir de noyau pour d'autres applications.

(017076)



Nous ne pouvons malheureusement pas répondre in extenso à toutes les lettres relevant des questions techniques. Dans cette rubrique nous répondons à des lettres pouvant présenter un intérêt général et concernant des montages âgés de moins de 2 ans. Vu le nombre de lettres qui nous arrivent mensuellement, nous regrettons de ne pas pouvoir répondre séparément à chacune d'entre elles et sommes dans l'impossibilité de donner suite à des souhaits individualisés d'adaptation de montages publiés ou de réalisation de montages à publier ni même de répondre à des demandes d'information additionnelle concernant un montage décrit dans Elektor.

Copie des CD-ROM sur disque dur ?

Je trouve très ennuyeux d'avoir à mettre chacun de mes 3 CD de la « bibliothèque numérique » dans mon lecteur de CD-ROM lorsque je veux jeter un coup d'oeil sur l'un des montages. N'est-il pas possible de les installer sur le disque dur pour y accéder directement depuis le disque dur ?
Bob Nijman

Pour peu que votre disque dur ait la capacité suffisante, il n'y a pas le moindre problème à y recopier les CD-ROM. Le seul problème est que nombre de programmes refuseront purement et simplement de fonctionner dans ces conditions. On peut remédier à cette situation par l'installation d'un programme qui simule un CD-ROM sur le disque dur. « Virtual CD-ROM » est un exemple de ce type de programmes.

Transfos de Tesla

J'aimerai bien savoir pourquoi Elektor n'a jamais publié de projet de réalisation d'un transformateur de Tesla. Cela me paraît un sujet très intéressant.
P. Notebaart

Il nous est arrivé d'envisager de publier un article comportant un transformateur de Tesla, mais pour une raison ou une autre, cela n'a rien donné de concret jusqu'à présent. Cela tient au fait que l'aspect sécurité constitue un problème d'une gravité indiscutable. Il ne saurait être question que la réalisation d'un projet décrit dans Elektor puisse entraîner, pour le lecteur qui s'y essaierait, le moindre risque. Selon les termes de la loi, cela engagerait notre responsabilité.

PC-audio

Dans le sillage du nouveau convertisseur A/N que nous vous avons proposé, un petit truc qui pourra peut-être servir à nombre d'audiophiles travaillant sur PC. J'ai eu la chance de pouvoir tester un certain nombre de cartes-son dotées d'entrées et de sorties S/PDIF et force m'a été de constater l'un ou l'autre problème.
Ma première carte fut une Terratec EWS64, qui ne mérite pas la moindre critique mais qui nécessitait un connecteur ISA. J'ai eu ensuite une Terratec DMX. Après enregistrement il apparut que le signal de sortie présentait un niveau inférieur de 3 dB à celui de la reproduction (je parle toujours de signal numériques) décalage très difficile à élimer par voie logicielle. Plus gênant encore, l'existence d'une différence de niveau de 0,5 dB entre le canal gauche et le canal droit.
Ma carte suivante fut une Audioworks2 de Emagic. Avec elle, la qualité du son était OK, mais la priorité des pilotes n'était elle pas idéale. Il n'est pas même possible d'ouvrir le Bloc-note (Notepad) sans que l'enregistrement ou la reproduction ne se mette à souffrir. La ligne de téléphone de support m'a conseillé d'utiliser les pilotes AISO, mais malheureusement rares sont les programmes Windows à les supporter.
Ma dernière carte fut une Di02448-man de M-Audio. Elle n'appelle pas la moindre critique : d'un prix très abordable pour une carte professionnelle et parfaitement stable. Même en cours de scanning d'une photo en Photoshop, cette carte ne commet pas la moindre erreur ni en enregistrement ni en reproduction. Les pilotes les plus récents permettent même une bonne reproduction MIDI.
Mark Wuyts

Nous pensons que certain de nos lecteurs pourront tirer profit de cette expérience, ce qui explique que nous publiions cette lettre avec plaisir.

Soucis d'E/S numériques

J'ai réalisé votre projet 990097 (« entrées/sorties numériques pour SB Live ! Value », Elektor n° 258, décembre 1999, page 56 et suivantes) y compris la mini-platine de l'adaptateur. J'ai connecté le circuit à ma Soundblaster Live! 1024 et ai ensuite procédé à toute une batterie de tests à l'extérieur du PC. J'installe ensuite Liveware 3.0. D'après l'information d'Aide de Liveware je devrai voir apparaître, lors du paramétrage de la carte, un tableau « digital I/O » mais cela n'est pas le cas. Si je procède à des mesures sur les sorties je n'y détecte que du bruit. Que puis-je faire ? Ce projet est-il incompatible avec Liveware 3.0 ?
Raf Smet

Nous pensons avoir affaire ici, à un problème de logiciel typique. D'après son concepteur, le tableau pour « digital I/O » doit toujours apparaître, même sur la SB Live! 1024 est présente sans l'adaptateur numérique. Il vous faudra peut-être réinstaller l'ensemble.

Télécommande RC

Avez-vous jamais publié un projet de télécommande RC (pour un modèle réduit de voiture de course par exemple), ou avez-vous l'intention de publier quelque chose de ce genre dans un avenir proche ?
Kris Coopman

En raison de l'existence de toute une série de règlements nous nous trouvons dans l'impossibilité de procéder à la publication de télécommandes RC travaillant sur les fréquences prévues à cet effet. Il vous faudra partant utiliser les modules tout faits vendus dans le commerce. En septembre 1998 nous avons parlé d'une télécommande sans fil travaillant sur 433 MHz. Cet article pourrait peut-être vous intéresser.

Détecteur de métaux

Vous avez, au cours des années '80, publié un projet de détecteur de métaux. J'avais, à l'époque, trouvé ce projet sensationnel et l'avais réalisé avec grand plaisir. Il devrait être possible, avec la technologie actuelle, de réaliser, en principe, des détecteurs de métaux encore bien meilleurs. Voici partant nma question : avez-vous l'intention, à court ou moyen terme, de décrire un nouveau projet de détecteur de métaux ? Je suis persuadé que cela ferait plaisir à nombre de vos lecteurs.
H. Kunst

En ce qui nous concerne, nous n'avons pas, chez Elektor, de plans allant dans ce sens (à moins qu'il ne nous tombe du ciel un projet d'une qualité telle que nous ne pourrions passer outre...). Il est bien prévu la parution très prochaine, en langue allemande seulement malheureusement, d'un ouvrage consacré aux détecteurs de métaux. Il y sera fait référence très prochainement sur notre site Internet.

Télécommande par GSM

Il devrait être possible, étant donné, actuellement, le prix dérisoire des GSM (à carte prépayée), de commander à distance, par le biais de 2 téléphones GSM et d'un décodeur de tonalités DTMF, toute sorte d'appareils domestiques ou embarqués à bord de véhicules. Vous pourriez, par exemple, être averti du déclenchement de l'alarme dans votre voiture, ou en cours de route ou au travail, de l'entrée en fonction du chauffage central. Il devrait être possible de se contenter d'entrer le numéro du GSM suivi d'un code d'entrée et d'un code de commande. Pourriez-vous réfléchir à cette idée ?
W. Wubs

Nous pensons qu'il s'agit là d'une idée méritant d'être examinée de plus prêt. Qui sait, peut-être qu'elle pourra se traduire en un montage fonctionnel. Merci en tout cas de votre suggestion.

Mini-serveur Web

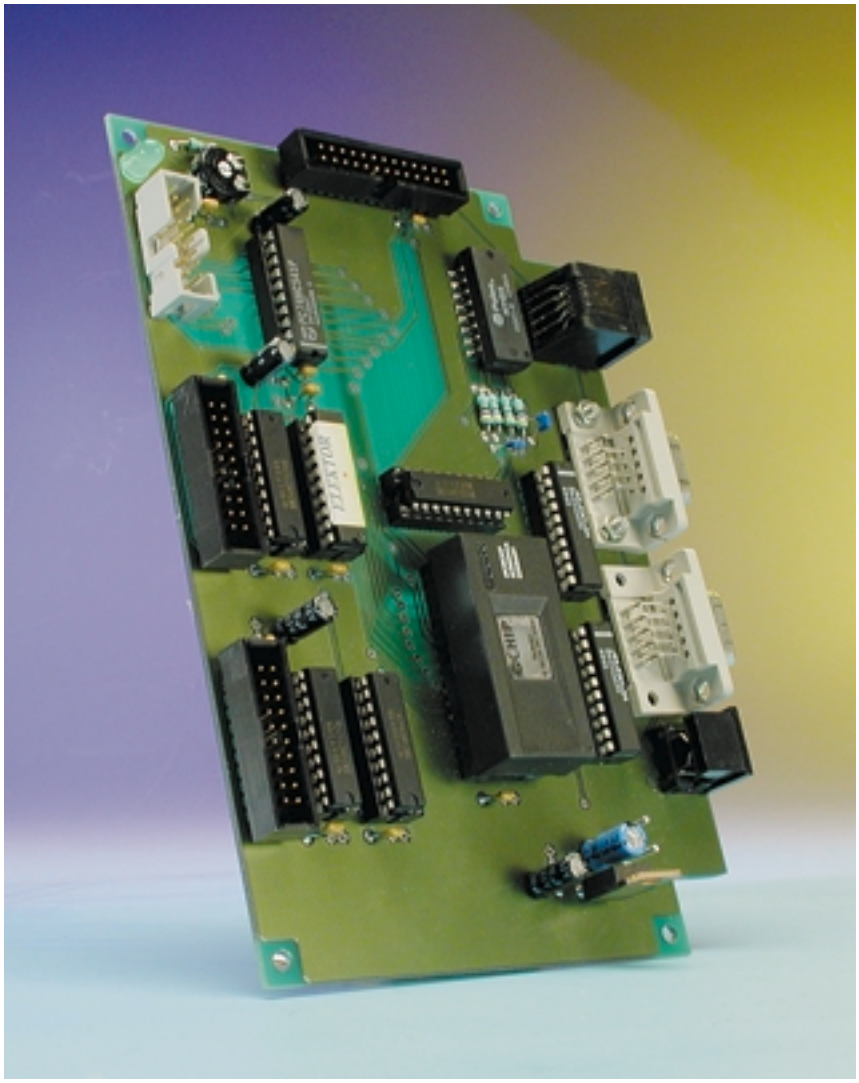
3ème partie : carte de développement à PC enfoui

Le coeur de ce mini-serveur Web prend la forme d'un PC enfoui (*embedded-PC*) proposé dans un boîtier DIL. Dans le cadre de cet article nous vous présentons une carte d'évaluation pour ce contrôleur et verrons comment l'utiliser. À la fin de cet exposé vous devriez être en mesure de piloter n'importe quel appareil électronique par le biais de ce mini-serveur Web, que ce soit par la voie d'Internet ou par celle d'Ethernet.

Il devient de plus en plus fréquent d'avoir l'impression, lorsque l'on est un électronicien passionné de réalisations personnelles, que l'électronique moderne ne convient plus à ce genre d'aspirations ô combien captivantes. Les microcontrôleurs ont pris trop d'embonpoint et de pattes, le circuit intégré DSP requiert, pour être doté de son progiciel, un programmateur professionnel hors de prix et les composants passifs sont devenus si minuscules qu'il faut une loupe pour les retrouver, sans parler d'une quasi-impossibilité de les souder, sans même insister lourdement sur le fait que la réalisation d'une platine multi-couches est loin d'être à la portée de Mr-Tout-Le-Monde.

Le mini-serveur Web objet de cet article prouve que les choses peuvent également se présenter différemment. Ce minuscule montage facile à réaliser, qui permet de télécommander un appareil électronique, aussi loin que porte Internet, est actuellement un « hot item » et n'existe pas encore sous la forme d'un système produit en masse par l'industrie que l'on peut acheter, pour quelques euros, dans tout magasin vendant de l'électronique ou de la « téléphonique ».

Nous nous sommes intéressés, dans les numéros de mai et de juin derniers de ce magazine, aux tenants et aboutissants relativement complexes d'une communication par le biais d'Internet ou d'un réseau local. Le contrôleur « planté » au coeur du mini-serveur Web doit être « prédestiné» à pouvoir s'accommoder de toute une série de protocoles



Caractéristiques du mini-serveur Web

– CPU 16 bits du type 80186 tournant à 20 MHz
– 512 Koctets de RAM, 512 Koctets de Flash
– RTOS à système de données en Flash
– Téléchargement de programmes via RS-232 ou Ethernet
– Protocoles : TCP/IP, PPP, HTTP, FTP, Telnet, POP3, SMTP,ICMP et DHCP
– Connexion du modem : PPP par connecteur Sub-D à 9 contacts
– Ethernet : 10Base-T avec PHY par embase RJ45
– Connexion vers terminal : port sériel rapide avec RS-323 (RXD, TXD, CTS, RTS) par le biais d'un connecteur Sub-D à 9 contacts
– Bus I²C (maître) : par le biais d'une embase mini-DIN à 6 contacts
– Chien de garde
– 2 sorties et 2 entrées temporisateur (timer)
– Détection de disparition de la tension d'alimentation (NMI) avec sauvegarde des données
– Bus A/D multifonctionnel compatible Intel
– 16 entrées numériques
– 16 sorties numériques
– Possibilité de connexion directe d'un affichage LCD
– Connecteur d'extension adressable (bus A/D et I²C-Bus, Interrupt, ALE et RD/WR)

Fonctions de la CPU

– Bus d'adresse/de données multiplexé (8 bits)
– 6 signaux de sélection de puce (chip-select) programmables
– 6 entrées interruptibles
– Possibilité de connexion d'un contrôleur d'interruption externe
– Chien de garde hardware à double niveau
– 3 temporisateurs
– 2 entrées de comptage rapides
– 2 sorties de comptage rapides
– Contrôleurs DMA
– Interface 10Base-T
– 2 interfaces sérielles asynchrones
– 1 interface sérielle synchrone (SSI)
– Capacité de travailler en maître de bus I²C
– Jusqu'à 14 broches d'E/S programmables
– Générateur de RAZ/Power-fail (NMI)

de communication.
Implémenter TCP/IP, HTTP, FTP, SMPT, systèmes d'exploitation, sans parler du reste dans un microcontrôleur n'est pas une mince affaire.
Il existe heureusement dans le commerce quelques (rares) types de microcontrôleurs qui embarquent les fonctions requises sur leur puce (*on-chip*) ou du moins les possèdent intégrées dans un petit module ce qui permet de ne plus avoir à se casser la tête en ce qui concerne l'aspect réseau du serveur et que l'on peut partant se concentrer uniquement aux applications.
Et c'est très exactement là la raison d'être et le but du matériel proposé ici. Il a été conçu à dessein sous la forme d'un système de développement et met à disposition, par le biais d'embases et de connecteurs, toutes les interfaces et ports disponibles du microcontrôleur utilisé. Rien n'interdit bien évidemment non plus de n'implanter sur la carte de développement que les composants indispensables et de s'en servir comme prototype lorsque l'application en question laisse la latitude de cette solution.
La platine comporte une embase de connexion à laquelle on pourra connecter l'application ou encore une extension pour la carte de développement dotée, par exemple, de convertisseurs N/A et A/N, mais également d'autres composants servant à réaliser l'une ou l'autre interface.

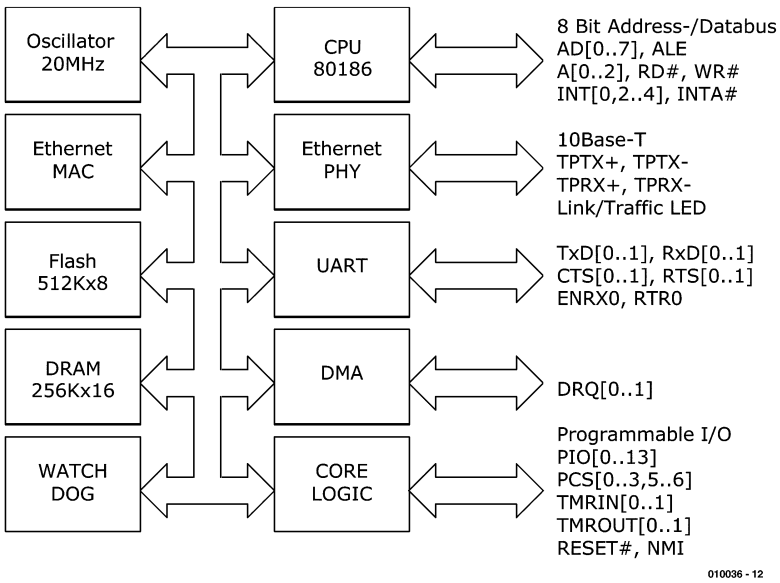


Figure 1. Structure interne du PC mono-puce (Single-Chip-PC) IPC@CHIP SC12.

PC-Internet mono-puce

Le IPC@CHIP SC12 de la société allemande Beck est l'un des rares microcontrôleurs « enfouis » (*embedded*) sur lequel le commun des mortels, vous et nous, puisse mettre la main. Le SC12, comme nous l'appelons affectueusement, est proposé dans un boîtier DIL à 32 broches. Il repose, comme l'illustre le synoptique de la structure interne de la **figure 1**, sur un noyau de processeur 80186 travaillant à une fréquence d'horloge de 20 MHz et dispose de 512 Koctets de DRAM et d'autant de mémoire Flash. Cette version de base a été dotée de fonctions additionnelles (puissantes) telles que communication sérielle par le biais d'un UART, une interface I²C et une interface de bus directe, sans par-

ler de l'adjonction de 64 Koctets de mémoire.
Le SC12 permet une communication Ethernet (10Base-T) par le biais d'une paire torsadée (*twisted pair*) avec l'ensemble des commandes d'accès (*Medial Access Controls* = MAC) selon la norme IEEE802.3.
Le SC12 respecte une architecture ISA propre modifiée en certains points mais sans bus ISA ni interface vidéo. De plus, la programmation des ports sériels, des DMA, PIC et temporisateurs (*timer*) diffère de ce que l'on trouve sur un PC du type IBM Standard.
L'un des encadrés de l'article récapitule les fonctions CPU les plus importantes. Le composant intègre un système d'exploitation multi-tâche connu sous la dénomination de RTOS et dont la présentation rappelle celle du DOS tant apprécié (???) il y a quelques lustres (déjà !).
Ce n'est pas sans raison que la société Beck appelle le résultat de cette recette, un PC mono-puce qui s'accommode de tous les outils de développement prévus pour le microcontrôleur 80C186 et tourne les applications prévues pour ce dernier. Cela signifie en clair que chacun d'entre vous ayant déjà eu l'occasion de programmer un microcontrôleur de ce type en C, Pascal ou Assembleur, ne devrait pas avoir de problème avec le SC12. Autre aspect très intéressant : il existe des nombre d'outils de développement pour le 80186 au prix abordable, sans oublier un système d'exploitation compatible DOS.
Si, en dépit de tout cela, vous n'arrivez pas à créer un programme fonctionnant, sachez qu'il vous est proposé nombre d'applications et de programme (avec codes-source) pouvant servir d'exemples pour les fonctions de base du SC12 sur les sites Internet de la société Beck et sur le nôtre.
On peut difficilement s'attendre à trouver un circuit intégré aussi performant et à la pointe de la technologie pour pratiquement rien. Le composant SC12 est disponible, à la date de mai 2001, pour 66 euros, un compilateur-C de Borland étant proposé (en combinaison avec l'acquisition d'un microcontrôleur) au prix de 49 euros. Il faut ajouter à cela une licence individuelle Runtime pour le système d'exploitation RTOS avec DOS multi-tâche, pile TCP/IP, serveur WSeb avec interface CGI, serveur FTP et serveur Telnet, ainsi que API pour programmation en C, 25 euros supplémentaires, ceci cependant pour les applications professionnelles uniquement. Étant donnés la simplicité relative du matériel et du logiciel d'une part et de l'autre les possibilités offertes par un tel système, nous pensons qu'il s'agit là d'un investissement acceptable. Notons au passage que la société Beck propose en outre, sous la dénomination de DK40, une petite platine de

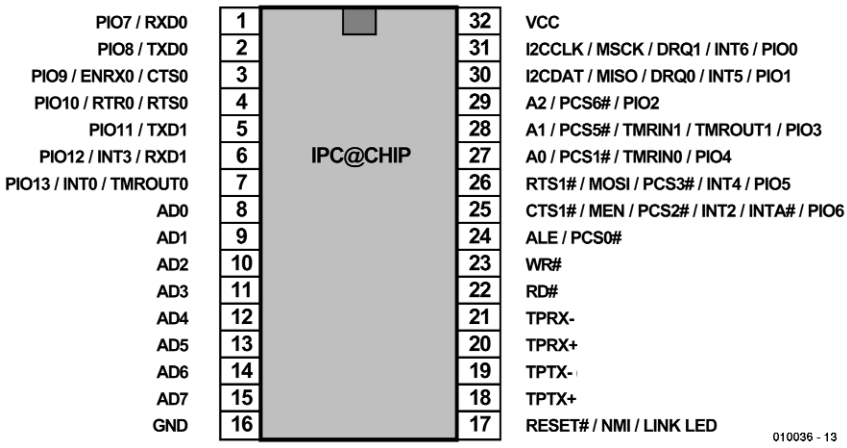


Figure 2. Nombre des broches du SC12 remplissent plusieurs fonctions.

développement qui ne comporte cependant que des interfaces TTL (à embases RJ12/45) et 8 lignes d'E/S tamponnées.

Bus et interfaces

Le SC12 n'est pas indissociablement imbriqué avec le matériel de la carte de développement. Il n'est pas requis impérativement qu'il faille définir une interface sérielle comme port (TxD0, RxD0) pour le terminal à des fins de déboguage. On pourra également se passer de la connexion d'un modem dès lors que l'application envisagée travaillera exclusivement avec Ethernet. Dans ces conditions on aura également la seconde interface sérielle à sa disposition.
Côté application, nombre des lignes de port du SC12 sont multi-fonctionnelles. Ainsi, si l'on n'a que faire du bus I²C, rien n'interdit de programmer librement les lignes de port correspondantes voire de les utiliser en

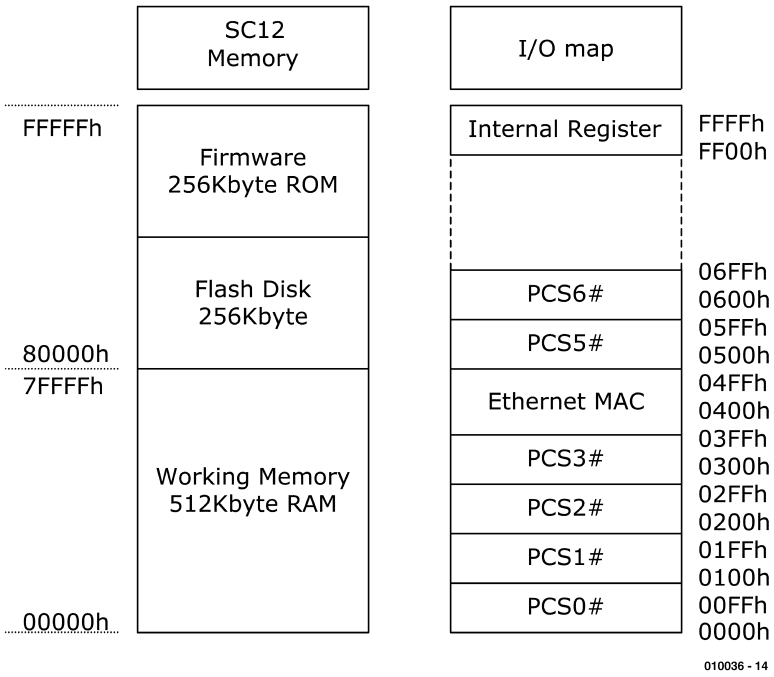


Figure 3. Cartographie de la mémoire et attribution des adresses d'E/S de base.

tant que sources d'interruption.
Ces affirmations sont autant d'arguments pour examiner d'un peu plus près les fonctions des différentes broches. Le croquis de la **figure 2** représente le brochage du SC12 où l'on découvre nombre de fonctions doubles voire triples.

Bus d'adresses/de données
La première structure à sortir de la grisaille est le bus d'adresses et de données multiplexé, **AD0** à **AD7**, au fonctionnement synchrone et travaillant sur niveaux. On y voit apparaître, au cours de la première période d'un cycle de bus, les 8 bits de poids faible d'une adresse suivis de 3 périodes de données. Il est possible de supprimer la phase concernant l'adresse.
Le bus d'adresses est complété par les sorties d'adresse synchrones non multiplexées **A0** à **A2**. L'adresse est valable pendant une durée d'une demi-période d'horloge pour AD0 à AD7.
La sortie de validation de verrou d'adresse **ALE** (*Address Latch Enable*) signale, par son flanc avant, la présence d'une adresse valide sur le bus d'adresses/de données combiné. Les lignes **RD** et **WR** autorisent respectivement les processus de lecture (Read) et d'écriture (Write).

Lignes d'E/S programmables
Les 14 lignes **PIO0** à **PIO13** sont, au niveau des attributions de fonction, direction et niveau actif, des lignes de port à drain ouvert, programmables et travaillant de façon synchrone. Après la réinitialisation après mise sous tension (POR = *Power-on-Reset*), le paramétrage adopté est celui par défaut de sorte qu'il faudra reconfigurer les paramètres par logiciel pour qu'ils soient ceux que l'on requiert.

Signaux de sélection de puce (CS) programmables
Les sorties de sélection synchrones **PCS0** à **PCS6** servent à permettre un accès d'E/S sur une mémoire périphérique. Les sorties PCS travaillent en association avec le bus d'adresses AD multiplexé.

Interruptions et Temporisateur
Les broches **INT0**, **INT2** à **INT4** sont chacune une entrée d'interruption asynchrone masquable. En cas d'apparition d'une interruption sur une entrée non masquée, on aura interruption du programme au profit de la routine sur laquelle pointe le vecteur d'interruption correspondant. La synchronisation des interruptions se fait en interne, leur déclenchement pouvant se faire soit par niveau soit par flanc. INT1 est attribuée au Ethernet-MAC et n'est pas accessible de l'extérieur.
Si INT0 travaille en mode cascade, ce qui implique partant la connexion de plusieurs sources d'interruption au travers d'un encodeur de priorité, l'entrée INT2 se transforme en une sortie synchrone, **INTA**. En cas d'activation de INTA, le matériel doit placer une valeur à 8 bits sur le bus de données de manière à permettre au contrôleur d'identifier la source d'interruption.
Le microcontrôleur dispose de 2 temporisateurs (*timer*) internes accessibles de l'extérieur par le biais des entrées **TMRIN0** et **TMRIN1**. Après la synchronisation interne, chaque flanc montant appliqué à l'entrée TMRIN incrémente le compteur correspondant. Si on n'utilise pas l'entrée TMRIN, il faudra, en externe, la forcer au niveau haut.
Les sorties de temporisateur **TMROUT0** et **TMROUT1** mettent à disposition soit une impulsion unique soit un signal continu dont le rapport cyclique est programmable. Ce dernier signal est le résultat d'une combinaison en interne, à chaque fois, de 2 entrées d'interruption et de temporisateur en mode DLI (**D**émodulation en **L**argeur d'**I**mpulsion (**PWD** = *Pulse Width Demodulation*). PWD pilote d'une part TMRIN0 et INT2 et, en inversé, TMRIN1 et INT4. En cas de validation de INT2 et INT4 et de configuration correcte des temporisateurs, le rapport cyclique correspond au rapport des valeurs des 2 temporisateurs. En mode DLI (PWD), TMRIN0, TMRIN1 et INT4 peuvent être utilisés comme des PIO. Ceux d'entre eux qui ne seraient pas utilisés seront purement et simplement ignorés par le progiciel du microcontrôleur.

Interface Ethernet 10Base-T
Le SC12 est doté d'un transceiver/contrôleur Ethernet complet accessible au travers des entrées **TPRX+** et **TPRX–** et les sorties **TPTX+** et **TPTX–**. L'émetteur/récepteur (*transceiver*) émet et reçoit des signaux par le biais d'un transmetteur de sortie et d'une liaison par paire torsadée (*twisted pair*).
Une LED prise à la sortie **LINK LED** est allumée faiblement en cas de présence d'une liaison Ethernet, s'allumant très visiblement en cas de transmission de données.

Ports sériels asynchrones
Le SC12 permet la transmission full duplex de 2 interfaces sérielles asynchrones disposant de 7 à 9 bits à des taux de transmission pouvant aller jusqu'à 115-200 bauds par l'intermédiaire des lignes **TxD0**, **TxD1** et **RxD0** et **RxD1**. **CTS0**, **CTS1**, **RTS0**, **RTS1**, **ENRX0** et **RTR0** permettent une sélection libre d'un acquittement (*handshake*) matériel pour chacun des ports.

Accès direct à la mémoire (DMA)
Un composant externe signale, par le biais de l'entrée de demande d'accès DMA (*Direct Memory Access*), **DRQ0** ou **DRQ1**, une transmission au travers de l'un des 2 canaux DMA. DRQ0 est à déclenchement par flanc et à synchronisation interne, DRQ1 étant déclenchée par niveau. DRQ n'est pas verrouillé (*latched*) et devra partant être actif pendant la transmission.

Interface périphérique synchrone (SPI)
La SPI (*Synchronous Peripheral Interface*) avec ces lignes **MEN**, **MSCK**, **MISO** et **MOSI** n'est malheureusement pas encore implémentée.

Bus inter-IC
Le SC12 dispose d'une interface I^2C par le biais des lignes d'horloge **I2CCLK** et de données **I2CDAT**. Le microcontrôleur est en mesure de piloter un maximum de 127 esclaves externes, étant impérativement toujours lui-même le maître.

RAZ, générateur Power Fail
Un niveau appliqué à la broche **RESET** asynchrone tombant en-deçà de 0,8 V produit immédiatement une cessation de toutes les activités en cours, réinitialise la mémoire interne et ordonne à l'unité centrale (la CPU) de sauter à l'adresse $FFFF0_{HEX}$. En cas de chute de la tension d'alimentation du système à une valeur inférieure à 4,5 V ou de déclenchement du chien de garde (*watchdog*), la ligne de RAZ est forcée au niveau de la masse.
L'entrée **NMI** (*Non-Maskable Interrupt*) synchrone et déclenchée par flanc est la source d'interruption possédant la priorité la plus élevée. Elle est, pour INT, totalement indé-

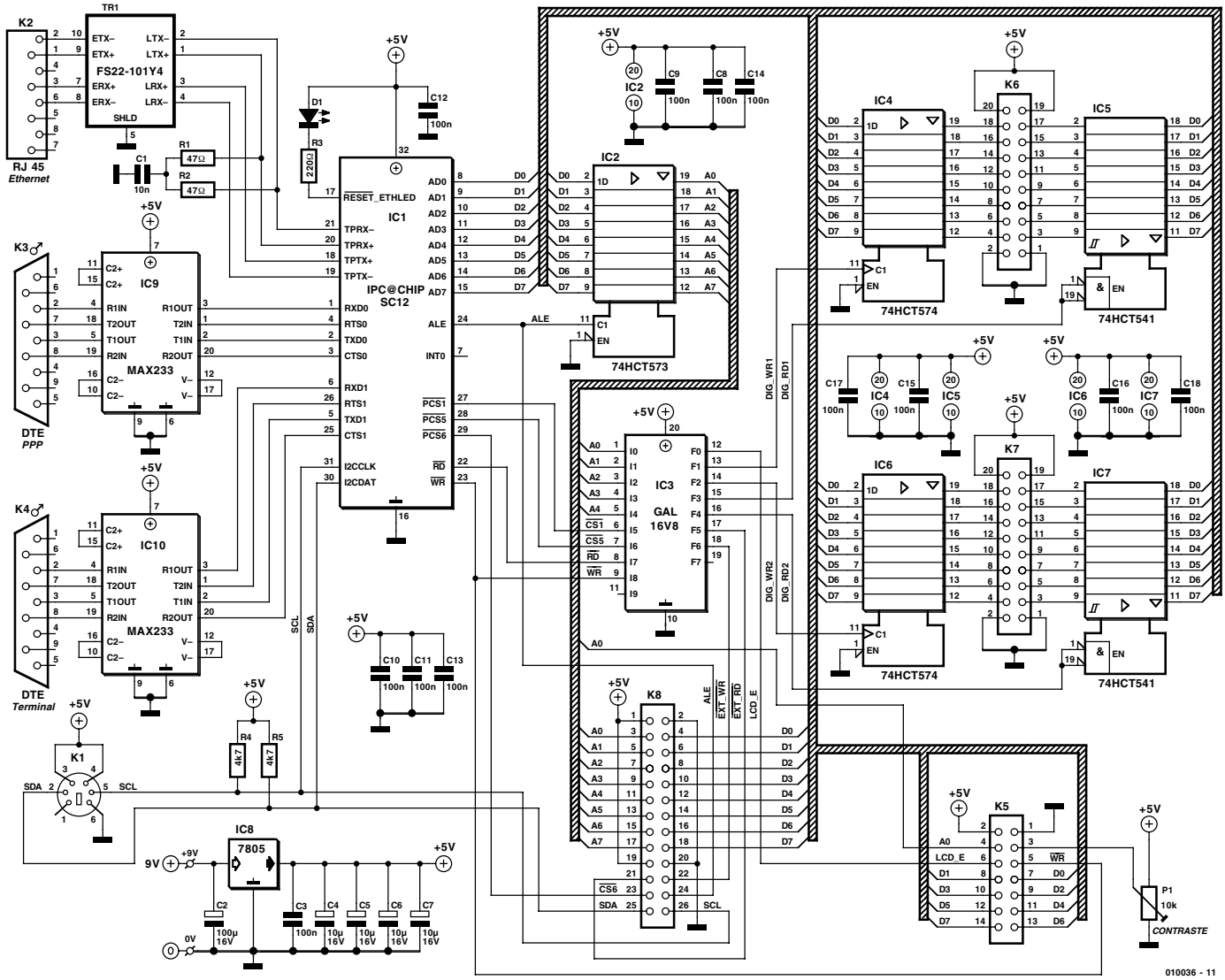


Figure 4. L'électronique de la carte de développement pour le SC12.

pendante du contrôleur d'interruption. Une condition NMI interrompt l'exécution du programme qui saute à l'endroit du programme désigné par le vecteur NMI. NMI est utilisé en combinaison avec le chien de garde interne.

En guise de conclusion à la description du brochage nous vous proposons en **figure 3**, une cartographie de la mémoire et les adresses d'E/S de base.

Sur le plan matériel

Le schéma de principe du mini-serveur Web reproduit en **figure 4** montre enfin quelles sont, sur la carte de développement, les fonctions attribuées aux broches si souples d'utilisation du SC12. Le verrou d'adresses IC2 se charge de démêler l'écheveau que constitue le bus d'adresses/de données multiplexé combiné, AD. Les bits d'adresse A0 à A4 arri-

Liste des composants

Résistances :

R1,R2 =47 Ω
R3 = 220 Ω
R4,R5 = 4kΩ7
P1 = ajustable 10 kΩ

Condensateurs :

C1 = 10 nF
C3,C8 à C18 = 100 nF
C4 à C7 = 10 µF/16 V vertical
C2 = 100 µF/16 V vertical

Semi-conducteurs :

D1 = LED verte
IC2 = 74HCT573
IC4,IC6 = 74HCT574
IC3 = GAL16V8 (programmée **EPS010036-31**)
IC5,IC7 = 74HCT541
IC8 =7805
IC9,IC10 =MAX233CPP (Maxim)
IC1 = IPC@CHIP SC12 (à commander auprès de Beck : http://www.bcl-online.de)

Divers :

PC1,PC2 = picot
TR1 = FS22-101Y4 (à commander auprès de Beck : http://www.bcl-online.de)
K1 = embase mini-DIN 240 ° encartable à 6 contacts
K2 = embase RJ45 encartable
K3,K4 = embase Sub-D à 9 contacts mâle encartable en équerre
K5 = embase autosécable à 2 rangées de 7 contacts
K6,K7 = embase autosécable à 2 rangées de 10 contacts
K8 = embase autosécable à 2 rangées de 13 contacts
Boîtier

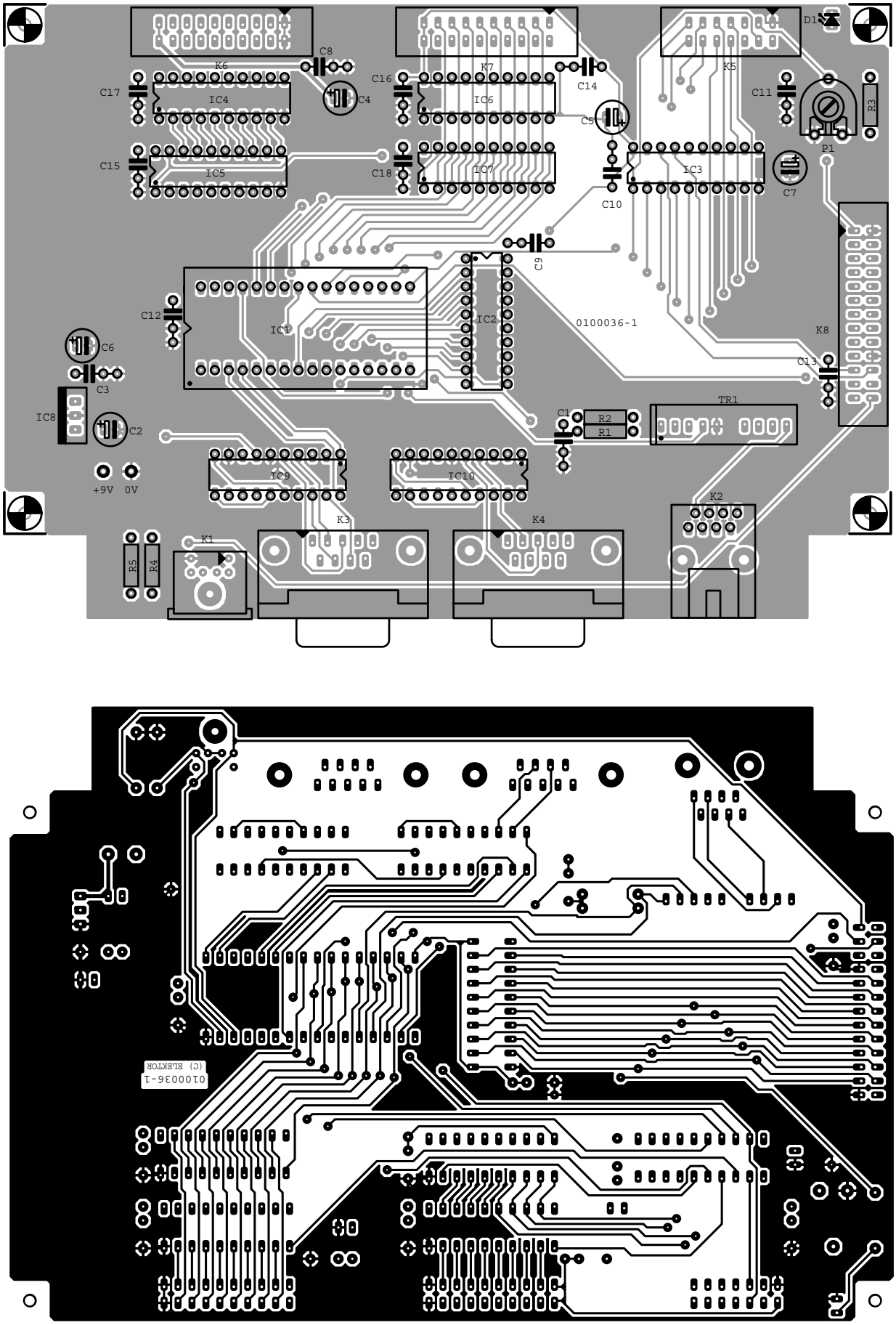
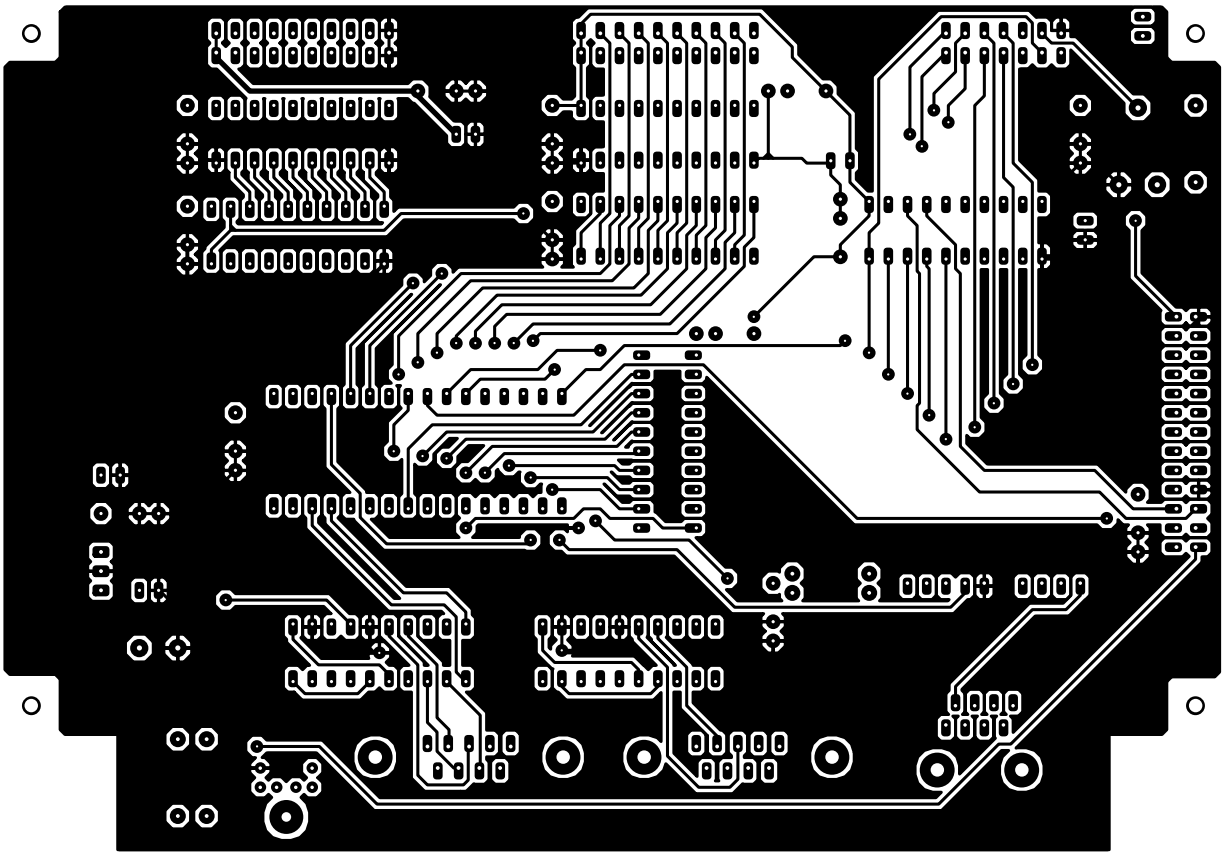
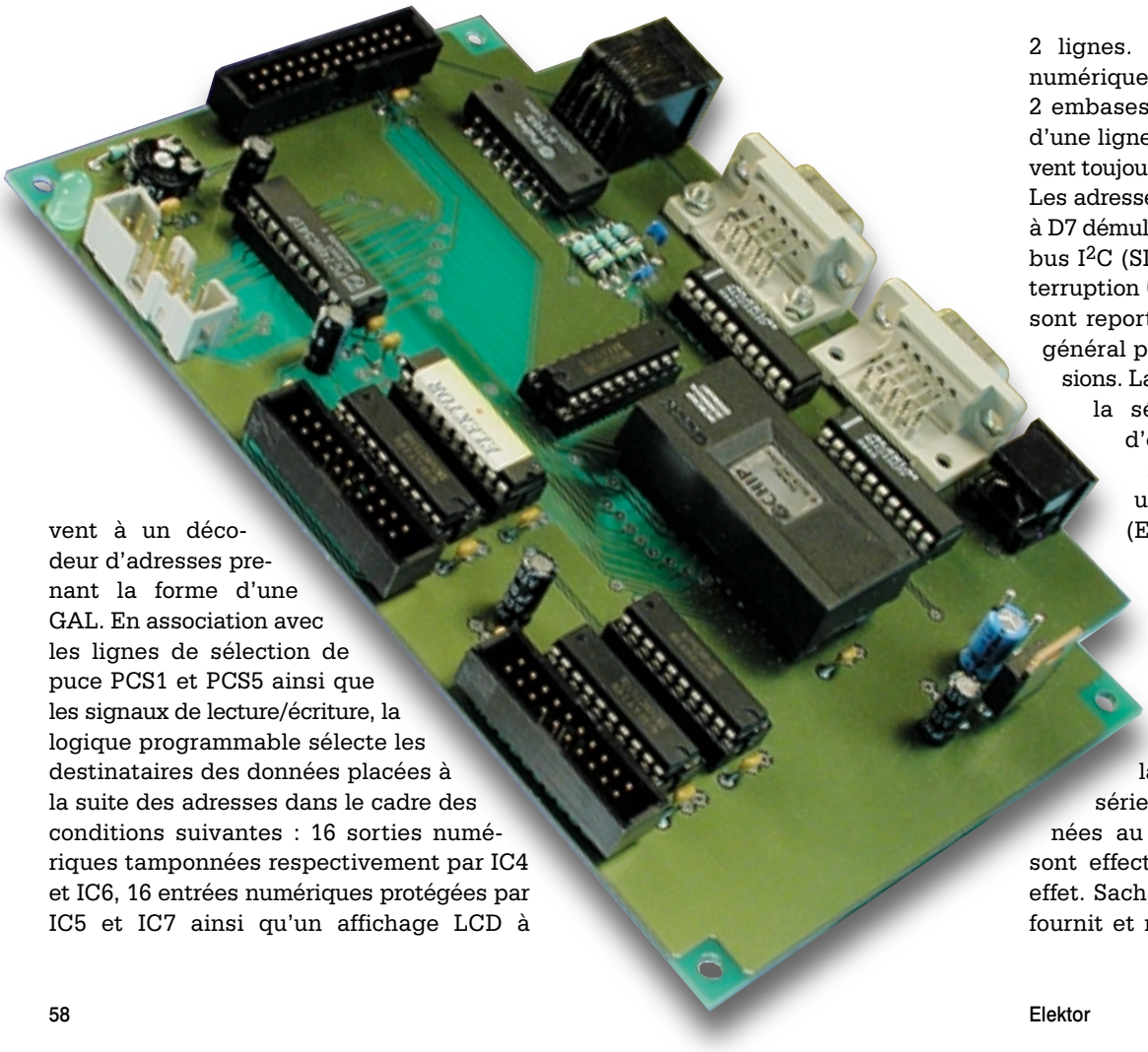


Figure 5. Ce n'est pas trop la bousculade au niveau de cette carte double face à trous métallisés.





vent à un décodeur d'adresses prenant la forme d'une GAL. En association avec les lignes de sélection de puce PCS1 et PCS5 ainsi que les signaux de lecture/écriture, la logique programmable sélecte les destinataires des données placées à la suite des adresses dans le cadre des conditions suivantes : 16 sorties numériques tamponnées respectivement par IC4 et IC6, 16 entrées numériques protégées par IC5 et IC7 ainsi qu'un affichage LCD à 2 lignes. Les entrées et sorties numériques sont reportées sur les 2 embases K6 et K7, où les lignes d'une ligne de port donnée se trouvent toujours l'une en face de l'autre. Les adresses A0 à A7 et données D0 à D7 démultiplexées de même que le bus I^2C (SDA, SLC), la source d'interruption 0 (INT0) et le signal ALE sont reportées vers un connecteur général pour d'éventuelles extensions. La GAL effectue également la sélection du connecteur d'extension, sélection faisant la distinction entre un processus d'écriture (EXT_WR) et de lecture (EXT_RD). La tension d'alimentation de +5 V est également présente sur le connecteur d'extension.

De l'autre côté du SC12, c'est le standard qui fait la loi : les 2 interfaces sérielles asynchrones destinées au modem et au terminal sont effectivement utilisées à cet effet. Sachant que le contrôleur ne fournit et n'accepte de traiter que

des niveaux TTL, on ne sera guère étonné de trouver, sous la forme de IC9 et IC10, une paire de convertisseurs d'interface du type MAX233 (similaire au MAX232 mais sans condensateurs électrochimiques externes), de façon à disposer, sur l'embase sub-D à 9 contacts, de signaux de niveaux RS-232 en sortie, et de pouvoir y appliquer en entrée des signaux de même type. Notons au passage qu'il s'agit, dans les 2 cas, d'interfaces du type DTE (*Data Terminal Equipment* = périphérique) ce qui signifie qu'il faudra, pour le PC, utiliser un câble en modem zéro et un câble 1:1 « normal » pour le modem.

Il est impératif d'assurer, entre le contrôleur/*transceiver* Ethernet intégré dans le microcontrôleur et le réseau, une terminaison correcte des signaux différentiels ainsi qu'une isolation galvanique efficace. C'est là la fonction remplie par R1, R2, C1 et le transformateur spécial, Tr1. La broche commune de Reset, NMI et LINK-LED sert à la commande d'une LED qui visualise l'activité ayant lieu sur l'interface Ethernet.

Nous avons en outre pensé aux nombreux « fans de l'I²C » que compte le lectorat d'Elektor ce qui explique que nous ayons doté l'interface I²C de résistances de forçage au niveau haut (*pull-up*) et d'une embase mini-DIN, K1, de sorte qu'il suffit de connecter des applications I²C existantes décrites précédemment dans ce magazine.

La régulation de la tension d'alimentation se fait par le biais d'un régulateur de tension +5 V intégré épaulé par quelque condensateurs de découplage. L'alimentation pourra se faire à l'aide d'un adaptateur secteur fournissant, sous 9 V, le courant suffisant, à savoir 400 mA au minimum. Il faudra bien entendu tenir compte également de la consommation de courant de l'application connectée à la carte de développement au cas où elle aussi devrait être alimentée par l'alimentation.

Le dessin de la platine double face à trous métallisés est adapté à ce que l'on peut attendre d'une carte de développement : disposition claire et pas trop serrée, un maximum des connexions étant disposées sur le bord de la carte. C'est en vain que vous tenterez de trouver des ponts de câblage ou des composants CMS. La technique de soudure reste classique.

L'implantation des composants ne devrait pas poser de problème elle non plus : tous les circuits intégrés pourront prendre place dans des supports. Après vous être assuré, tout comme dans le cas des condensateurs électrochimiques, de ne pas avoir commis d'erreur dans l'orientation des circuits intégrés et avoir jeté un coup d'oeil critique à votre travail, il ne vous reste plus qu'à attendre le numéro prochain d'Elektor. En effet, il verra la première entrée sur scène du mini-serveur Web. Nous y verrons, entre autres choses, comment utiliser le programme de terminal et comment établir une liaison Ethernet, http ou ftp. Nous verrons également, à l'aide d'un certain nombre d'applications, comment résoudre certains problèmes tant matériels que logiciels. Vous pourrez toujours, d'ici là, si vous n'avez pas la patience d'attendre, faire un tour sur le site Internet sis à l'adresse :

http://www.bcl-online.de

où vous découvrirez nombre de documents en langue anglaise (et allemande) concernant ce thème et le microcontrôleur SC12.

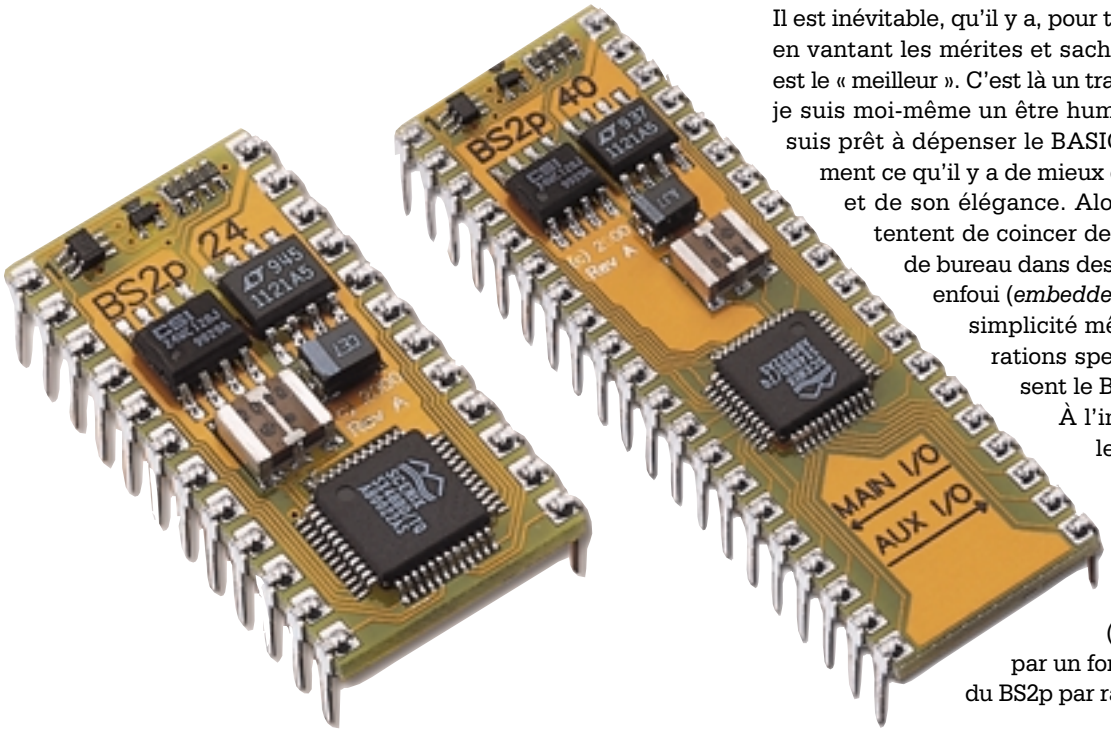
(010036)

Plus de puissance et de meilleures perfos

Rendez-vous avec le BS2p de Parallax

Jon Williams jonwms@aol.com

On pourrait s'attendre, en raison du succès incontestable du BASIC STAMP et sa domination du marché des petits microcontrôleurs travaillant en BASIC, à ce que Parallax s'asseye sur ses lauriers et jouisse des fruits des efforts entrepris. Visiblement, ce n'est pas le cas. Si l'on en croit l'introduction du dernier modèle de BASIC Stamp, le BS2p, Parallax est loin de s'endormir.



Il est inévitable, qu'il y a, pour tout produit, un utilisateur en vantant les mérites et sachant expliquer pourquoi il est le « meilleur ». C'est là un trait de la nature humaine et je suis moi-même un être humain. Pour l'argent que je suis prêt à dépenser le BASIC Stamp est tout simplement ce qu'il y a de mieux en raison de sa simplicité et de son élégance. Alors que ses compétiteurs tentent de coincer des langages d'ordinateurs de bureau dans des microcontrôleurs de type enfoui (*embedded*), Parallax a opté pour la simplicité même en dépit des améliorations spectaculaires qui caractérisent le BS2p.

À l'image des BS2sx et BS2e, le BS2p est un module multi-programmes basé sur le microcontrôleur SX de Ubicom (ex-Scenix). Des améliorations au niveau du progiciel (firmware) se traduisent par un fonctionnement plus rapide du BS2p par rapport au BS2sx d'environ

Note de la Rédaction La taille totale des listages 3 à 7 (eux aussi en anglais) est telle que nous n'avons pas pu l'inclure dans le cadre de cet article. Tous les listages de programme dont il est question dans le présent article sont télédéchargeables de la rubrique Téléchargements du site d'Elektor sur Internet sis à l'adresse : www.elektor.presse.fr. Le numéro du fichier concerné est 014232-11.

20%, soit de l'ordre de 12 000 IPS (instructions par seconde) tout en consommant approximativement 33% de courant en moins, caractéristiques que ne manqueront pas d'apprécier ceux d'entre vous qui ont la robotique comme violon d'Ingres mais dont profiteront également toutes les autres applications alimentées par piles ou accus.
Voici résumées rapidement les additions et les améliorations apportées au BS2p :
– Routines LCD en parallèle pour le Hitachi HD44780
– Routines I²C Philips
– Routines Dallas 1-Wire™
– Interrogation de broches entre instructions
– Tamponnage sériel vers de la RAM bloc-note (*scratchpad*)
– Versions 24 broches (16 E/S) et 40 broches (32 E/S)
– Dispose d'une RAM presse-papier double de celle du BS2sx (127 octets maintenant)

Si vous utilisez le BS2 ou le BS2sx depuis un certain temps vous ne manquerez pas d'admettre qu'il s'agit là d'une liste de caractéristiques extrêmement intéressantes et puissantes qui augmentent très sensiblement l'universalité du Stamp. Prenons le temps de passer en revue ces nouvelles fonctions.

Support LCD

Le BS2p supporte, tel quel, le très populaire contrôleur d'affichage à cristaux liquides (LCD = Liquid Cristal Display) HD4470 LCD de Hitachi. Les routines qui supportent la commande LCD sont :

– LCDCMD *E-pin, command*
– LCDOUT *E-pin, command, [output data]*
– LCDIN *E-pin, location, [input data]*

Ces routines requièrent de l'affichage LCD qu'il soit configuré en mode d'opération bits et ont des exigences spécifiques en ce qui concerne les positions admissibles de connexions. La syntaxe de chaque instruction (la broche de commande pour LCD.E en particulier) indique au BS2p la technique de connexion utilisée pour le LCD.
Ce tableau montre que l'affichage pourra être connecté aux broches de OutL (0 à 7) ou aux broches OutHH (8 à 15) et mentionne des exigences spécifiques quant à la connexion requise des lignes E, R/W, RS et de données. Rappelez-vous qu'il n'est pas nécessaire de connecter la ligne R/W de l'affichage si vous n'avez pas l'intention d'en lire la RAM. Il suffira dans ce cas-là de mettre la broche LCD.R/W de l'affichage à la masse et d'utiliser la broche de commande du Stamp pour d'autres fonctions. Notez que la documentation Parallax suggère de forcer la ligne LCD.E à la masse à l'aide d'une résistance de 4kΩ7.

La première instruction LCD est LCDCMD; elle sert à transmettre un code de commande vers l'affichage LCD. Cette instruction servira lors de l'initialisation, pour la remise du curseur en début d'affichage (home), effacer le LCD, etc. Voici quelques instructions typiques :

Clear the LCD	$01
Move cursor home	$02
Move cursor left	$10
Move cursor right	$14

Il y en a d'autres. Examinez les listages des programmes donnés ici (en sur Internet) et lisez la documentation technique du HD44780 de Hitachi.
L'écriture de données vers le LCD a été sensiblement simplifiée grâce à l'instruction **LCDOUT**.
Cette instruction possède 2 caractéristiques intéressantes : il est possible d'envoyer un octet de commande à l'affichage, effacer le LCD par exemple, avant l'écriture et les données en sortie sont structurée à l'image de l'instruction **SEROUT**. Cela signifie qu'il est possible d'utiliser les modificateurs (*modifier*) SEROUT typiques tels que BIN, HEC, DEC, STR et REP.
La lecture de l'information du LCD est tout aussi facile avec l'instruction **LCDIN** dont la syntaxe est identique à **LCDOUT**. Avec **LCDIN** l'octet de position (location byte) doit pointer à l'emplacement de mémoire du début de lecture. Les constantes suivantes sont utiles dans le cadre de programmes qui utilisent les instructions ayant trait au LCD :

```
DDRam   CON   $80
CGRam   CON   $40
```

DDRam est la mémoire dans laquelle se trouvent les caractères à afficher.
CGRam est la zone de mémoire de 64 octets dans laquelle sont stockés les formats des caractères propres à l'utilisateur (custom). Si votre programme n'utilise pas cette mémoire pour des caractères spécifiques, vous pourrez, avec les instructions LCDOUT et LCDIN, l'utiliser comme de la RAM externe (off-board). Le modificateur STR pourra être utilisé pour le transfert d'un bloc d'octets vers le LCD ou en provenance de l'affichage.
Le **listage 1** démontre la simplicité de la sortie vers le LCD avec le BS2p.

Support du I²C de Philips

Le bus I²C de Philips est un bus bidirectionnel bifilaire très populaire. Il existe de nombreux composants I²C qui dès à présent peuvent être utilisés très facilement avec le BASIC Stamp. Les 2 lignes de bus sont SDA (*Serial DAta*) et SCL (*Serial CLock*). Les lignes SDA et SCL doivent être forcées à Vdd (+5 V) par le biais de résistances de 4kΩ7.
Le BS2p supporte les composants I²C par le biais des instructions :

I2COUT *pin, slave_addr, addr{\low_addr}, [output data]*
I2CIN *pin, slave_addr, addr{\low_addr}, [input data]*

Les seules broches auxquelles les composants I²C puissent être connectés sont les groupes de broches 0 & 1 ou 8 & 9.

Connexions LCD :

LCD	Option 1	Option 2
LCD.E	BS2p.0 ou BS2p.1	BS2p.8 ou BS2p.9
LCD.R/W	BS2p.2	BS2p.10
LCD.RS	BS2p.3	BS2p.11
LCD.DB4	BS2p.4	BS2p.12
LCD.DB5	BS2p.5	BS2p.13
LCD.DB6	BS2p.6	BS2p.14
LCD.DB7	BS2p.7	BS2p.15

Bus I²C	Option 1	Option 2
SDA	BSP.0	BSP.8
SCL	BSP.1	BSP.9

Le paramètre broche (pin) de chaque instruction désigne la broche SDA. L'adresse esclave (slave addr) désigne le composant I²C auquel se connecter. Addr est l'emplacement de mémoire au coeur du composant I²C où doit se faire la lecture ou l'écriture. L'utilisation du caractère « \ » (*backslash*) permet un adressage sur 2 octets pour les composants qui offriraient une telle possibilité. Le paramètre placé après le backslah est l'octet de poids faible de l'adresse.

En guise d'illustration des routines I²C, le programme donné dans le **listage 2** génère un nombre pseudo-aléatoire, l'écrit dans une RAM I²C, attend 100 ms avant de relire la donnée écrite. On utilise un LCD pour

Listing I. LCD output with the BS2p.

```
' ---[ Title ]-------------------------------
'
' File..... LCDDEMO.BSP
' Purpose... BASIC Stamp 2p -> LCD
' Author.... Jon Williams
' E-mail.... jonwms@aol.com

' {$STAMP BS2p}

' ---[ Program Description ]-------------------------
'
' This program initializes the LCD in 4-bit mode and sends a simple message
'
' Connections:
'   - LCD.E    -> Pin0 (pulled down [to ground] through 4.7K)
'   - LCD.R/W  -> Pin2 (or grounded for write-only operation)
'   - LCD.RS   -> Pin3
'   - LCD.D4   -> Pin4
'   - LCD.D5   -> Pin5
'   - LCD.D6   -> Pin6
'   - LCD.D7   -> Pin7

' ---[ I/O Definitions ]---------------------------
'
LCDpin          CON     0                               ' data on pins 4 - 7

' ---[ Constants ]--------------------------------
'
' LCD control characters
'
NoCmd           CON     $00                             ' just print
ClrLCD          CON     $01                             ' clear the LCD
CrsrHm          CON     $02                             ' cursor home
CrsrLf          CON     $10                             ' cursor left
CrsrRt          CON     $14                             ' move cursor right
DispLf          CON     $18                             ' shift display left
DispRt          CON     $1C                             ' shift displayright
DDRam           CON     $80                             ' Display Data RAM control
Line1           CON     $80                             ' address of line 1
Line2           CON     $C0                             ' address of line 2

' ---[ Initialization ]----------------------------
'
LCD_Setup:
  LCDCMD LCDpin,%00110000 : PAUSE 5          ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000 : PAUSE 0          ' 4-bit mode
  LCDCMD LCDpin,%00101000 : PAUSE 0          ' 2-line mode
  LCDCMD LCDpin,%00001100 : PAUSE 0          ' no crsr, no blink
  LCDCMD LCDpin,%00000110                          ' inc crsr, no disp shift

' ---[ Main Code ]-------------------------------
'
Main:
  LCDOUT LCDpin,ClrLCD,[" Parallax BS2p"]    ' splash screen
  LCDOUT LCDpin,Line2,["  LCD Control"]
  END
```

Listing 2. Random number generator, I2C demonstration.

```
' ---[ Title ]-------------------------------
'
' File..... PCF8570.BSP
' Purpose... Demonstrates I2CIN and I2COUT
' Author.... Jon Williams
' E-mail.... jonwms@aol.com

' {$STAMP BS2p}

' ---[ Program Description ]-------------------------
'
' Writes to and reads from I2C RAM. Data is displayed on line 2 of a 2x16
' LCD.

' Program requires 2x16 LCD
'   - LCD.E    -> Pin0 (pulled down [to ground] through 4.7K)
'   - LCD.R/W  -> Pin2 (or grounded for write-only operation)
'   - LCD.RS   -> Pin3
'   - LCD.D4   -> Pin4
'   - LCD.D5   -> Pin5
'   - LCD.D6   -> Pin6
'   - LCD.D7   -> Pin7

' ---[ Constants ]-----------------------------
'
LCDpin          CON     0                       ' LCD is connected to OutL
I2Cpin          CON     8                       ' SDA on 8; SCL on 9

NoCmd           CON     0
ClrLCD          CON     $01                     ' clear the LCD
CrsrHm          CON     $02                     ' move cursor to home position
CrsrLf          CON     $10                     ' move cursor left
CrsrRt          CON     $14                     ' move cursor right
DispLf          CON     $18                     ' shift displayed chars left
DispRt          CON     $1C                     ' shift displayed chars right
DDRam           CON     $80                     ' Display Data RAM control
CGRam           CON     $40                     ' Custom character RAM control
Line1           CON     $80
Line2           CON     $C0

addr            VAR     Byte                    ' address to write to / read from
rVar            VAR     Word                    ' for random number
tOut            VAR     Byte                    ' test value to write to LCD
tIn             VAR     Byte                    ' test value to read from LCD
temp            VAR     Word                    ' temp value for numeric display
width           VAR     Nib                     ' width of rt justified display
pos             VAR     Byte                    ' column position for display
digits  VAR     Nib                             ' number of digits to display

' ---[ EEPROM Data ]-----------------------------
'
Super2          DATA    %01100                  ' super-script 2
                DATA    %00010
                DATA    %00100
                DATA    %01000
                DATA    %01110
                DATA    %00000
                DATA    %00000
                DATA    %00000

' ---[ Initialization ]--------------------------
'
LCD_Setup:
  PAUSE 500
  LCDCMD LCDpin,%00110000 : PAUSE 5          ' 8-bit mode
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00110000 : PAUSE 0
  LCDCMD LCDpin,%00100000                    ' 4-bit mode
  LCDCMD LCDpin,%00101000                    ' 2-line mode
  LCDCMD LCDpin,%00001100                    ' no crsr, no blink
```

```
  LCDCMD LCDpin,%00000110                          ' inc crsr, no disp shift

  ' download custom character map to LCD

  LCDCMD LCDpin,CGRam                        ' write to CGRAM

  FOR addr = Super2 TO (Super2 + 7)          ' build custom char
    READ addr,temp                                 ' get byte from EEPROM
    LCDOUT LCDpin,NoCmd,[temp]                     ' put into LCD CG RAM
  NEXT


' ---[ Main Code ]-------------------------------
'
Main:
  LCDOUT LCDpin,ClrLCD,["  BSP <-> I",0,"C"]
  LCDOUT LCDpin,Line2, [" Communications"]
  PAUSE 2000
  LCDCMD LCDpin,ClrLCD

  LCDOUT LCDpin,ClrLCD,["I",0,"2:     Out="]
  LCDOUT LCDpin,Line2 + 10,["In="]

  FOR addr = 0 TO 255                        ' test all addresses
    RANDOM rVar                                    ' create "random" value
    tOut = rVar.HighByte

    I2COUT I2Cpin,$A0,addr,[tOut]                  ' write to I2C RAM
    PAUSE 100
    I2CIN I2Cpin,$A0,addr,[tIn]                    ' read it back

    ' display results

    LCDOUT LCDpin,Line1 + 4,[DEC addr]
    temp = tOut : width = 3 : pos = Line1 + 13
    GOSUB RJ_Print
    temp = tIn : width = 3 : pos = Line2 + 13
    GOSUB RJ_Print
    PAUSE 350
  NEXT

  PAUSE 1000
  GOTO Main

  END

' ---[ Subroutines ]-----------------------------
'
RJ_Print:                                          ' right justify
  digits = width
  LOOKDOWN temp,<[0,10,100,1000,65535],digits
  LCDOUT LCDpin,pos,[REP " "\(width-digits),DEC temp]
  RETURN
```

visualiser l'adresse et les données de sortie et d'entrée.

Support du Dallas 1-Wire™

Le bus Dallas 1-Wire™ est un système comportant un unique maître de bus et un esclave (ou plus). Dans ce cas-là, Le BS2p fait office de maître de bus. Chaque composant du bus 1-Wire™ possède un numéro de série (utilisé pour l'adressage) unique dont il est doté indélébilement lors de la fabrication.
Le support des composants 1-Wire™ se fait à l'aide de 2 instructions à la mise en oeuvre aisée :

OWOUT *pin, reset*, [*output data*]
OWIN *pin, reset*, [*input data*]

La communication avec des composants 1-Wire™ peut se faire par le biais de toute broche disponible. Cette broche devra être forcée à Vdd (+5 V) par le biais d'une résistance de 4kΩ7. Le paramètre Reset possède plusieurs options, tels que, par exemple :

0 Pas de reset, mode octet, vitesse faible
1 Reset avant la donnée, mode octet, vitesse faible
2 Reset après la donnée, mode octet, vitesse faible
3 Reset avant et après la donnée, mode octet, vitesse faible
4 Pas de reset, mode bit, vitesse faible
5 Reset avant la donnée, mode bit vitesse faible
8 Pas de reset, mode octet, vitesse élevée
9 Reset avant la donnée, mode octet, vitesse élevée

Lors de l'écriture du code il pourrait

être plus facile de définir les CONstantes de la manière suivante :

```
OW_FERst      CON  %0001
   ' Reset amont (front-end)
OW_BERst      CON  %0010
   ' Reset aval (back-end)
OW_BitMode    CON  %0100
OW_HighSpd      CON  %1000
```

Ces valeurs peuvent être additionnées pour obtenir le paramètre de reset correct.
Lors de la réception de données en mode bit, toutes les variables de l'argument de *données d'entrée* (*input data*) ne recevront qu'un bit quel que soit le type de variable.
À l'image de ses prédécesseurs, le BS2p supporte la définition de variables de la taille du bit (*bit-sized variable*). Il est sage d'utiliser cette possibilité pour économiser de l'espace variable précieux lorsque l'on a affaire à la réception de données en mode bit.
Étant donné que les composants 1-Wire™ possède chacun leur propre numéro de série unique, la première opération que devra faire un programmeur est d'aller chercher ce numéro dans le composant. On pourra utiliser le petit programme du **listage 3** à cet effet.
Une fois que les numéros de série des composants sont connus, on pourra les associer au même bus. Le programme donné dans le **listage 4** procède à la lecture de 2 capteurs de température du type DS1820 connectés à la même broche .

Interrogation de broche (interruptions progicielles)

Bien que les utilisateurs du Stamp aimeraient disposer de cette possibilité, le BS2p n'a pas la capacité de traiter de vraies interruptions –fonction très délicate pour un interpréteur quel qu'il soit, et la compacité de l'interpréteur PBASIC ne fait que compliquer les choses. Le BS2p est cependant en mesure d'interroger des broches entre deux instructions PBASIC et de prendre une action spécifique en fonction du résultat de cette opération. Après paramétrage et validation le BS2p procèdera aux opérations suivantes dans le cas d'une interruption d'interrogation :
– Ne rien faire
– Mettre une broche de sortie à un état donné
– Exécuter un autre programme
– Attendre (mettre le programme en pause) jusqu'à l'apparition de la condition d'interruption
– Toute combinaison de possibilités 2, 3 et 4.

Permettez-moi de revenir sur ce point pour que les choses soient parfaitement claires. Une fois paramétré et validé, le BS2p vérifiera, entre 2 instructions PBASIC, l'état des broches d'entrée à interroger. En cas de constatation de la condition d'entrée spécifiée, l'état d'« interruption » est mis à l'état vrai (true) ce qui se traduira par la prise de l'action spécifiée (qui pourra, le cas échéant, en comporter plusieurs).
On utilisera, pour la définition des broches d'entrée à interroger, l'instruction POLLIN dans la syntaxe est la suivante :

POLLIN pin,state

Le paramètre de broche (pin) sera toujours compris entre 0 et 15, l'état (state) pouvant être lui soit 0 (bas) ou 1 (haut). Comme nous le disions, lors de l'activation, les broches spécifiées seront interrogées entre 2 instructions PBASIC.
Lorsque l'état d'interruption est vrai, on pourra commander les broches de sortie à interroger. Après paramétrage, la prise de contrôle sur ces broches est automatique et se fait après une condition d'interruption.
On utilisera l'instruction **POLLOUT** pour définir les broches de sortie :

POLLOUT pin,state

Le paramétrage est identique à celui de POLLIN, à ceci près que le BS2p commande une sortie. La broche spécifié sera forcée à l'état prévu lorsque la condition d'interruption sera devenue vraie.
Une fois les entrées et sortie à interroger définies, on utilisera l'instruction **POLLMODE** pour les valider.

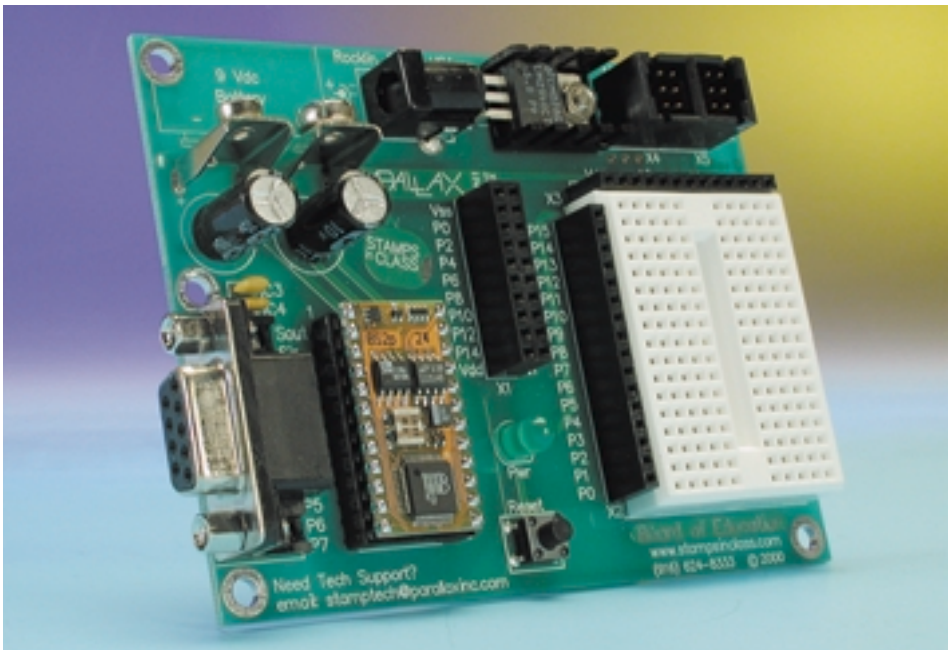
Mode POLLMODE

Le paramètre de mode pourra aller de 0 à 15 en respect des définitions données ci-après :
– Désactiver l'interrogation et effacer la définition des entrées et sorties à interroger
– Désactiver l'interrogation et sauvegarder la définition des entrées et sorties à interroger
– Activer l'interrogation sur les seules sorties à interroger
– Activer l'interrogation sur les sorties à interroger et le processus d'exécution de l'interrogation
– Effacer la configuration des entrées à interroger
– Effacer la configuration des sorties à interroger
– Effacer la configuration et des entrées et des sorties à interroger

Les modes 8 à 15 sont identiques aux modes 0 à 7 à ceci près que la condition d'interruption est verrouillée (*latched*).
Le petit morceau de code donné dans le **listage 5** démontre la fonction d'interrogation d'entrées et de sorties.
Lors de l'exécution de ce code on a ouverture d'un écran **DEBUG** dans lequel se déroule tranquillement le message « *Waiting...* ». Actionnez maintenant la touche reliée à la broche 4 et maintenez-là enfoncée. La LED connectée à la broche 0 va s'allumer. Si vous relâchez la touche vous verrez la LED d'éteindre.
Avec cette possibilité de commander des sorties par interrogation d'entrées, vous pouvez réaliser une « porte logique en arrière-plan » qui travaille pendant l'exécution du programme par le Stamp. Ajoutez la ligne suivante au code du listage 5 et vous verrez que la broche 4 ou la broche 5 produira l'allumage de la LED –une porte OU en arrière-plan :

```
POLLIN 5,0      ' interruption si
   broche 5 est au niveau bas
```

La nature du BS2p permet la commande de broches de sortie interrogeables par le biais de broches de sortie standard. Le **listage 6** donne un exemple de cette fonction.
En l'absence de définition d'interrogation, tout ce qui pourrait se passer est un clignotement de la LED en broche 0 (en raison de l'instruction **TOGGLE**). Après activation de l'interrogation la LED reliée à la broche 1 se mettra à clignoter à la même fréquence que la LED connectée à la broche 0 (dans le cas présent dans l'état inverse).
Il ne faudra pas oublier que l'état d'interruption est une condition globale et que, si aucune des broches d'entrée à interroger ne respecte les exigences l'état d'interruption sera effacé. Il peut arriver que l'on veuille suivre un nombre de broches et prendre l'une ou l'autre action en fonction des entrées. On



pourra le faire par un verrouillage de la condition d'interruption et par identification de (ou des) broches ayant déclenché l'état d'interruption.
Le programme suivant, celui du **listage 7**, est un programme d'alarme rudimentaire qui verrouille la condition d'interruption. Les broches 4 et 5 sont paramétrées pour déclencher une interruption lorsqu'elles sont forcées au niveau bas. Lorsque la condition d'interruption est vraie la LED Alarm prise à la broche 0 s'allume. On utiliser l'instruction **POLLMODE 10** (commande des sorties et verrouillage de la condition d'interruption) en vue de déterminer quelle(s) broche(s) a déclenché la condition d'interruption.
Au cours de la boucle, on a lecture de l'emplacement de mémoire 128 du bloc-note pour identifier toute broche d'interruption active. Les interruptions ayant été verrouillées, la variable *iPins* donnera les broches ayant été actives au cours du cycle d'interrogation. Dès que l'on dispose de cette information il est facile de réaliser un affichage simple. Après avoir procédé à l'action impliquée par l'interruption, on pourra réinitialiser (reset) le total par une nouvelle instruction **POLLMODE**.
Dans les exemples donnés jusqu'à présent, le programme du BS2p a passé du temps dans une boucle dans l'attente d'une interruption à se faire. L'instruction **POLLWAIT** permet au BS2p de suspendre l'exécution du programme jusqu'à ce que la condition d'interruption devienne vraie. Lorsque ce basculement de faux vers vrai à eu lieu, le programme se poursuit à la ligne suivante.

POLLWAIT *period*

L'instruction **POLLWAIT** est très proche de l'instruction **NAP** qui fait passer le BS2p dans un mode faible consommation (low power). En fonction de la durée définie, le BS2p quittera sont mode faible consommation et procèdera à l'interrogation des broches d'entrée à interroger définies. Si l'une des conditions est remplie, le programme se poursuit, sinon le BS2p retourne dans l'état faible consommation. Les valeurs de durée possibles sont les suivantes :

– 18 ms
– 36 ms
– 72 ms
– 144 ms
– 288 ms
– 576 ms
– 1 152 ms (1,152 seconde)
– 2 304 ms (2,304 secondes)
– Attente hors mode faible consommation

Tout comme dans le cas d'une instruction **NAP**, toute sortie fera un écart lorsque le BS2p quitte son mode faible consommation (durée 0 à 7 ci-dessus) pour procéder à sa vérification. Essayez donc le code suivant :

```
HIGH 3  ' allumer la LED connectée à la broche 3
POLLIN 5,0      ' interroger la broche 5 pour un niveau bas
POLLOUT 2,1     ' allumer la LED de la broche 2 en cours d'interruption
POLLMODE 2      ' activer la commande d'interrogation de sortie
Loop:
  POLLWAIT 0    ' passer en mode faible conso pour 18 ms
  TOGGLE 1      ' commuter la LED de la broche 1
  GOTO Loop
```

Lors du lancement du programme, la LED connectée à la broche 3 s'allumera. Examinez-là de près. Voyez-vous la pulsation apparente de cette LED ? Ceci est dû à la sortie du mode faible consommation du BS2p, instant auquel toutes les broches se trouvent, un court instant, paramétrées en entrées. Vous pourrez constater un changement par modification de la durée de **POLLWAIT**. Vous constaterez qu'une augmentation de la durée se traduit par un « glissement » moins fréquent, la fréquence d'interrogation devenant elle aussi plus faible. Une action maintenue sur la touche activera l'interruption lors de chaque interrogation de sorte que la LED reliée à la broche 0 va changer d'état à une fréquence déterminée par la durée de **POLLWAIT**.
Pour les applications où l'on n'a que faire du mode faible consommation mais où l'on a besoin de **POLLWAIT**, on optera pour une durée de type 8 (attente hors-mode faible consommation). Dans ce cas-là le BS2p attend une condition d'interruption mais ne passe pas en mode « low power .»
Nous pouvons, pour finir, utiliser POLLRUN pour forcer le BS2p à sauter à un autre créneau (*slot*) de programme lorsque la condition d'interruption devient vraie.

POLLRUN *program*

Le créneau de programme par défaut pour **POLLRUN** est 0. En cas de choix d'un **POLLMODE** 3 ou 4, et en l'absence de définition de programme **POLLRUN**, le BS2p sautera au programme 0 à l'apparition de la condition d'interruption. En cas, parallèlement, de définition et de validation de sorties à interroger (**POLLMODE** 4), les sorties seront positionnées avant exécution du nouveau programme. Les modes **POLLRUN** 3 et 4 ont un comportement « non redéclenchable » (*one-shot*) pour éviter que le BS2p ne

donne l'impression de se retrouver verrouillé en raison d'un saut incessant vers le programme spécifié.

Tamponnage sériel vers la RAM- bloc-note.

Il est des circonstances où une application requiert la réception et le stockage de toute une série de données. Dans nombre de ces applications, seule une petite partie de cette chaîne est nécessaire. Si le modificateur WAIT peut être utile dans ce cas-là, il n'est pas toujours très fiable à des taux de transfert très élevés.
Le BS2p permet au programmeur de rediriger toute entrée sérielle (**SERIN**, **I2CIN**, **OWIN**, **LCDIN**) vers la zone de RAM- bloc-note (*scratchpad RAM*). Une fois que les données y sont, on pourra utiliser l'instruction **GET** pour examiner la chaîne d'entrée pour y trouver les données requises. Cette fonction est extrêmement facile dès lors que l'on connaît la position des données. Cette technique permet au BS2p de stocker jusqu'à 127 octets de données sérielles. La syntaxe ressemble alors à ceci :

SERIN *pin,baud,[SPSTR L]*

Le nouveau modificateur SPSTR (*scratchpad string*), est similaire au modificateur STR à ceci près qu'il redirige les donnés vers la RAM-bloc-note, en commençant à l'emplacement 0. Le paramètre L(ongueur) (*Length*) indique au Stamp nombre d'octets à tamponner (stocker).

Plus de broches

Les utilisateurs du Stamp demandent plus de broches depuis longtemps; ils ont été entendus. Le BS2p existe également dans une version à 40 broches qui offre pas moins de 16 broches d'E/S sortie additionnelles à l'utilisateur du Stamp. L'utilisation des broches supplémentaires rappelle quelque peu la mise en oeuvre des créneaux de programme : il faut indiquer spécifiquement au Stamp d'utiliser les broches additionnelles. Toutes les instructions qui requièrent un numéro de broche ne changent pas dans le sens où elles s'attendent à recevoir un numéro de broche allant de 0 à 15. Avec le BS2p-40, le programme indique quel set de broches d'E/S il faut utiliser.
Il existe 2 moyens pour choisir le second set de broches d'E/S :

```
AUXIO   ' sélecter les broches
  d'E/S auxiliaires
IOTERM 1        ' sélecter les
  broches d'E/S auxiliaires
```

Pour revenir aux broches d'E/S principales on pourra utiliser l'une des 2 instructions suivantes :

```
MAINIO'sélecter les broches
  d'E/S principales
IOTERM 0  'sélecter les broches
  d'E/S principales
```

Sachez, au cas où vous vous poseriez la question que les broches interrogées peuvent être définies dans chacun de ces groupes d'E/S et qu'elles sont toutes actives quel que soit le set de broches d'E/S qui soit utilisé à ce moment-là. Il vous est possible de déterminer quelle est celle des 32 broches ayant induit la condition d'interruption par un examen de la RAM-bloc-note (cf. ci-après).
Notez que **MAINIO**, **AUXIO** et **IOTERM** ne sont pas disponibles sur le BS2p-24 pour la simple et bonne raison qu'il ne dispose que d'un set de 16 broches d'E/S.

Utilisation efficace de l'espace EEPROM

L'un des domaines les plus populaires de l'utilisation des microcontrôleurs des types BS2sx et BS2e est l'acquisition de données (data logging). Certains programmeurs ont utilisé des créneaux de programmes supplémentaires en tant que stockage non volatile avec les instructions **WRITE** et **READ** du Stamp. La difficulté cependant s'avère la nécessité de transférer des données d'un créneau de programme à un autre en utilisant le bloc-note comme mémoire intermédiaire. Ce problème n'existe plus avec le BS2p.
Le BS2p dispose d'une nouvelle instruction appelée **STORE**. Sa syntaxe est simple :

STORE *location*

Location (emplacement) est le créneau de programme (0 à 7) à utiliser comme cible pour les instructions READ et WRITE. Le nombre attribué à emplacement par défaut est le même que celui du programme en cours, sachant qu'il est passé au créneau du nouveau programme dès l'envoi d'une instruction RUN. L'intérêt de STORE est qu'à partir de maintenant le programmeur peut mettre à contribution les créneaux de programmes non utilisé et, avec quelques lignes de code, les traiter comme un seul grand bloc d'EEPROM.

Emplacements de RAM RO

Comme nous le disions plus haut, le BS2p possède 2 fois plus de RAM- bloc-note (127 octets) que le BS2sx. 5 octets en fin du bloc-note sont des emplacements à lecture seule (d'où le RO de *Read Only*); ils fournissent une information utile.

Programme en cours (quartet de poids faible (*low nibble*)) et emplacement **STORE** (quartet de poids fort (*high nibble*))
Détection de broche d'interruption, broches principales 0 à 7
Détection de broche d'interruption, broches principales 8 à 15
Détection de broche d'interruption, broches auxiliaires 0 à 7
Détection de broche d'interruption, broches auxiliaires 8 à 15

Les emplacements 128 à 131 pourront être utilisés par votre programme pour déterminer quelle(s) broche(s) est (sont) la cause d'une interruption. Une broche d'interruption active est identifiée par un « 1 » quel que soit son état d'entrée (actif au niveau bas ou au niveau haut). Les bits présents à ces emplacements ne sont actifs que lorsque l'interruption est active, de sorte qu'il se peut que vous ayez à verrouiller la condition d'interruption pour déterminer la cause après effet.

Plus qu'un nouveau Stamp

Le nouveau BS2p est loin d'être purement et simplement un nouveau BASIC Stamp, il s'agit d'une amélioration significative dans la lignée des mini-microcontrôleurs de Parallax –sans doute la plus importante depuis l'introduction du BS2. Associant une puissance de traitement supérieure à celle de son prédécesseur et ce à une consommation de courant moindre, il est, espérons-le, un signe précurseur d'autres évolutions intéressantes chez Parallax.

(014132)

TDA8444

Convertisseur A/N - N/A (CNA)

ELEKTOR INFOCARTE 9/2001

Valeurs caractéristiques (V_{CC} = 12 V; T_{amb} = 25 °C, sauf mention contraire)

Symbole	Paramètre	Conditions	Min.	Typ.	Max.	Unité
Alimentation						
V_{CC}	Tension d'alimentation		4,5	12	13,2	V
I_{CC}	Consommation de courant donnée	V_{MAX} = V_{CC} = 12 V, data = 00H	12	14	19	mA
P	Puissance dissipée			170	250	mW
V_{rst}	Tension d'initialisation (Power-Reset)		1		4	V
Broche V_{MAX}						
$V_{i(VMAX)}$	Tension d'entrée efficace		1		V_{CC}–2,0	V
I_i	Courant d'entrée	V_{MAX} = V_{CC} ou V_{MAX} = 1 V			10	mA
SDA et SCL du bus I2C						
V_I	Tension d'entrée		0		5,5	V
V_{IL}	Tension d'entrée LOW				1,0	V
V_{IH}	Tension d'entrée HIGH		3,0			V
I_{IL}	Courant d'entrée LOW	V_{SDA} = V_{SCL} = –0,3 V			–10	µA
I_{IH}	Courant d'entrée HIGH	V_{SDA} = V_{SCL} = 6 V			±10	µA
V_{OL}	Tension de sortie SDA LOW	I_L = 3 mA			0,4	V
$I_{O(sink)}$	Consommation de courant de sortie SDA LOW		3	8		mA
Bits d'adresse A0 à A2						
V_I	Tension d'entrée		0		V_{CC}	V
V_{IL}	Tension d'entrée LOW				1,0	V
V_{IH}	Tension d'entrée HIGH		2,2			V

TDA8444

Convertisseur A/N - N/A (CNA)

ELEKTOR INFOCARTE 9/2001

TDA8444
Octuple CNA 6 bits à bus I2C

Fabricant

Philips Semiconductors

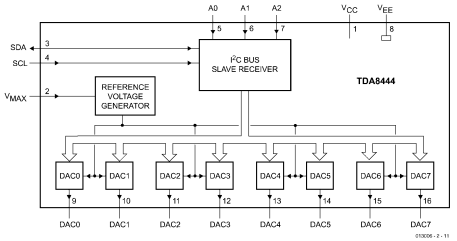
http:/www.semiconductors.com/acrobat/datasheets/TDA8444 3.pdf

Caractéristiques techniques :

– Octuple convertisseur numérique/analogique à résolution de 6 bits
– Plage de tension de sortie ajustable en commun
– Sorties en push-pull
– Sorties protégées contre les courts-circuits
– 3 bits programmables par adresse-esclave
– Large plage de tension d'alimentation
– Coefficient de température faible

Description

Le TDA8444 intègre 8 convertisseurs numérique/analogique ayant une résolution de 6 bits, possède des sorties programmables et une interface de bus I²C (récepteur-esclave) dotée de 3 bits d'adresse paramétrables (la version SO16 n'en possède que 2) ainsi qu'une entrée (W_{MAX}) qui permet de définir la tension de sortie maximale. Il est possible, par le biais d'un mot de 6 bits, de paramétrer séparément chacun des CNA à l'une des 64 valeurs possible sachant cependant que V_{MAX} définit la tension de sortie pour la totalité des CNA. La résolution atteint de ce fait de l'ordre de 1/64ème de V_{MAX}. Après mise sous tension tous les CNA sont paramétrés à leur valeur la plus faible.



Structure interne

Application

Interface I²C pour servos, Elektor n°279, septembre 2001

Types de boîtier et brochages

TDA8444 : DIP16
TDA8444T : SO16
TDA8444AT : SO20

Symbole	DIP16	SO16	SO20	Description
V_{CC}	1	1	1	Tension d'alimentation
V_{MAX}	2	2	2	Entrée de commande
SDA	3	3	3	Données sérielles (E/S) du bus I2C
SCL	4	4	4	Horloge du bus I2C
A0	5	6	7	Bit d'adresse 0 programmable du bus I2C (récepteur-esclave)
A1	6	7	8	Bit d'adresse 1 programmable du bus I2C (récepteur-esclave)
A2	7	-	9	Bit d'adresse 2 programmable du bus I2C (récepteur-esclave)
V_{EE}	8	8	10	Masse
DAC0	9	9	11	Sortie de tension analogique 0
DAC1	10	10	13	Sortie de tension analogique 1
DAC2	11	11	14	Sortie de tension analogique 2
DAC3	12	12	15	Sortie de tension analogique 3
DAC4	13	13	16	Sortie de tension analogique 4
DAC5	14	14	17	Sortie de tension analogique 5
DAC6	15	15	18	Sortie de tension analogique 6
DAC7	16	16	20	Sortie de tension analogique 7
n.c.	-	5	5,6, 12,19	Non connecté

Valeurs-limites impératives :

Symbole	Paramètre	Min.	Max.	Unité
V_{CC}	Tension d'alimentation	–0,5	+18	V
I_{CC}	Courant consommé	–10	+40	mA
$P_{(MAX)}$	Puissance dissipée	–	500	mW
$V_{i(n)}$	Tension d'entrée SDA et SCL V_{MAX}, A0 à A2 et DAC0 à DAC7	–0,5 –0,5 –0,5	+5,9 +5,9 VCC+0,5	V
$I_{(n)}$	Courant pour toutes les broches exception faite de V_{CC} et V_{EE}	–	±10	mA
T_{amb}	Température de service	–20	+70	°C

Description du fonctionnement :

Interface du bus I²C

L'interface de bus I²C ne supporte que le mode esclave. Elle accepte des données au format :

S	0 1 0 0 A2 A1 A0 0	A	I3 I2 I1 I0 SD SC SB SA	A	X X D5 D4 D3 D2 D1 D0	A	P

où

S	Condition initiale
A2 à A0	Bits d'adresse programmables
A	Acquiescement (*Acknowledge*)
I3 à I0	Bits d'instruction
SD à SA	Bits de sous-adresse
X	Sans signification
D5 à D0	Bits de donnée
P	Condition d'arrêt

Adresses

Les adresses valides pour le TDA8444 et le TDA8444AT sont 40HEX, 42_{HEX}, 44_{HEX}, 46_{HEX}, 48_{HEX}, $4A_{HEX}$, $4C_{HEX}$ et $4E_{HEX}$, sachant que la plage des adresses du TDA8444T est elle limitée à 48_{HEX}, $4A_{HEX}$, $4C_{HEX}$ et $4E_{HEX}$ vu que la ligne A2 est forcée au niveau haut. Il est possible d'adresser jusqu'à 8 TDA8444 sur un même bus I²C.

Instructions

Les instructions valides sont : 00_{HEX} à $0F_{HEX}$ et $F0_{HEX}$ à FF_{HEX}. Le TDA8444 ne réagit pas à d'autres combinaisons des 4 bits d'instruction I3 à I0 différents de 0 ou F, mais donne cependant à chaque fois un signal d'acquiescement (*Aknowledge*). La différence entre les instructions 0 et F ne prend d'importance que dans le cas d'une émission, au cours d'un même transfert, de plus d'un octet de donnée. L'instruction 0 a pour effet d'écrire, dans le verrou du CNA, les octets de donnée à plusieurs adresses successives qui débutent par celle définie par l'octet d'instruction (auto-incrémentation des sous-adresses).
L'instruction F a elle pour effet d'écrire les octets de donnée successivement dans le même verrou du CNA, celui dont la sous-adresse a été définie par l'octet d'instruction. S'il n'y a qu'un unique octet de donnée le verrou du CNA reçoit les données à la même sous-adresse que celle définie par l'octet d'instruction.

Sous-adresses

La plage des adresses valides va de 0_{HEX} à 7_{HEX}. Les sous-adresses correspondent aux sorties DAC0 à DAC7 du CNA. La fonction d'auto-incrémentation (AI) de l'instruction 0 fonctionne avec la totalité des sous-adresses 0 à F possibles. La sous-adresse qui suit la sous-adresse F est, cycliquement, la sous-adresse 0. Le transfert des données se fait sur le flanc montant du signal d'horloge commandé par le signal d'acquiescement.
Les lignes SCL et SDA respectent les spécifications I²C classiques. Il faudra protéger les broches contre des impulsions de tension positive par la mise en place de diodes zener vers la masse qui limiteront à 5,5 V la tension véhiculée par le bus. Les entrées d'adresse A0 à A2 peuvent être forcées à la masse (A = 0) ou à V_{CC} (A = 1). Les entrées restées en l'air sont considérées comme étant à « 1 ».

V_{MAX}

L'entrée V_{MAX} offre la possibilité de limiter la plage des tensions de sortie. À pleine résolution, la tension de sortie maximale d'un CNA atteint V_{MAX} + $V_{DAC(min)}$. Ceci permet d'obtenir une précision élevée même dans le cas de niveaux de tension de sortie faibles.

CNA

Les CNA intègrent un verrou (*latch*) à 6 bits, des commutateurs de courant et un amplificateur opérationnel. Les sources de courant pilotées par les commutateurs ont un poids allant de 2^0 à 2^5. L'amplificateur opérationnel convertit la somme des courants commutés en une tension allant de quelque 0,5 à 10,5 V, si tant est que V_{MAX} ait été paramétrée à V_{CC} = 12 V. Les sorties des CNA sont protégées contre les courts-circuits jusqu'aux potentiels de la tension d'alimentation. Il faudra faire en sorte que la charge capacitive qu'elles ont à subir ne dépasse pas 2 nF ceci en vue d'éviter des oscillations. Même dans le cas le plus défavorable le coefficient de température de chaque sortie est meilleur que 0,1 LSB/K.

TDA8444

Convertisseur A/N - N/A (CNA)

ELEKTOR INFOCARTE 9/2001

Valeurs caractéristiques (V_{CC} = 12 V, T_{amb} = 25 °C, sauf mention contraire)

Symbole	Paramètre	Conditions	Min.	Typ.	Max.	Unité
I_{IL}	Courant d'entrée LOWLOW	$V_{An} = V_{EE}$	−10	−15		µA
I_{IH}	Courant d'entrée HIGH	$V_{An} = V_{CC}$			1	µA
DAC0 à DAC7						
V_O	Tension de sortie CNA	$V_{MAX} = V_{CC}$	0,1		V_{CC}−0,5	V
$V_{O(min)}$	Tension de sortie minimale donnée	data = 00H, I_L = −2 mA	0,1	0,28	0,5	V
$V_{O(max)}$	Tension de sortie maximale donnée	data = 3FH, I_L = −2 mA, $V_{MAX} = V_{CC}$	10,0	10,5	11,5	V
		data = 3FH, I_L = −2 mA, $I < V_{MAX} < 10$ V		(Anm.)		V
$I_{O(sink)}$	Courant de sortie en drain (sink) donnée	$V_{DAC} = V_{CC}$, data = 1FH	2	8	15	mA
$I_{O(source)}$	Courant de sortie en source (source)	$V_{DAC} = V_{EE}$; data = 1FH	−2		−6	mA
Z_O	Impédance de sortie donnée	data = 1FH, −2≤ I_L ≤+2 mA		4	50	Ω
DNL	Non-linéarité différentielle	$V_{MAX} = V_{CC}$, I_L = −2 mA			<±0,5	LSB
INL	Non-linéarité intégrale	$V_{MAX} = V_{CC}$, I_L = −2 mA			<±0,5	LSB
ΔG_{FS}	Erreur de gain CC à pleine modulation donnée	data 3FH, I_L = −2 mA			5	%
$\Delta G/\Delta data$	Erreur de gain CC en cas d'échange de données avec autre CNA donnée	data 3FH, I_L = −2 mA		<±0,5		LSB
TC	Coefficient de température donnée	data 3FH, I_L = −2 mA		<±1		LSB/K

Note: La tension de sortie typique est de $(V_{swing} / V_{CC}-2,0) \cdot V_{MAX} + V_{O(00H)}$ avec $V_{swing} = VO(3FH) - VO(00H)$ pour $V_{MAX} = V_{CC}$

Convertisseur S-VHS/Vidéo

De C/Y vers CVBS

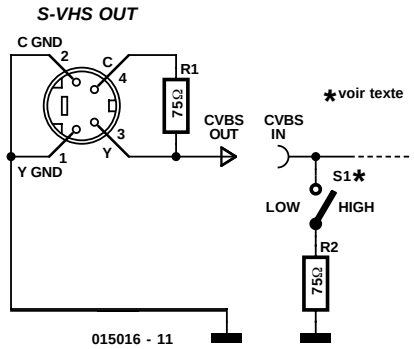
Wilfried Foede

Nombre de cartes vidéo, ordinateurs portables et lecteurs de DVD ne fournissent souvent, à l'intention du téléviseur chargé de visualiser l'information d'image, que les composantes Y (luminance) et C (chrominance) du signal S-VHS. Dans ces conditions, les téléviseurs et magnétoscopes ne disposant que d'une entrée vidéo standard (CVBS) ne sont pas, sans autre forme de procès, utilisables.

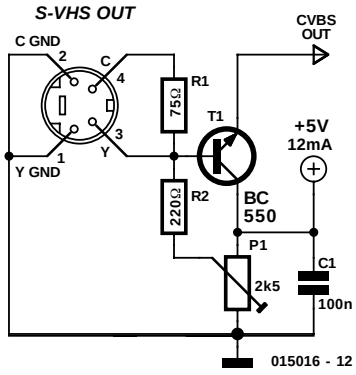
Les téléviseurs et magnétoscopes s'attendent, normalement, à trouver, sur l'entrée vidéo (broche 20 de la prise Péritel ou sur l'embase Cinch correspondante), un signal CVBS (*Colour*, *Video*, *Blanking* = suppression de faisceau et *Synchronisation*). Si la source de signal vidéo ne dispose que d'une sortie vidéo de type S-VHS on se trouve confronté à un problème de compatibilité. En effet, S-VHS possède 2 lignes de signal, à savoir Y et C, alors que CVBS n'en comporte qu'une. Le petit tableau ci-dessous fournit les informations concernant les signaux vidéo S-VHS et CVBS.

+Y = 1 V_{CC} sur 75 Ω, avec répartition de 0,7 V_{CC} pour la vidéo et 0,3 V_{CC} pour Sync
C = Burst de 0,3 V_{CC}, signal de chrominance correspondant au contenu de l'image
+CVBS = 1 V_{cc} sur 75 Ω, C et Y étant additionnés.

Ce tableau permet de s'imaginer qu'il devrait être possible d'obtenir, à partir des 2 composantes du signal S-VHS, et pour peu que l'on réussisse la combinaison requise, un signal vidéo de type CVBS.
L'approche la plus simple et qui est, souvent, parfaitement viable prend la forme de l'électronique passive représentée en **figure 1**. Dans ce schéma, la résistance interne de l'un des composants constitue la résistance de charge de l'autre. Cependant, en cas de connexion à une entrée CVBS la résistance d'entrée de 75 Ω constitue une charge supplémentaire. Le signal est trop faible d'environ 30%. On pourra, pour compenser, diminuer à dessein le signal C à l'aide d'une résistance de découplage de 75 Ω, ce qui se traduit par un rehaussement du signal CVBS. Il devient possible alors, par action sur la commande de réglage de contraste couleur du téléviseur, de corriger la composante couleur du signal. Il est également possible d'envisager de faire en sorte de pouvoir mettre la résistance d'entrée de 75 O en et hors-circuit. Nombre d'entrées vidéo disposent d'ailleurs d'un tel commutateur souvent identifié par la mention High/Low. L'erreur d'adaptation ne joue qu'un rôle insignifiant pour les liaisons inférieures à 2 m.
L'électronique active de la **figure 2** comporte un transistor et requiert de ce fait une tension d'alimentation de 5 V/15 mA. Il faudra connecter ce montage directement à la sortie S-VHS et non pas à l'autre extrémité du câble (côté entrée vidéo CVBS).
C et Y sont montés en parallèle tout comme dans le circuit de la figure 1. L'ajustable P1 permet de jouer sur le point de fonctionnement du circuit. On ajustera sa position de manière à obtenir l'image de la meilleure qualité possible. Il faudra, à l'aide d'un multimètre numérique pris sur l'émetteur de T1, rechercher sur ce point la tension positive la plus faible donnant une image correcte. La raison de ce réglage est la commutation automatique dont disposent les cartes vidéo en cas de terminaison de la sortie vidéo TV sous 75 Ω. Si la tension positive est trop élevée, la carte ne passe pas à une sortie en C et Y, ni à la fréquence d'image d'un téléviseur, à savoir 50 Hz.
La résistance d'émetteur est constituée par la résistance d'entrée de l'entrée CVBS. Dans ces conditions le signal CVBS présente pratiquement une résistance de sortie de 0 Ω et le niveau requis de 1 V_{CC}. (015016)



1



2

Les mystères du DMX512, Elektor n° 275 , mai 2001, page 53 et suivantes

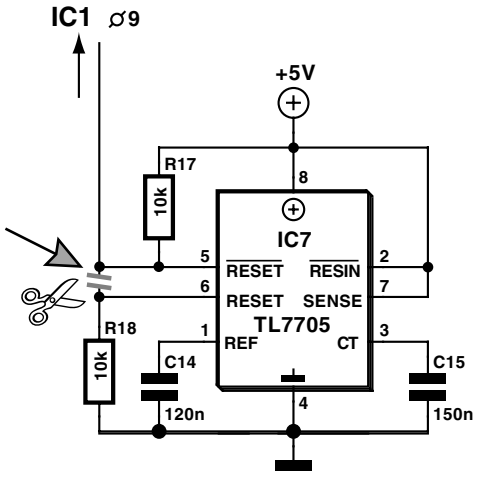
Il s' est malheureusement glissé une erreur dans le brochage des connecteurs XLR représenté en figure 5. La version simplifiée ne comporte que 3 broches de sorte que, dans les 2 dessins, la broche baptisée par erreur 5 est en fait la broche 2.

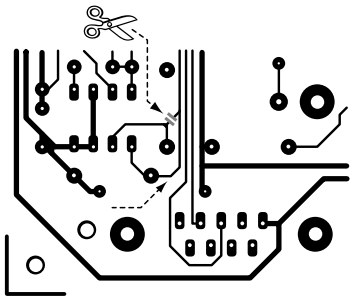
(010035)

Commutateur horaire, Elektor n° 276, juin 2001, page 14 et suivantes

Il y a eu interversion des broches 5 et 6 de IC7 tant au niveau du schéma qu'à celui de la platine. La broche 6 doit uniquement être reliée à la résistance R18. La broche 5 doit être reliée à la broche 9 (Reset\) de IC1. Les modifications à effectuer au niveau de la platine sont illustrées ci-contre.

(000184)





Préamplificateur RIAA à ECL 86, Elektor n° 270 , décembre 2000, page 70 et suivantes

Il y a malheureusement une erreur tant au niveau du schéma que de la liste des composants en ce qui concerne ce montage. Dans le schéma et la liste des composants R11 doit avoir une valeur de 1 MΩ et R12 de 33 kΩ.

(000016)

Ampli Hi-Fi P3, Elektor n°275, mai 2001, page 40 et suivantes

La légende de la figure 4 aurait dû dire : version équivalente transistorisée de l'étage de puissance PPP de la figure 3.

Le schéma de la figure 5 est celui d'une configuration à base de tubes EL34. Le type de tube associé au repère V3 et V4 devrait partant être EL34 et non pas KT88.

La diode située à proximité de C11 devrait s'appeler D5 et non pas D1. Le condensateur C6 devrait être doté d'un astérisque (à n'implanter que si nécessaire).

Dans la liste des composants la valeur de C10 devra être corrigée par 2 fois à 22 µF, la valeur mentionnée dans le schéma.

Les liaisons câblées requises pour la connexion du transformateur d'entrée mentionné dans le texte (taux de conversion de 1:1 ou de 1:2) n'apparaît ni sur le schéma ni sur la platine.

Au niveau de la sérigraphie de l'implantation des composants le rectangle représentant le transformateur est trop grand, recouvrant ainsi partiellement les picots de soudure B-2, A-2, B-1 et A-1. Il faudra, en fonction du rapport de conversion choisi pour Tr1, connecter les picots de soudure de la façon suivante :

Taux	Connexion câblée
1:1	B2-A2 et B1-A1
1:2	A2-B1

En cas d'utilisation d'une embase Cinch, on peut utiliser l'amplificateur sans transformateur d'entrée. Le contact de masse de l'embase Cinch est alors à relier à la masse et le contact de signal à C1 (entrée de signal).

La liste des composants intitulée de l'alimentation en version stéréophonique (un canal) concerne la platine de l'alimentation stéréo qui n'a pas pu, pour des raisons de place, être incluse dans l'article. Nous la proposerons dès que possible au téléchargement sur notre site dans la section correspondante.

La LED présente dans le bloc « POWER » du plan de câblage (figure 6) est purement symbolique. En effet le circuit de signalisation de la tension se compose en fait d'une LED, d'une résistance de 470 Ω et d'une diode du type 1N4007, ces différents composants étant pris en série.

Les courants de repos des tubes de sortie de puissance sont respectivement de 35 mA pour le EL34 et de 65 mA pour le KT88. Par la prise de résistances de 1kΩ5 en parallèle sur R17 et R18, l'interrupteur « STAND BY » réduit le courant de repos à de l'ordre de 20% de la valeur nominale. Cet interrupteur est à câbler de la manière suivante.

La LED du bloc « STAND BY » est dotée d'une résistance-série de 1kΩ5 et polarisée de façon à ce que son anode soit reliée au point nodal R18/C14/P2 de la platine. Lorsque l'interrupteur « STAND BY » est fermé, la LED et sa résistance-talon de 1kΩ5 sont prises en parallèle sur R18, la LED s'allumant. Il faudra intercaler une résistance de 1kΩ5 dans la ligne reliant la résistance R17 au (second) contact de l'interrupteur « STAND BY ». Lors de sa fermeture, ce contact prendra la résistance de 1kΩ5 en parallèle sur R17.

Voici, pour info, le diamètre des conducteurs ayant servi au câblage de l'amplificateur : 1,5 mm^2 pour la tension de chauffage, 0,75 mm^2 pour la tension secteur, 0,5 mm^2 pour la HT et 0,25 mm^2 pour la polarisation de grille et les LED.

(000118)

Récepteur à M.A., Elektor n° 272, février 2001, page 8 et suivantes

Sur le circuit imprimé, le contact de masse de l'entrée d'antenne doit être relié au plan de masse de la platine.

(000176)

Timer pour chambre noire, Elektor n° 276, juin 2001, page 21 et suivantes

Dans la liste des composants la dénomination de IC3 est, comme le mentionne le schéma, S210S02 (Sharp) et non pas S201S01. Le S01 ne dispose pas d'un détecteur de passage par zéro et produira partant plus de bruit électrique que le S02. Il n'en reste pas moins que les 2 composants sont utilisables dans ce montage.

(000182)

GBDSO : Game Boy Digital Sampling Oscilloscope

Note Importante

Les embases destinées à recevoir les sondes (probes) doivent impérativement être du type jack 3,5 mm **STÉRÉO**. Le signal d'entrée arrive par le biais du contact central, les 2 autres contacts étant ceux de la masse et du blindage de la sonde. Il ne saurait être question d'utiliser un jack MONO sachant que cela entraînerait la mise en court-circuit du signal de mesure.

L'intérieur des boîtiers des cartouches de Nintendo comportent 2 petites butées qu'il faudra éliminer; en effet, l'une de ces butées peut venir s'appuyer sur le condensateur C25 au point d'en provoquer le desssoudage de sur la platine !

(90082)