ELEKTOR

ELEKTOR

ÉLECTRONIQUE et MICRO-INFORMATIQUE

Nº 283
JANVIER 2002
mensuel
33 F/€ 5.03
234 FB/€ 5.80
10.40 FS • 6.90 $Can.
ISSN 0181-7450
www.elektor.presse.fr

UART pour USB

Cours «Microcontrôleur»

Commande & Réglage par messages SMS

Testeur S/PDIF

EdiTS Pro 1.2

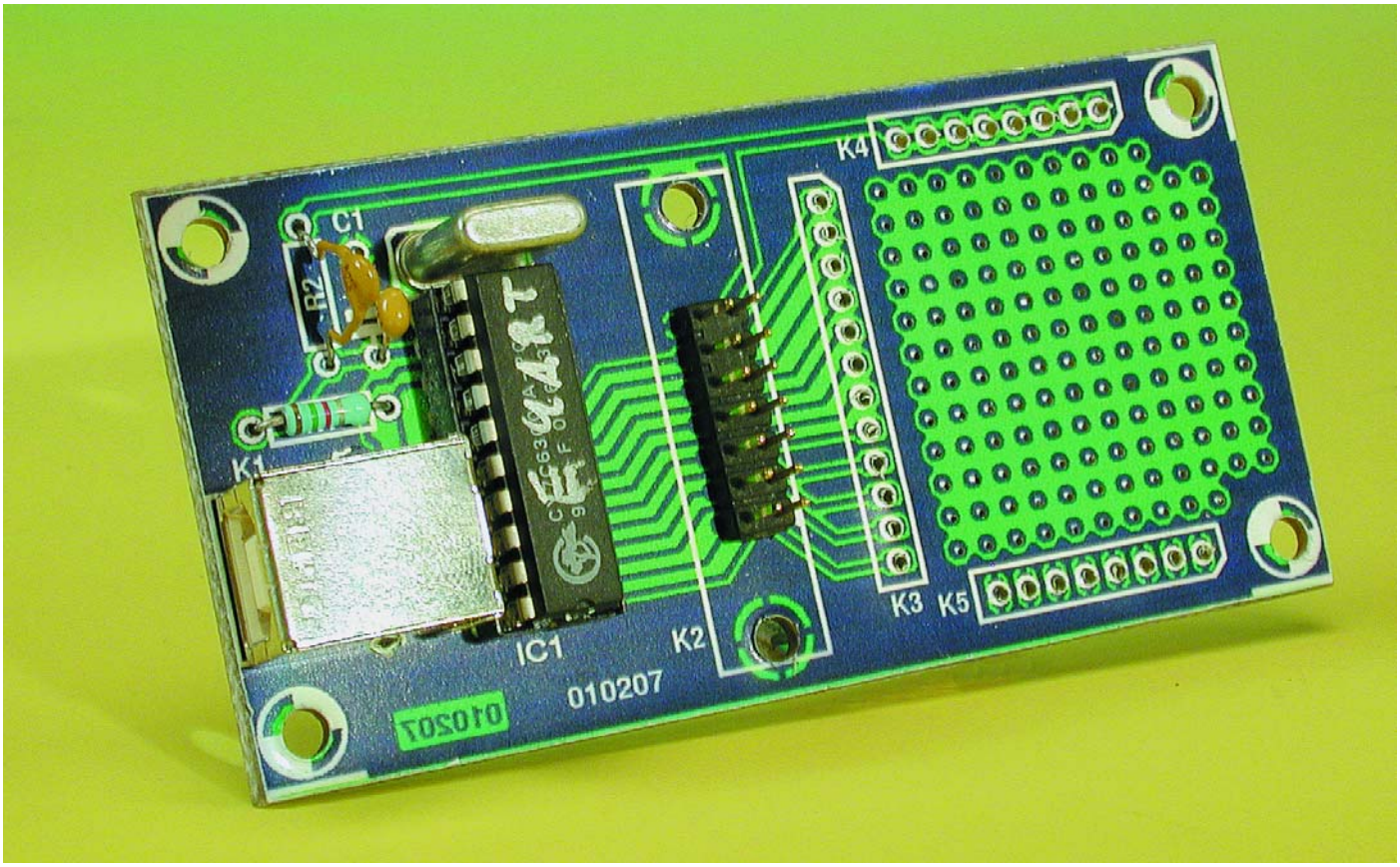
Commande de feux et de boîte de vitesses

UART pour USB

Libre accès à tous les ports

B. Kainka

Les ports USB sont bons à tout faire, en électronique. Mais contrairement à l'interface RS-232, on ne trouvait, jusqu'à ce jour, aucun convertisseur pour échanger des données sérielles USB avec un port parallèle. Les choses changent, avec un UART en USB d'emploi général.



Tout comme le légendaire UART (*Universal Asynchronous Receiver-Transmitter*, interface asynchrone) AY3-1015 fournissait une solution purement électronique pour créer un port RS-232, il existe aujourd'hui la puce USB d'Elektor qui offre un accès général à l'USB (*Universal Serial Bus*). Il ne faut pas y ajouter beaucoup plus qu'un résonateur à la céramique ou un quartz et deux condensateurs pour se confectionner une interface USB complète. Vous disposez alors de 12 ports accessibles en entrée comme en sortie. En outre, on peut régler le courant sur chacun d'eux, pour commander directement la luminosité d'une LED, par exemple.

Avec les RS-232, nous étions habitués à devoir nous limiter dans le nombre de voies disponibles, nous n'avions souvent plus qu'un seul

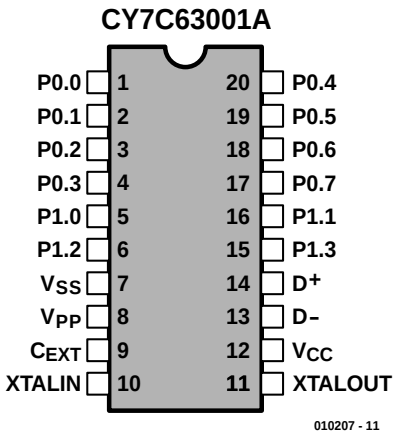


Figure 1. Brochage du circuit intégré pour port USB.

Vss, Vpp	Masse
Vcc	+5 V
D+, D-	Lignes de données USB
XTALIN, XTALOUT	Résonateur céramique pi quatrz 6 MHz
P0.0 à P0.7	Port 0 à 8 lignes
P1.0...P1.3	Port 1 à 4 lignes

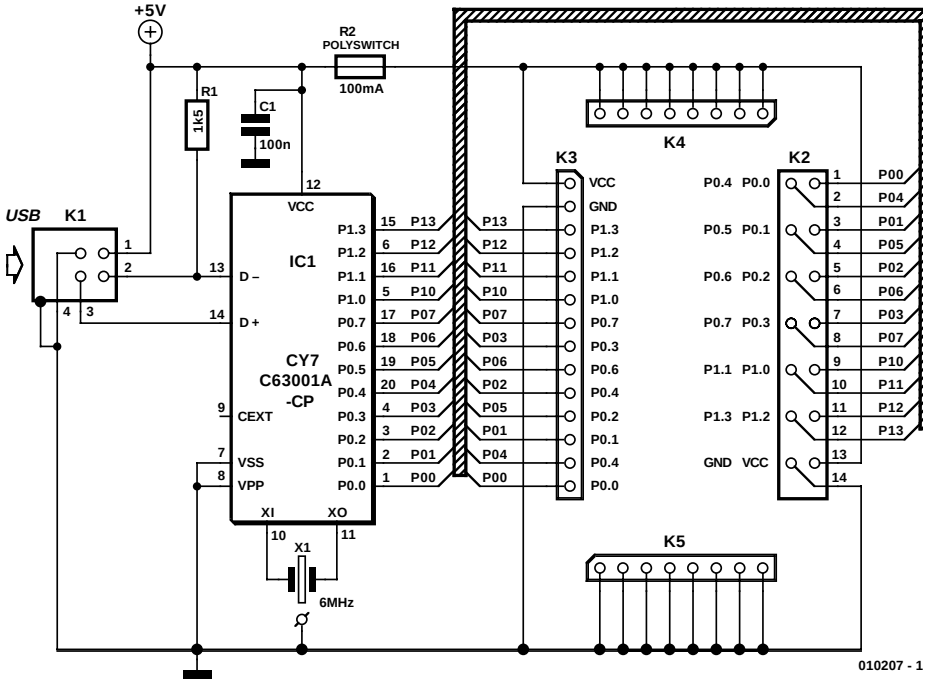


Figure 2. L'électronique de la platine d'expérimentation se limite à l'application de base du CY7C63001.

port COM libre. En ajouter d'autres n'était pas facile. Comme le système USB est par définition un bus, il permet le raccordement de divers périphériques. Y relier plusieurs appareils du même type, ce n'est plus qu'une question de **pilote** (*driver*). Le pilote pour le UART USB autorise différentes combinaisons. On peut ainsi en équiper plusieurs appareils et les utiliser ensemble avec le même PC. La plupart du temps, une carte mère récente est dotée de deux prises USB. Celui qui veut en utiliser davantage doit se procurer un distributeur de bus (*hub*, littéralement le moyeu, auquel sont attachés les rayons).

Le montage, tout comme la première interface USB d'Elektor (n° 267, septembre 2000), se base sur le microcontrôleur CY7C63001A de Cypress, dont le brochage est reproduit à la **figure 1**. Comme il ne s'agit plus ici d'une application particulière dans un périphérique, on pourra se servir de tous les ports. En outre, le courant disponible n'est plus réservé à une seule prise.

Le circuit intégré est livré avec son pilote USB. Pour la cause, Elektor s'est affilié au club des concepteurs USB, a sollicité et obtenu de l'Organisation USB une **licence** de distributeur agréé. Vous aurez ainsi, en qualité de lecteur, la possibilité de télécharger le pilote en question et de concevoir votre propre appareil USB, sans avoir à demander d'autre autorisation ni à remplir de formalités fastidieuses.

La **figure 2** présente un montage classique du UART USB. Vous y remarquez d'emblée la liaison directe des lignes USB D+ et D–. L'alimentation (+5 V) passe souvent par l'USB. La chronométrie utilise un résonateur céramique bipolaire ou un quartz, mais comme la platine comporte aussi une troisième pastille, les modèles tripolaires sont les bienvenus. Les condensateurs internes peuvent, mais ne doivent pas nécessairement retourner à la masse, pour s'additionner en parallèle. Depuis l'apparition de la version A du processeur, les problèmes de l'oscillateur interne sont résolus, on peut y brancher sans crainte un quartz de 6 MHz. La **figure 3** montre les trois variantes possibles d'oscillateur.

Tous les ports sont pourvus d'une résistance intégrée de rappel au niveau haut (*pull-up*) et peuvent donc servir d'entrée sans autre formalité. Le port 0 et le port 1 diffèrent en courant de drain. Le port 0 commute au maxi-

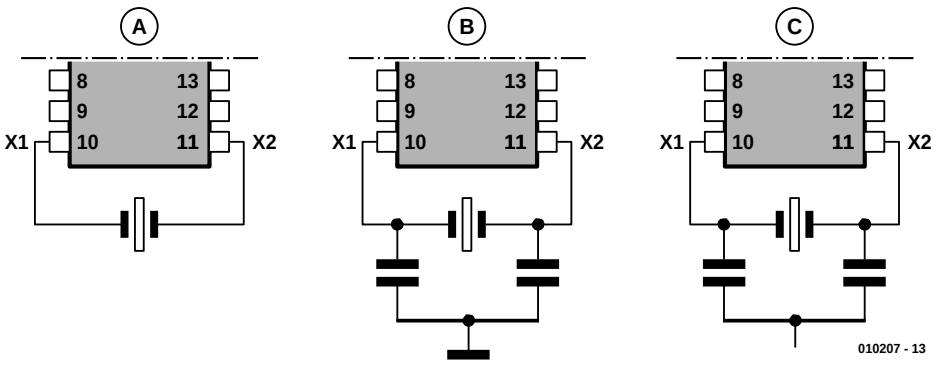


Figure 3. Variations d'oscillateur (6 MHz) A : Quartz ou résonateur 2 broches
B : résonateur 2 ou 3 broches à connexion de masse
C : résonateur 2 ou 3 broches sans connexion de masse.

Liste des composants

Résistances :
R1 = 1kΩ5
R2 = Polyswitch 100 mA

Condensateurs :
C1 = 100 nF

Semi-conducteurs :
IC1 = CY7C63001A (Cypress), (programmé **EPS010207-41**)

Divers :
K1 = câble USB type B
K2 = embase mâle à 2 rangées de 7 contacts avec détrompeur
K3 à K5 = ensemble de 20 picots
X1 = quartz 6 MHz ou résonateur à 2 ou 3 broches

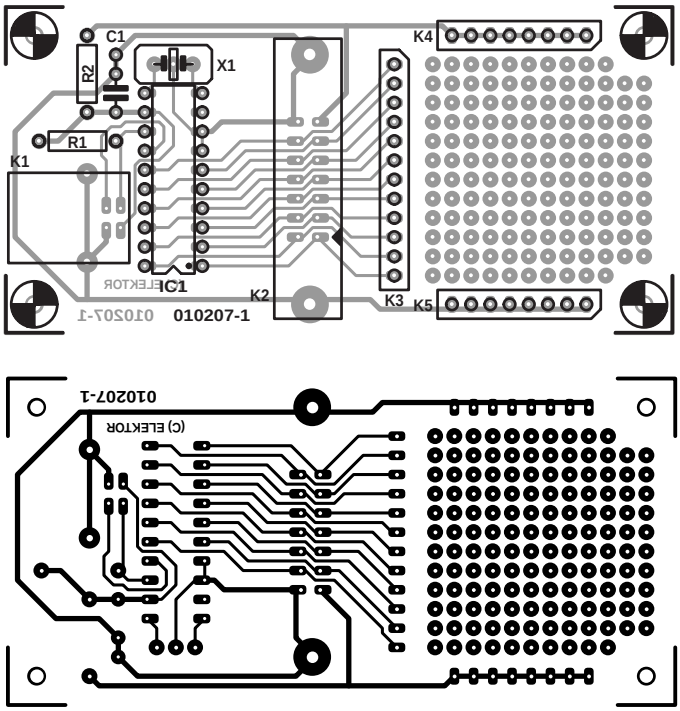


Figure 4. Dessin des pistes et sérigraphie de la platine d'expérimentation.

mum 1,5 mA, le port 1 va jusqu'à 15 mA et accepte donc d'attaquer directement une LED, par exemple.
Notre UART USB est un circuit intégré destiné aux montages personnels, commande d'une cafetière électrique ou maillon d'une centrale d'alarme, peu importe. Quelqu'un s'en servira peut-être pour superviser la conduite de ses trains miniatures ou le greffera sur un robot. Dans chaque cas, ce n'est rien d'autre qu'une puce et un circuit de base. Mais pour guider vos premiers pas, nous vous proposons une platine d'expérimentation. Tous les ports y sont à votre disposition. La petite platine de la **figure 4** peut se monter en impériale sur une plus grande et servir de module d'E/S universel.

Installation

Utiliser un appareil en USB sous Windows réclame toujours un pilote, qui lui permet de voir le UART USB comme un organe indépendant. Tout accès du logiciel d'application passe par le pilote.
La disquette disponible auprès des adresses habituelles sous la référence **EPS010207-11** porte le pilote **USBuart.sys** et un fichier d'information **USBuart.inf**, ainsi que quelques logiciels à titre d'exemple. Windows doit aller chercher la classification de l'appareil dans son pilote, le copier et adapter la banque de données des pilotes de périphériques. La disquette n'est indispensable que lors de la première connexion. Ultérieurement, le pilote sera chargé automatiquement, lorsque la puce entrera en contact avec l'USB.
Chaque fois qu'un périphérique USB est raccordé pour la première fois, l'annonce reproduite à la **figure 5** s'affiche.
Il y a détection de la présence d'un nouveau périphérique à cause de la résistance entre Vcc et D–. La ligne de donnée D– est ainsi forcée au niveau haut pour informer le PC qu'il s'agit d'un périphérique USB lent. Ensuite, Windows demande les spécifications du nouvel appareil. Celles qui l'intéressent le plus sont : Vendor-ID (Elektor = 0C7D) et Device-ID (USB-UART = 0001), qui permettent un recensement univoque des périphériques. Grâce à ces deux informations, Windows peut à présent rechercher le pilote adéquat. Il interroge en premier lieu la banque de données des pilotes connus sur le PC même. S'il n'en trouve aucun valable, il demande à l'utilisateur de fournir une disquette porteuse des informations et du pilote.
Pour installer le pilote, il suffit de suivre les instructions à l'écran. Le pilote se copie automatiquement dans le répertoire `Windows\System32`. Le fichier Inf se place dans `Windows\Inf`. En outre, le pilote se charge dans la mémoire vive du PC. On peut en voir le résultat dans `Windows\System\Iosubsys\` : le pilote pour UART-USB est chargé (**figure 6**).
L'installation se passe d'habitude sans difficulté. Mais la multitude de

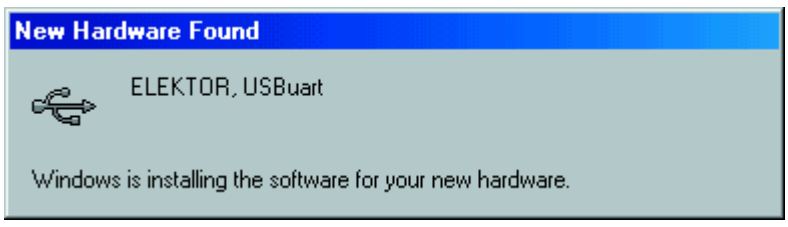


Figure 5. Le Wizard découvre un nouveau périphérique.

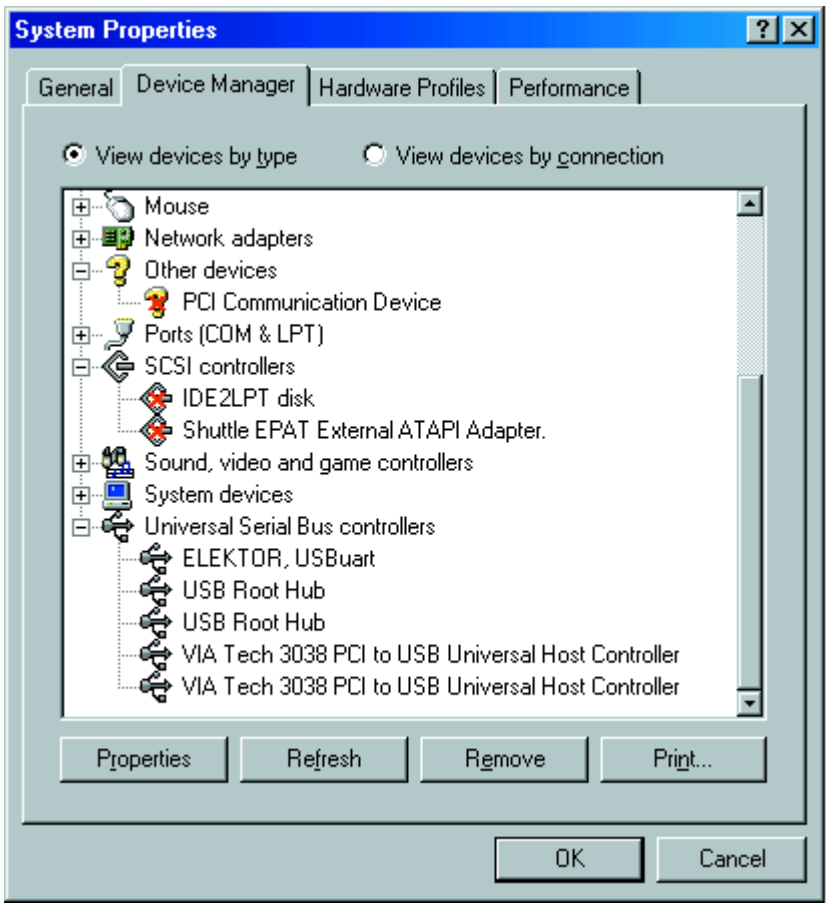


Figure 6. Chargement du pilote Elektor.

versions de Windows rend les essais pour chaque cas de figure bien compliqués. Dans certains cas, il se peut que Windows s'escrime à réclamer le CD d'installation de Windows (**figure 7**). En pareil cas, on lui désigne comme source le fichier **USBuart** sur la disquette A : et on lui dit de poursuivre par OK.

Le pilote fonctionne aussi sous Windows 2000. Cependant, il ne possède pas de signature numérique, que Microsoft vend très cher aux grandes firmes pour certifier la compatibilité du logiciel. Alors, on se débrouille comme suit. Dans Panneau de configuration – Propriétés Système - Gestionnaire de périphériques – Périphériques Système - Signature de périphérique, on indique qu'on veut l'installer sans signature. Lors de l'initialisation du système, il arrive que Windows 2000 ne copie pas le pilote dans le bon répertoire, auquel cas, on le copiera soi-même dans le répertoire System32. Le pilote utilisé ici se base dans une large mesure sur un développement de la firme Anchor Chips, actuellement reprise par Cypress. Celle-ci met à disposition via l'adresse Internet www.cypress.com une documentation détaillée pour la conception en USB.

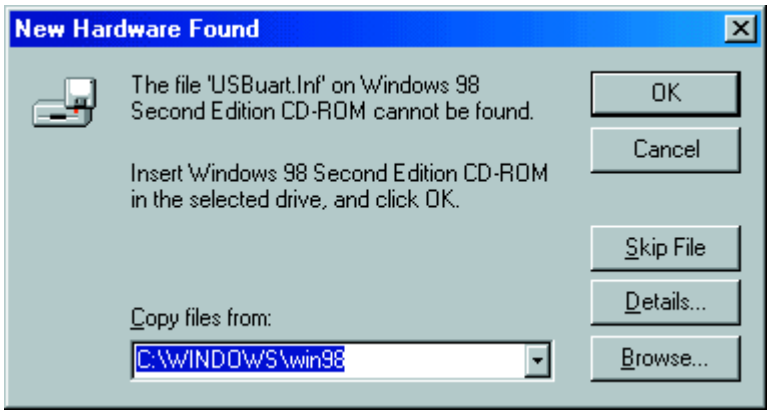


Figure 7. Recherche infructueuse typique d'un pilote Windows adéquat.

Adaptation du pilote en Visual Basic

Tout appareil en USB doit posséder un pilote, qui se chargera automatiquement (*Plug and Play*) lors de la connexion. Un logiciel ne peut communiquer avec l'appareil que par le truchement du pilote. Pour que Windows sache quels pilotes il doit charger, chaque périphérique doit se présenter spontanément lors de l'énumération (l'inscription comme participant au bus) et décliner les informations nécessaires à son identification indubitable. Au moment de la première rencontre, Windows s'aperçoit qu'un petit nouveau s'est fait remarquer, donc qu'il lui faut installer un pilote en priorité. Le module **USB1.bas** propose la confection d'un pilote en Visual Basic. Il offre les fonctions de base suivantes :

WrPort0, WrPort1 : écriture de l'état du port
RdPort0, RdPort1 : lecture de l'état du port
WrPullups : activation des résistances de rappel au niveau haut
WrIsink : réglage du courant que chaque port peut drainer.

Le pilote autorise l'utilisation de plusieurs appareils en parallèle sur le même UART USB. Le pilote virtuel du premier périphérique sera :

```
sFileName="\\.\usbuart_0"
```

pour le deuxième périphérique :

```
sFileName="\\.\usbuart _1"
```

et ainsi de suite.

Lecture et écriture par le port USB

Le premier exemple de programme, **USBuart1.vbp** permet l'accès à tous les ports. D'un simple clic, on peut faire passer au niveau haut chaque ligne, brancher ou

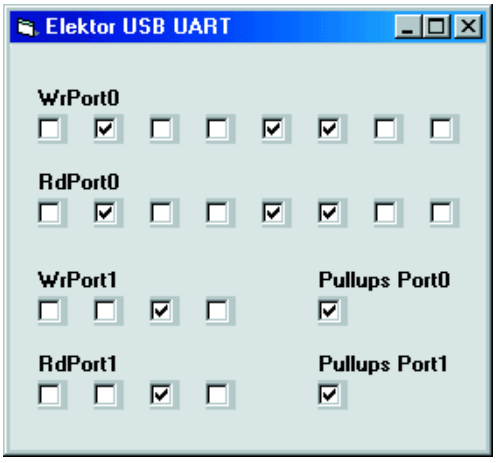


Figure 8. Accès au port sous Visual Basic.

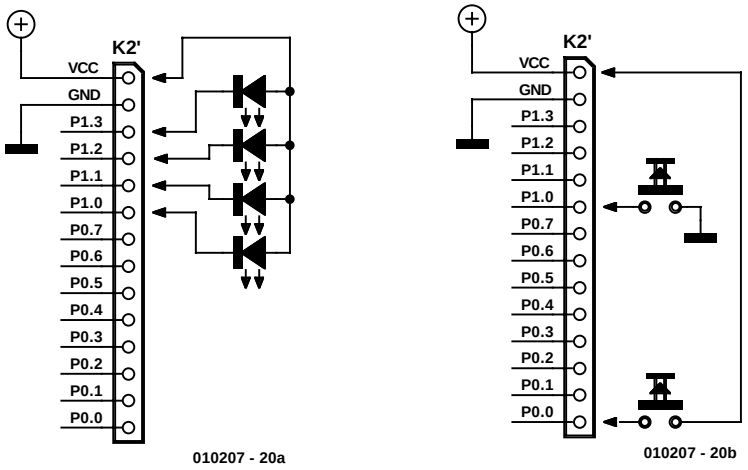


Figure 10. Exemple de branchement d'interrupteurs et de LED.

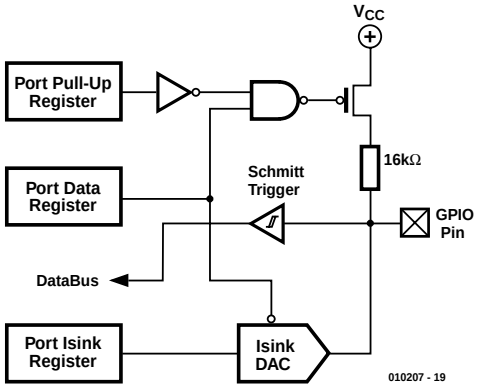


Figure 9. Structure interne d'une broche de port à courant ajustable.

Brochages

K1: Embase USB

1	+5V
2	D-
3	D+
4	Gnd

K2:

P00	1	2	P04
P01	3	4	P05
P02	5	6	P06
P03	7	10	P11
P12	11	12	P13
Vcc	13	14	Masse (Gnd)

K3:

1	P13
2	P12
3	P11
4	P10
5	P07
6	P03
7	P06
8	P02
9	P05
10	P01
11	P04
12	P00

K4	8 x Vcc
K5	8 x Masse

isoler les résistances de rappel haut d'un port. En l'absence d'intervention extérieure, le logiciel ne change rien aux sorties, si ces résistances sont en service. Si on les débranche, alors les lignes passent à haute impédance et leur niveau n'est plus défini. Mais une ligne commandée au niveau bas, sans résistance *pull-up*, est et reste

Listage 1. Les procédures interface nécessaires dans le module USB1.basType SECURITY_ATTRIBUTES

```
nLength As Long
lpSecurityDescriptor As Long
bInheritHandle As Long
End Type

Type OVERLAPPED
 Internal As Long
 InternalHigh As Long
 offset As Long
 OffsetHigh As Long
 hEvent As Long
End Type
Declare Function CreateFile Lib "kernel32" Alias "CreateFileA" (ByVal lpFileName As String, ByVal dwDesiredAccess
As Long, ByVal dwShareMode As Long, lpSecurityAttributes As SECURITY_ATTRIBUTES, ByVal dwCreationDisposition As
Long, ByVal dwFlagsAndAttributes As Long, ByVal hTemplateFile As Long) As Long
Declare Function DeviceIoControl Lib "kernel32" (ByVal hDevice As Long, ByVal dwIoControlCode As Long, lpInBuffer
As Any, ByVal nInBufferSize As Long, lpOutBuffer As Any, ByVal nOutBufferSize As Long, lpBytesReturned As Long,
lpOverlapped As OVERLAPPED) As Long
```

```
Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long

Public Security As SECURITY_ATTRIBUTES
Public gOverlapped As OVERLAPPED
Public hgDrvrHnd As Long
Public Const GENERIC_READ = &H80000000
Public Const GENERIC_WRITE = &H40000000
Public Const FILE_SHARE_WRITE = &H2
Public Const FILE_SHARE_READ = &H1
Public Const OPEN_EXISTING = &H3

Dim sFileName As String
Dim htemp As Long
Dim lIn As Long, lInSize As Long, lOut As Long, lOutSize As Long, lSize As Long
Dim lTemp As Long

Public Sub USB_IO()
  sFileName = "\\.\usbuart_0"
  hgDrvrHnd = CreateFile(sFileName, GENERIC_WRITE Or GENERIC_READ, FILE_SHARE_WRITE Or FILE_SHARE_READ, Security,
OPEN_EXISTING, 0, 0)
  lTemp = DeviceIoControl(hgDrvrHnd, 4&, lIn, lInSize, lOut, lOutSize, lSize, gOverlapped)
  htemp = CloseHandle(hgDrvrHnd)
End Sub

Public Function RdPort0() As Integer
  lIn = 0 * 256 + 20
  lInSize = 2
  lOutSize = 2
  USB_IO
  RdPort0 = (lOut / 256) And 255
End Function

Public Function RdPort1() As Integer
  lIn = 1 * 256 + 20
  lInSize = 2
  lOutSize = 2
  USB_IO
  RdPort1 = (lOut / 256) And 255
End Function

Public Sub WrPort0(Wert)
  lIn = 65536 * Wert + 0 * 256 + 21
  lInSize = 3
  lOutSize = 1
  USB_IO
End Sub

Public Sub WrPort1(Wert)
  lIn = 65536 * Wert + 1 * 256 + 21
  lInSize = 3
  lOutSize = 1
  USB_IO
End Sub

Public Sub WrIsink(Pin, Wert)
  lIn = 65536 * Wert + Pin * 256 + 23
  lInSize = 3
  lOutSize = 1
  USB_IO
End Sub

Public Sub WrPullups(Port, Wert)
  lIn = 65536 * (255 - Wert) + (Port + 16) * 256 + 23
  lInSize = 3
  lOutSize = 1
  USB_IO
End Sub
```

bien au niveau bas.
Les lignes de communication du contrôleur sont quasi bidirectionnelles, comme celles du microcontrôleur 8051. Particularité, on peut ajuster leur courant de drain. Elles peuvent attaquer directement une LED, dont la luminosité sera donc réglable. Elles ne peuvent cependant pas fournir (comme source) de débit supérieur à 300 μA. Le réglage du courant que chaque ligne peut drainer s'opère à l'aide d'un convertisseur N/A d'une résolution de 4 bits. La plage ainsi couverte par chacune des lignes du port 0 s'étend de 0,3 mA à 1,5 mA, mais pour le port 1, la plage couvre de 4,8 mA jusqu'à 15 mA. Les deux ports sont équipés de résistances de rappel haut de 16 kΩ, que l'on peut utiliser ou non. Cette sélection, tout comme les réglages de débit de chaque port se font par USB. Rappelons brièvement les différentes caractéristiques des ports :

– entrées du port à haute impédance, avec les caractéristiques d'une entrée CMOS
– entrées dotées de résistances *pull-up* de 16 kΩ
– sorties compatibles CMOS
– sorties à drain ouvert pour la commande directe de LED, par exemple
– courant de drain réglable par convertisseur N/A à 4 bits
– convertisseur A/N simple à 4 bits de résolution.

Le programme **USBuart1** permet une investigation précise des propriétés des ports. Tout comme pour les ports quasi bidirectionnel du microcontrôleur 8051, c'est un FET (**figure 9**) qui ramène la ligne au niveau bas quand elle doit délivrer un zéro en sortie. En réalité, le courant qui assure normalement l'excursion basse est très petit, 0,3 mA au port 0 et 4,8 mA au port 1. C'est pourquoi, pour le port 0, une résistance de 10 kΩ vers Vcc suffit déjà à attirer la ligne au niveau haut, malgré les efforts du transistor de sortie. On se retrouve alors avec un « 1 » logique, alors que la sortie aurait dû rester à zéro. Dans la deuxième partie de cet article, nous verrons comment il est possible d'augmenter par étapes le courant de drain.
Ces propriétés particulières des ports du processeur permettent de sortir résolument des traditions consacrées et de travailler en logique positive par rapport à Vcc, comme on le voit à la **figure 10**. Mais ceci n'est vrai qu'avec le port 0, sur l'autre, les courants risqueraient de devenir trop forts. Le port 1, en effet, se prête bien à l'attaque directe d'une LED dont l'autre borne est reliée à Vcc, sans aucune résistance en série. La limitation de courant s'opère à l'intérieur du port. Rien qu'avec le courant prévu d'origine, soit 4,8 mA, on peut commander un afficheur à LED bien visible. Sur le port 0, en revanche, si l'on veut brancher des diodes électroluminescentes, mieux vaut n'utiliser que des composants à haut rendement sous 1,5 mA.

(010207-I)

Dans la deuxième partie, nous verrons comment régler le courant des ports et, en prime, comment rédiger le logiciel d'un convertisseur A/N simple.

Listage 2.

Accès de port généraux par le biais de USBuart1.vbp

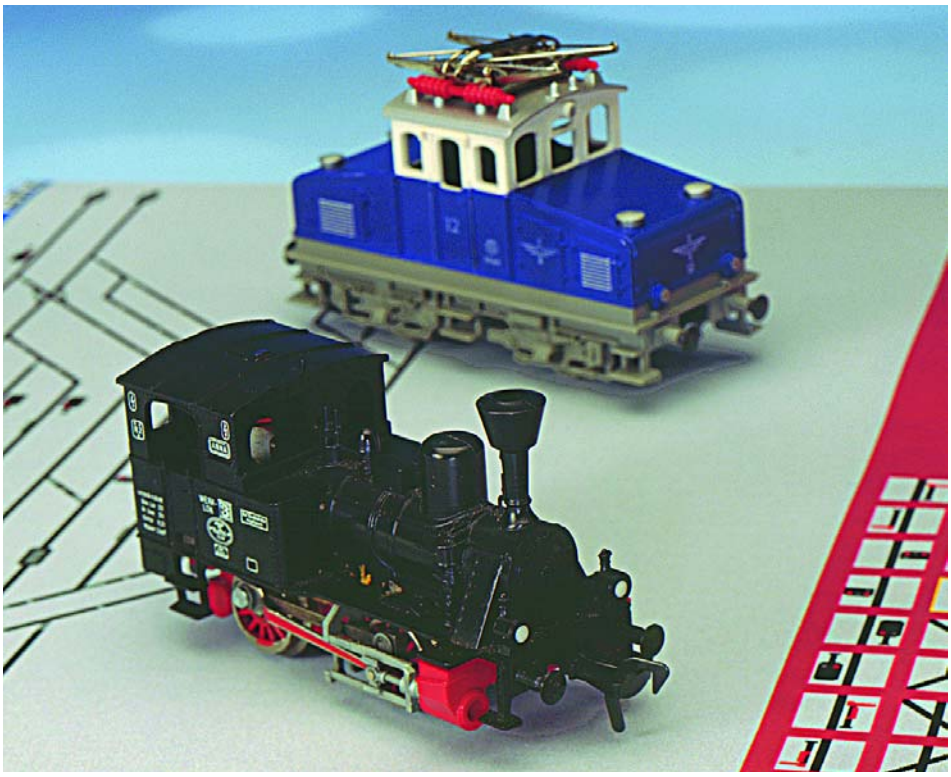
```
Private Sub Check25_Click()
  If Check25.Value = 1 Then
    WrPullups 0, 255
  Else
    WrPullups 0, 0
  End If
End Sub

Private Sub Check26_Click()
  If Check25.Value = 1 Then
    WrPullups 1, 255
  Else
    WrPullups 1, 0
  End If
End Sub

Private Sub Form_Load()
  WrPullups 0, 255
  WrPullups 1, 255
End Sub

Private Sub Timer1_Timer()
  Dat = 0
  Dat = Dat + Check1.Value
  Dat = Dat + Check2.Value * 2
  Dat = Dat + Check3.Value * 4
  Dat = Dat + Check4.Value * 8
  Dat = Dat + Check5.Value * 16
  Dat = Dat + Check6.Value * 32
  Dat = Dat + Check7.Value * 64
  Dat = Dat + Check8.Value * 128
  WrPort0 Dat
  Dat = RdPort0()
  Check9.Value = Dat And 1
  Check10.Value = (Dat And 2) \ 2
  Check11.Value = (Dat And 4) \ 4
  Check12.Value = (Dat And 8) \ 8
  Check13.Value = (Dat And 16) \ 16
  Check14.Value = (Dat And 32) \ 32
  Check15.Value = (Dat And 64) \ 64
  Check16.Value = (Dat And 128) \ 128
  Dat = 0
  Dat = Dat + Check17.Value
  Dat = Dat + Check18.Value * 2
  Dat = Dat + Check19.Value * 4
  Dat = Dat + Check20.Value * 8
  WrPort1 Dat
  Dat = RdPort1()
  Check21.Value = Dat And 1
  Check22.Value = (Dat And 2) \ 2
  Check23.Value = (Dat And 4) \ 4
  Check24.Value = (Dat And 8) \ 8
End Sub
```

EDiTS Pro 1.2

À programme nouveau, possibilités nouvelles

Steffen van de Vries

Il a déjà coulé beaucoup d'eau sous les ponts depuis le dernier article consacré à EDiTS Pro, notre système de pilotage de réseau ferroviaire modulaire. Si l'on tient compte du nombre d'extensions réalisées par l'auteur, la durée de cette période de gestation se justifie indubitablement. Grâce aux étroits contacts qu'il a établi avec les groupes d'utilisateurs (n'est ce pas Mr Dezaire) il a pu se faire une bonne idée quant à la direction à donner aux nouveaux développements concernant EDiTS Pro.



La « liste de souhaits » ainsi établie que l'on retrouve d'ailleurs dans l'ouvrage consacré au sujet [1] a été réalisée de bout en bout dans les nouveaux matériels et logiciel. Nous allons lui consacrer 2 articles.

Un bref condensé

Programme EDiTS Pro

Au niveau du programme pour EDiTS Pro, auquel nous consacrerons le premier article, l'extension la plus marquante se situe dans les lignes de programme utilisées pour le pilotage du réseau ferroviaire.

Il est possible maintenant d'ajouter, dans ces lignes de programme, des fonctions et des informations de vitesse, et cela même pour les trains qui ne sont pas dotés du système de détection de train à infrarouge d'EDiTS Pro. L'une des fonctionnalités importante à ce niveau est le suivi virtuel d'adresses par le biais duquel il est possible de connaître, pour toute locomotive et ce quelle que soit la marque de son décodeur, son adresse de décodeur et partant où elle se trouve.

Ce suivi virtuel d'adresses est réalisé sans avoir à établir de tableau de programme par train (succession de cantons) et ne requiert qu'un nombre minimum de lignes de programme saisies en mode de programmation. Il est même possible, par exemple, d'utiliser, sur la voie libre, un système-bloc électrique conventionnel.

Unité de commande EDiTS Pro

La première raison d'être du développement d'EDiTS Pro était la réalisation d'une interface pour ordina-

Figure 1. Le panneau d'étiquetage permet de doter chaque symbole d'un numéro pouvant prendre 5 couleurs différentes.

teur. Sachant cependant que l'unité de commande est facile à réaliser rapidement et que l'on dispose alors de 8 régleurs à format Motorola étendu, l'unité de commande est un montage très populaire, même chez les utilisateurs de systèmes autonomes (les amateurs de réseaux à échelle faible en combinaison avec le gros booster EDiTS).
Le pilotage des locomotives ne pose pas le moindre problème à l'unité de commande. Les choses ne commencent à se compliquer que lorsqu'il y a plus de 8 locomotives sur le réseau et qu'il faut commander des aiguillages. La nouvelle unité de commande apporte une réponse à ces questions. Il est possible de la démarrer dans un mode spécial grâce auquel il devient possible de modifier les adresses affectées aux régleurs manuels, commander aiguillages et signaux de circulation devient possible sans plus avoir recours au PC.
L'auteur a de plus, en collaboration avec Jürgen Freiwald (Railroad & Co), ajouté de nouvelles instructions au set de sorte que l'unité de commande a vu son accessibilité s'améliorer surtout pour celui qui envisage d'écrire son propre logiciel de pilotage (professionnel ou amateur éclairé).
Un autre aspect, celui de la vitesse de lecture des répondeurs, a pris une importance capitale lors de ce nouveau développement de l'unité de commande. Avec la première version, bien que normalement très rapide, l'interrogation des répondeurs présentait, sous certaines conditions défavorables, quelques lenteurs. Avec la nouvelle version, ce problème a totalement disparu grâce à une astucieuse gestion du temps. L'interrogation est intégrée au cycle d'émission des ordres sur la voie (pendant lequel d'inévitables temps morts existent), et n'est plus effectuée après l'émission de ces mêmes ordres. Petite nuance qui fait toute la différence... Avec la nouvelle version d'unité de commande, cette vitesse de lecture a été multipliée par 16, le système disposant en outre d'une instruction permettant de savoir, par interrogation, quelles unités ont vu leur état d'entrée changer depuis la lecture précédente.

Nouveau matériel

La nouvelle unité de commande ne requiert pas de nouveau circuit imprimé, on pourra fort bien utiliser la platine du contrôleur d'origine. Lors de la conception de cette dernière il avait déjà été tenu compte d'une possibilité d'activation de nouvelles fonctions (telles qu'adressage et claviers). Même la signalisation d'un court-circuit sur la voie (autre nouvelle fonction du contrôleur) ne requiert pas de modification.
Nous avons cependant dessiné une nouvelle platine pour le paramétrage des adresses des régleurs. Il est possible, par le biais de cette nouvelle platine et d'un rotacteur à 8 positions, de modifier l'adresse de chaque régleur manuel.
Il est également possible de réaliser une version de luxe n'impliquant pas moins de 8 de ces platines, ce qui permet un paramétrage direct de l'adresse de chaque régleur et qui, de plus, visualise instantanément, par le biais d'un affichage à 7 segments, l'adresse attribuée à chaque régleur.

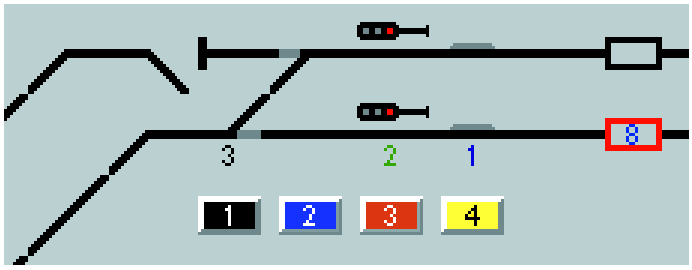


Figure 2. Il est également possible maintenant d'utiliser des aiguillages anglais à double bobine.

Le logiciel

Généralités

Problèmes de vitesse du processeur

Dans le monde des ordinateurs personnels les choses vont très vite. À l'époque du développement d'EDiTS Pro, la vitesse des processeurs venait juste de passer la barre des 100 MHz. Aujourd'hui personne ne s'étonne de voir la vitesse maximale faire des bons de 500 MHz (dernier Pentium 4). Il est peu probable, en règle générale que l'on utilise ce type de monstre de vitesse pour lui confier l'humble tâche de piloter un réseau ferroviaire miniature, mais les systèmes les plus rapides d'aujourd'hui sont les « esclaves » de demain, cela arrivera tôt ou tard. Il apparaît que le système EDiTS ancienne mouture se croise les bras à partir d'une vitesse de processeur de l'ordre de 400 MHz.
La solution à ce problème est aussi simple que son origine. Sur les systèmes à processeur lent, le nombre d'interrogations du port sériel s'est toujours avéré suffisant pour permettre à l'unité de commande de répondre; avec un système à processeur rapide le nombre d'interrogations est atteint avant même que l'unité de commande n'ait eu le temps de réagir, ce qui se traduit par une pause (time-out) : tout s'arrête. IL a suffit d'augmenter ce nombre pour que tout revienne dans l'ordre.

Vitesse du programme

Avec cette nouvelle version du logiciel, le pilotage en temps réel des convois joue une tôle important et dans ce cadre la visualisation (graphique) du réseau relativement lente pose un problème. La visualisation requérait tant de temps processeur que pendant ce processus de dessin du réseau sur l'écran et de libération des voies il était impossible au système de réagir aux informations fournies par les répondeurs. Dans le cas le plus défavorable ceci pouvait se traduire par un retard de quelques secondes.
En vue de résoudre ce problème nous avons changé notre fusil d'épaule en ce qui concerne la visualisation : nous avons abandonné les calculs graphiques pour passer à la modification pixellisée de l'image. Un

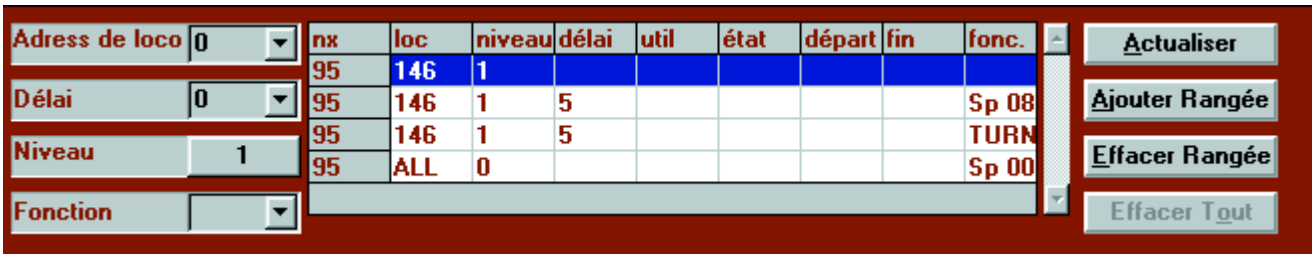


Figure 3. On peut, avec cette nouvelle version, travailler avec le niveau 0 dans l'écran de programmation.

exemple : s'il fallait, auparavant, pour la mise en place d'une ligne pointillée jaune, modifier le symbole complet de 27 x 27 pixels, on se contente maintenant de modifier les 2 surfaces jaunes. Non seulement cela se traduit par une augmentation impressionnante de la vitesse, mais offre en outre la possibilité de doter les symboles d'aiguillage et de signal que l'on aura définis soi-même d'une étiquette (label).
Ce choix a cependant eu comme conséquence d'avoir à opter pour la plus grande uniformité possible. De ce fait, les boutons de répondeurs et de détection ont maintenant la même forme, leur distinction se faisant au niveau de leur couleur grise ou blanche.

Compatibilité
La version 2 d'EdiTS Pro est bien entendu compatible avec la version 1. Cependant, eu égard au profond remaniement du pilotage automatique des convois, la ressaisie des lignes de programme est vivement conseillée, voire obligatoire. Dans la continuité, l'ancienne unité centrale n'intègre évidemment pas les nouvelles instructions issues de ce remaniement. Si l'on tient tout à la fois à conserver l'ancienne UC et à faire évoluer le SOFT, on aura soin à ne pas utiliser ces instructions « nouvelle génération ».
Compte-tenu de l'amélioration apportée, on admettra de bonne grâce ce petit inconvénient.

Modes de programme
De manière à mieux pouvoir recenser les nouvelles adjonctions et modifications nous les avons groupées en fonctions des points de menu.

Création de tableau de commande
L'écran de création de réseau est le seul à ne pas avoir subi la moindre modification.

Adresses décodeur
La modification majeure est visible dans le coin inférieur gauche de l'écran (**figure 1**). Si, à l'origine, on trouvait à cet endroit affiché, auprès d'un symbole, ses index et index nx (ce qui pouvait à l'occasion entraîner quelque confusion), la seule référence de symbole affichée dans la nouvelle version du logiciel est l'index.
Parallèlement, dans les lignes de programme comportant la mention de l'index nx, celle-ci a été remplacée par le numéro d'index.
Adjonction notable : le tableau d'étiquettes; il permet de doter chaque symbole (et partant également un morceau de voie droite par exemple) d'un numéro pouvant se colorer de 5 couleurs différentes. Nous nous sommes limités à 2 chiffres (1 à 99) en raison de l'espace disponible et de la lisibilité (cf. **figure 2**).
Ce tableau permet de doter les boutons représentant des commutateurs et des boutons-poussoirs d'une couleur ou d'une autre; ces boutons peuvent en outre être dotés d'une référence numérotée.
On peut à partir de maintenant également utiliser des aiguillages anglais à double bobine de commande. L'adressage a été étendu à cet effet et l'on peut remplir 4 champs d'adresse/de donnée. Un aiguillage anglais à 2 bobines de commande ne requiert pas plus de 2 champs d'adresse/de donnée, de sorte que le programme peut déduire de l'attribution des champs de quel type de système il s'agit.
L'adressage des décodeurs a été étendu à 240, de sorte que l'on peut utiliser la totalité des 5 lignes de sélection d'adresse du MC 145027. Cette extension recouvre du même coup le domaine des 4 fonctions additionnelles d'antan, de sorte qu'il est également possible de s'en servir par le biais de boutons-poussoirs.
Notons que le décodeur de signal universel d'EDiTS permettait déjà de paramétrer ces adresses.
En cas d'utilisation de décodeurs de commutation d'autres marques cela est également possible, mais implique l'une ou l'autre modification.
Dès l'origine d'EdiTS, l'adressage des décodeurs de commutation (commutations des feux de signalisation, des éclairages publics, des passages à niveau…) comportait 5 cavaliers susceptibles de définir une adresse ternaire sur 5 trits. D'où 243 décodeurs adressables. Ceci n'était pas vrai pour les décodeurs d'aiguillages. EdiTS différenciait les 2 types de décodeurs par le trit 5 de l'adresse, à niveau haut pour les décodeurs d'aiguillage et niveau bas pour les décodeurs de commutation, le niveau haute impédance (broche en l'air) étant tout simplement inutilisé pour des raisons de compatibilité avec Märklin.
Ce que l'auteur propose en fait est de piloter indifféremment les décodeurs, aiguillages et commutation), en adjoignant un type de décodeur au symbole. D'où l'extension à 243 décodeurs en tout.

Itinéraires
Les exigences strictes de définition des tracés qui existaient à l'origine n'ont plus lieu d'être.
Cela permet de définir un itinéraire qui ne soit plus totalement fermé sur lui-même.
Cette possibilité prend toute son importance lorsque l'on se trouve en présence d'itinéraires devant se croiser. L'un des inconvénients de cette disparition de fonction d'arbitrage logiciel est qu'il n'y a plus de vérification pour s'assurer qu'un aiguillage se trouve dans la bonne position. Il faudra partant faire attention.

Programmation
C'est l'écran qui a subi les modifica-

tions les plus nombreuses.
Si, dans la première version du programme cet écran servait uniquement à la définition des lignes de programme, dans cette nouvelle version, cet écran permet la saisie de données d'adresses virtuelles. Pour ce faire, on actionne un bouton de répondeur ou de détection rouge (voie occupée) par un clic souris gauche ce qui permet d'entrer l'adresse du convoi responsable du bouton activé.
Le bouton droit de la souris sert à activer les lignes de programme liées au dit bouton.
Au niveau du tableau de programme, la modification la plus visible est une colonne servant à la définition des fonctions de locomotive. Les paramètres de pilotage les plus importants que l'on y trouve sont la vitesse, le retournement et 5 autres fonctions. L'une des possibilités de choix pratiques que l'on découvre est celle qui permet de reprendre, après un arrêt par exemple, la vitesse paramétrée par le biais d'un régleur logiciel.
Ces options peuvent être saisies dans les lignes de programme en combinaison avec une adresse de locomotive. Les numéros de niveau continuent de déterminer la priorité de traitement des alternatives.
Si l'on veut qu'une fonction soit exécutée immédiatement après l'activation d'un bouton de répondeur ou de détection, la ligne de programme correspondante ne doit pas comporter de temporisation ou d'itinéraire.
Si la ligne de programme comporte la mention d'un itinéraire, cette fonction ne sera activée qu'une fois l'itinéraire ouvert.
Si, auparavant, le niveau ayant la priorité la plus élevée était le niveau 1, la nouvelle version s'est vue dotée d'un niveau 0. À ce niveau 0, il est permis de définir uniquement des adresses de locomotive avec une fonction locomotive.
La combinaison de l'adresse locomotive ALL avec le niveau de vitesse 0 (cf. **figure 3**) possède, au niveau 0, une puissance redoutable. Il est possible ainsi, par le biais de cette ligne de programme, de faire s'arrêter tous les trains se trouvant en présence d'un signal non-sécurisé sans section hors tension.
L'adresse 79 étant l'adresse de programmation des super-décodeurs de locomotive, cette adresse ne peut pas être choisie.
Comme nous le disions dans le paragraphe des généralités, cette technique de programmation offre un maximum de possibilités pour un nombre de lignes de programmation minimum.

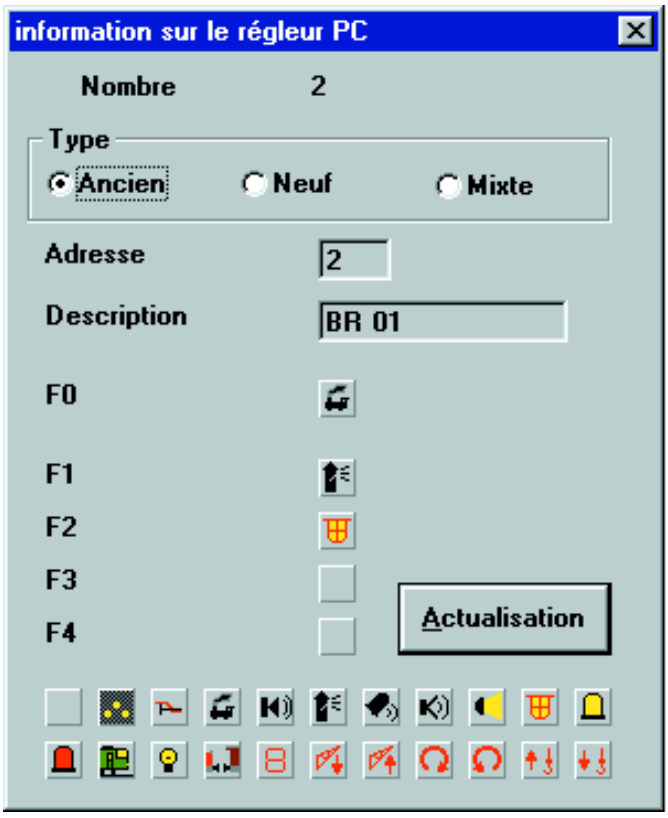


Figure 4. Il existe maintenant 3 types de régleurs; le domaine d'adressage est en outre passé à 255.

Régleurs logiciels
Ce point du menu (**figure 4**) s'est lui aussi étoffé de quelques extensions. Nous avons commencé par faire passer à 255 la plage d'adressage disponible, le codage des adresses étant conforme à celui d'Uhlenbrock (nous arriverons peut-être, qui sait, un jour à un standard unique).
Si la version originale du logiciel ne connaissait que 2 types (Normal et Etendu), la nouvelle en possède 3. La version ajoutée s'appelle Mixte (mixed). Sur la version normale (Ancien = old) il s'agit du pilotage des décodeurs de locomotive en respect de l'ancien format.
Il est possible maintenant, en vue de pouvoir piloter les fonctions même avec l'ancien format, de définir ces fonctions. Comme les décodeurs de locomotive ancien modèle n'ont pas de sorties de fonction, on pourra utiliser à cette fin le décodeur de fonction de Conrad voire le décodeur de voiture. La commande correspond aux touches de fonction sur l'ancienne unité de commande 80.
La version étendue (Neuf = new) est prévue pour la commande des décodeurs Motorola de type 2 (Motorola-2, notons au passage que Motorola n'a rien à voir ici), tant au niveau de la vitesse que des fonctions.
Le format mixte doit sa naissance au fait qu'il semble qu'il existe des trains dotés d'un décodeur Motorola-1 pour la commande du moteur et d'un décodeur Motorola-2 pour les fonctions additionnelles (le Märklin ICE 3). Cette option de choix peut s'avérer intéressante lorsque l'on envisage, par exemple, de combiner un Delta ancien format avec un décodeur de fonction Motorola-2 de type FD3 (Conrad).

Opérationnel
Offrir une possibilité d'adresser jusqu'à 255 trains n'a d'intérêt que s'il est effectivement possible de faire circuler 255 trains (!); cela implique partant qu'il faut faire passer le nombre de régleurs de 80 à 255.
Il ne nous a pas été possible d'arriver à ce nombre fatidique de 255, mais le nombre de 240 nous paraît plus que respectable. Il nous a fallu, pour donner à chacun des boutons requis, pas moins de 12, l'espace vital nécessaire, tout en faisant en sorte que le tout

puisse être visualisé sur un écran VGA (640 x 480), jouer des coudes. Le résultat final n'en est pas moins resté bien ordonné.
D'un point de vue opérationnel, il n'y a pas eu d'autres modifications visibles, cependant, mettant à profit l'adressage virtuel, les numéros de train apparaissent sur les boutons de détection et sont transmis en fonction des mouvements des convois.
Les itinéraires constituent le facteur important pour le suivi de l'adresse. Si l'on dirige un train de A vers B en libérant les sections de trajets de A vers B (manuellement ou par la saisie de lignes de programme), le programme sait, lors de l'activation du train par le biais du bouton B, qu'il s'agit du train en provenance de A ce qui permet de faire passer en conséquence l'adresse de A vers B.
Il en va tout autrement si l'on n'ouvre pas d'itinéraire et que le train se déplace de A vers B. On ne pourra dans ce cas-là avoir de transfert d'adresse correct qu'à condition de s'assurer, lors de l'activation de B, qu'il s'agit d'un train en provenance de A. Cela est le cas pour EDiTS Pro lorsqu'il n'y a qu'un itinéraire de défini vers B, à savoir de A vers B. Dans la pratique cela pourra être le cas, par exemple, sur une section de voie qui n'est toujours parcourue que dans un sens et ne disposant pas d'embranchement, un bloc automatique par exemple.

Régleurs manuels
De par l'extension au niveau du domaine d'adresses, les régleurs manuels peuvent eux aussi être paramétrés sur la plage d'adresses complète allant de 1 à 255.

Programmation du super-décodeur de locomotive
La programmation du super-décodeur de locomotive en version DIL n'a pas été modifiée, à ceci près qu'il est possible maintenant de programmer une adresse sur l'ensemble de la plage d'adresses étendue.
Une caractéristique du type fourni jusqu'à présent élimine cependant un certain nombre d'adresses (la version la plus récente permet une programmation de toutes les adresses).
Les adresses utilisables sont les suivantes :
83
123
145 à 190
193 à 255
Ceci signifie qu'il y a plus de 180 adresses disponibles.

Programmation
du super-décodeur de locomotive N
Ce décodeur disponible uniquement par le biais du site Internet EDiTS Pro, se laisse lui programmer sur l'ensemble de la plage d'adresses sachant en outre qu'il est même possible, avec ce type de décodeur, de programmer des niveaux de vitesse individualisés.

Programmation
de décodeurs Uhlenbrock
La programmation du décodeur Uhlenbrock (Motorola) connaît 3 étapes :

1. Basculement du décodeur du mode opération vers le mode programmation.
2. Programmation des caractéristiques du décodeur.
3. Fin de l'étape de programmation et retour au mode opération.

On choisit, dans la fenêtre de programme Uhlenbrock, l'adresse du décodeur (un décodeur sortant d'usine est paramétré à une adresse à 1) et on actionne le bouton « Start ».
Après une brève période de clignotement de l'éclairage avant le décodeur bascule en mode programmation, les boutons « Adresse Loco », « Vitesse Max (et Vitesse Mini dans le sens –), « Taux d'accélération/décélération » sont débloqués.
Si l'on veut modifier l'adresse il suffit d'entrer la nouvelle adresse et d'activer le bouton « Programmer » se trouvant à côté du champ de saisie de l'adresse.
En ce qui concerne le taux d'accélération/de décélération il suffit de glisser le curseur dans la position requise et d'activer le bouton « Programmer » situé à côté.

Les choses se passent un peu différemment en ce qui concerne le paramétrage de la Vitesse Max (ou Vitesse Mini). On commence par activer le bouton « Programmer » présent sur la droite de cette fonction. Au bout de quelques secondes le bouton correspondant passe au vert, instant à partir duquel il est possible de paramétrer la locomotive à l'aide du curseur. Une fois que l'on a défini la vitesse Max ou Mini, il reste à actionner une nouvelle fois le bouton « Programmer » pour mémoriser ce paramètre dans le décodeur.
Une fois que l'on a, de cette manière, entré tous les paramètres, il reste à quitter le mode programmation par une action sur le bouton « programme ».

Programmation de décodeurs Lenz
Les décodeurs Lenz connaissent en fait eux aussi les étapes indiquées dans le paragraphe précédent, à ceci près que le basculement du mode opération vers le mode programmation et inversement se fait par le biais d'un bouton-poussoir présent sur le décodeur lui-même.
Une fois que le décodeur se trouve en mode programmation (clignotement de l'éclairage avant), il est possible de programmer le décodeur par le biais de l'écran de programmation qui lui est spécifique.
On peut saisir les paramètres d'adresse, de vitesse et de taux d'accélération/décélération par le biais du champ de saisie réservé au premier et du positionnement de leurs curseurs respectifs pour les suivants.
Une action sur le bouton « Programmer » correspondant lance le transfert des valeurs vers le décodeur.

Exit

Si, avec l'ancienne version, le fait de sortie en oubliant de sauvegarder le paramétrage actuel n'était pas une trop grande catastrophe, les choses changent avec cette nouvelle version. En raison de la nouvelle fonction de suivi d'adresse, il est on ne peut plus ennuyeux d'avoir, lors du démarrage du système, à ressaisir les adresses virtuelles. Ceci explique d'EDiTS Pro demande explicitement, lors de la sortie de l'application, s'il faut « Sauvegarder » la situation du moment.
Dans un prochain article nous nous pencherons sur les nouveautés du logiciel de l'unité de commande et présenterons la platine de saisie d'adresse.
NdlR : Merci Mr Dezaire pour la relecture.

(015112)

Littérature

[1] EDiTS Pro,
Pilotage par ordinateur de modèle réduit ferroviaire,
Publitronic,
ISBN 2-86661-124-1

Kits de programmateurs

Pour les processeurs d'Atmel et de Microchip

Harry Baggen

Quiconque veut s'essayer aux générations de processeurs modernes a inévitablement besoin d'un programmateur capable de programmer ce type de composants. La société Kitrus propose un certain nombre de kit de programmateurs au prix abordable permettant une programmation sans complexe des processeurs des familles PIC et Atmel.

Les nouvelles générations de petits microcontrôleurs se caractérisent par leur programmation simple et leur mise en oeuvre confortable, la plupart d'entre eux étant même reprogrammables de par la présence de mémoire de type Flash. De plus en plus nombreux sont les amateurs (éclairés) et les professionnels qui utilisent partant ce type de contrôleurs dans leurs projets.

La programmation de ce genre de contrôleurs requiert une électronique spéciale qui, heureusement, est relativement simple dans la plupart des cas. Les amateurs endurcis seront plutôt tentés de trouver un schéma utilisable sur Internet, voire de feuilleter quelques numéros d'Elektor pour essayer d'y trouver leur bonheur, un programmateur utilisable pour le type de contrôleur concerné. Nous pouvons fort bien comprendre qu'un électronicien débutant puisse trouver intéressant l'existence de kits complets sur le marché, kits comportant toutes les pièces et composants requis. La société Kitsrus propose via le site Internet de son distributeur aux Pays-Bas (www.arexx.nl) un certain nombre de kits destinés à la programmation d'un plusieurs types de microcontrôleurs spécifiques.

La dénomination du kit **K81** est « *Introduction to the PIC16F84* ». Ce kit se compose d'un programmateur simple et d'un montage de test de ce microcontrôleur ô combien populaire (une LED clignotante). Le programmateur est relié au PC par le biais de son interface Centronics. On trouve, sur la disquette accompagnant ce kit un assembleur/programme de programmation pour F84, le site Internet de Kitsrus proposant une nouvelle version tournant sous Windows (*PICALLW*). Le montage ne comporte que peu de composants et est facile à réaliser même pour des doigts inexpérimentés. Le kit comprend bien entendu un PIC « pour se faire la main ».

Le kit **K96** (*P16PRO Serial PIC Programmer*) est lui aussi un programmateur de PIC piloté par le biais du port Centronics. Cette variante comporte plusieurs supports pour circuit intégré de taille différente ce qui permet la programmation de tous les PIC sériels à 8, 18, 28 et 40 broches. À nouveau, le nombre de composants nécessaires est faible et la réalisation sans problème.

Le kit le plus complet, le **K123** (*Atmel 89xxxx Programmer*), concerne un programmateur pour un certain nombre de microcontrôleurs de l'écurie Atmel tels que les 89C1051, 89C51, -52, -55, 89S8252, 89S53 et 87F51. Le programmateur dispose de son propre progiciel stocké dans un 89C51 préprogrammé. La liaison avec le PC se fait par voie sérielle (RS-232). Tout ce dont on a besoin au niveau du PC est un programme de terminal tout ce qu'il y a de plus simple de manière à pouvoir communiquer avec le programmateur par le biais de quelques instructions. Il n'est pas nécessaire de disposer de quelque autre logiciel que ce soit.

Tous les programmateur disposent d'un jack d'alimentation basse-tension auquel il faudra connecter un adaptateur secteur fournissant une tension continue de 16 V au minimum.

Le mode d'emploi de ces différents kits est, d'origine, en langue anglaise, mais l'importateur nous a assuré que les kits destinés à des clients français seraient accompagnés d'une notice traduite dans la langue de Molière. Le petit manuel en question ne décrit que la réalisation du programmateur en question. Il faudra, si l'on a besoin d'une documentation technique plus complète pour comprendre le fonctionnement d'un microcontrôleur ou d'un autre, faire un tour sur le site Internet du fabricant du type de microcontrôleur concerné.

(010080)

Les kits de Kitsrus peuvent être commandés directement sur le site Internet de l'importateur néerlandais :
www.arexx.nl
Tél. : (+31) 38 454 2028
Fax. : (+31) 38 452 4482

Adresses Internet :

Kitsrus :	www.kitsrus.com
Microchip :	www.microchip.com
Atmel :	www.atmel.com

Émetteur/récepteur IR pour PC (2)

Émetteur RC5 et liaison de données

par Burkhard Kainka

Le circuit de l'émetteur-récepteur IR a déjà été introduit dans le numéro précédent accompagné d'exemples de fonctionnement comme récepteur RC5 sur PC. La deuxième et dernière partie de cette contribution est dévolue au fonctionnement de l'émetteur : outre la télécommande par PC, la transmission par infrarouge de PC à PC sera introduites par des exemples compréhensibles et faciles à suivre.

La génération de signaux de télécommande peut s'avérer tout aussi utile que leur réception, par exemple pour la commande automatique d'un magnétoscope, la mise un marche d'un caméscope ou le passage d'un programme de télévision à un autre à une heure donnée. Il est possible en principe d'exécuter automatiquement n'importe quelle fonction d'une télécommande IR RC5 à partir d'un PC.

Émetteur RC5

Les télécommandes usuelles fonctionnent avec des impulsions de courte durée (moins de 10 % du temps de commutation) mais dont le courant peut atteindre 1 A. Le signal émis reste suffisamment puissant pour que le récepteur puisse le détecter lorsqu'il ne le reçoit que par l'entremise d'une réflexion sur une paroi blanche. L'interface sérielle n'est toutefois pas en mesure d'engendrer un courant aussi élevé. Mais la tension disponible suffit à faire fonctionner plusieurs LED IR en série. L'émetteur-récepteur IR fait appel à 2 diodes d'émission par lesquelles passe un courant impulsionnel de 200 à 300 mA ; les impulsions lumineuses courtes et puissantes ainsi engendrées ont une portée de plusieurs mètres.

Le premier programme d'émission (**listage 1**) fait appel à la procédure RC5tx pour engendrer l'impulsion de commande sur la ligne TXD. Les 3 trains de données Ctr, Adr et Dat sont passés lors de l'appel. L'utilisateur peut indiquer l'adresse de l'appareil et le code de la touche dans la fenêtre de texte. Le code de commande est répété 5 fois, de sorte que l'envoi des données dure 0,5 secondes.

La procédureRC5tx fonctionne en mode REALTIME. Elle engendre tout d'abord la séquence initiale avant de traiter les données de commande qui lui ont été passées. Chaque bit est décalé en position 0 au moyen d'une division par 2 pour le sélectionner. Ce bit commande la procédure RC5bit.

La fonction DLL REALTIME de Port.DLL augmente la priorité du programme qui l'appelle. Une procédure cruciale ne peut donc pas être interrompue par d'autres processus Windows. Ce procédé réduit le problème du manque de capacité temps réel d'un programme Windows. Les interruptions dues, par exemple, aux événements liés à la souris sont supprimées pour la durée d'un paquet RC5. Mais il faut ajouter la ligne « REALTIME false » à la fin de la procédure pour revenir à l'état normal.

L'émetteur RC5 peut servir de commande automatique. Il faut connaître pour cela l'adresse de l'appareil (voir le **tableau 1**) et le code de la touche (voir le **tableau 2**). L'émetteur peut alors engendrer par exemple la commande idoine à une heure donnée ou à la suite de tout autre événement.

Note de la rédaction :
la longueur du listage 5 nous empêche de l'inclure dans cet article. Il est disponible, ainsi que les autres d'ailleurs, gratuitement au téléchargement sur le site Internet d'Elektor *www.elelektor.presse.fr* avec les fichiers des programmes et se trouve dans la liste de téléchargement sous le numéro 010052-2-11 ainsi que sur le site de l'auteur (adresse en fin d'article). Les lecteurs ne disposant pas d'un accès à Internet peuvent aussi les commander par courrier, télécopie ou téléphone auprès de la rédaction.

Un exemple :
un magnétoscope doit commencer puis cesser d'enregistrer à des heures données. Le programme VCRtimer (**listage 2**) permet de spécifier les heures de commutation. Deux boutons sont aussi disponibles pour la commande manuelle.
Un autre exemple de programmation vient au secours du « doigt zappeur ». Ce mal redouté frappe de nombreux téléspectateurs dépassés par la multitude de programmes, tout comme le coude du tennisman atteint les fanas du sport et le syndrome de la souris les accros de l'ordinateur. L'émetteur-récepteur RC5 accompagné du petit programme du **listage 3** viendra à leur secours.

Transmission sérielle des données

On s'est limité jusqu'ici aux signaux infrarouges conformes à la norme RC5. Dans ce qui va suivre, il sera possible de transmettre des données sérielles entre PC par lumière infrarouge modulée. On peut se servir par exemple de l'émetteur-récepteur IR pour bavarder dans une salle informatique en échangeant des messages texte. Quelques essais préliminaires permettront de mieux cerner le processus d'envoi d'un signal sériel.

Figure 1. Programme de test pour signaux RC5.

Listage 1

Le programme de l'émetteur RC5 (RC5TxTest.vbp)

```
Dim Control
Private Sub Form_Load()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then
    i = OPENCOM("COM1,1200,N,8,1")
    Option1.Value = True
 End If
 If i = 0 Then MsgBox ("COM Interface Error")
 RTS 1
 DTR 1
 Control = 1
 Chan = 0
End Sub
Private Sub Form_Unload(Cancel As Integer)
  CLOSECOM
End Sub
Private Sub Option1_Click()
 i = OPENCOM("COM1,1200,N,8,1")
 If i = 0 Then MsgBox ("COM1 not available")
 RTS 1
 DTR 1
End Sub

Private Sub Option2_Click()
 i = OPENCOM("COM2,1200,N,8,1")
 If i = 0 Then MsgBox ("COM2 not available")
 RTS 1
 DTR 1
End Sub

Private Sub TxBit(ONOff!)
  If ONOff = 1 Then
    TXD 0
    DELAYUS 888
    TXD 1
    DELAYUS 888
  Else:
    TXD 1
    DELAYUS 888
    TXD 0
    DELAYUS 888
  End If
End Sub

Private Sub RC5tx(Ctr, Adr, Dat)
  BitTime = 888
  REALTIME True
  TIMEINITUS
  TXD 1
  While TIMEREADUS < (BitTime * 1)
  Wend
  TXD 0
  While TIMEREADUS < (BitTime * 2)
  Wend
  TXD 1
  While TIMEREADUS < (BitTime * 3)
  Wend
  TxBit (Ctr)
  BitVal = 16
  For n = 1 To 5
    b = Int((Adr And BitVal) / BitVal)
    TxBit (b)
    BitVal = BitVal / 2
  Next n
  BitVal = 32
  For n = 1 To 6
    b = Int((Dat And BitVal) / BitVal)
    TxBit (b)
    BitVal = BitVal / 2
  Next n
  TXD 0
  REALTIME False
End Sub

Private Sub Chan1()
  For n = 1 To 3
    RC5tx Control, 5, 1
    DELAY 100
  Next n
  If Control = 1 Then
    Control = 0
  Else
    Control = 1
  End If
  Chan = 1
End Sub

Private Sub Command1_Click()
  Adr = Val(Text1.Text)
  Dat = Val(Text2.Text)
  For n = 1 To 5
    RC5tx Control, Adr, Dat
    DELAY 100
  Next n
  If Control = 1 Then
    Control = 0
  Else
    Control = 1
  End If
End Sub
```

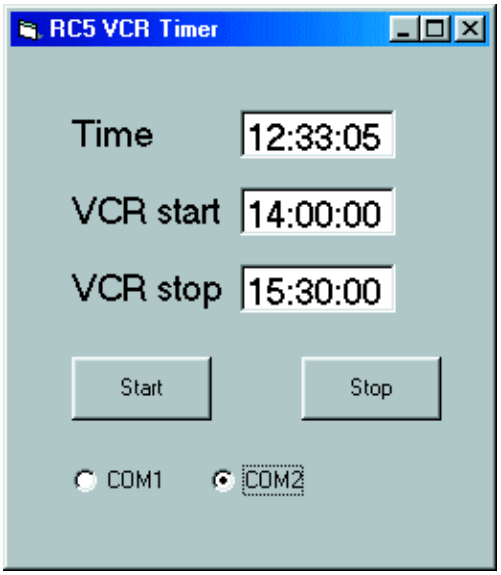


Figure 2. Minuterie vidéo en fonctionnement.

Dans la première partie de cette contribution, l'interface sérielle a été utilisée d'une façon complètement différente de celle prévue par ses réalisateurs.
Cette interface a été initialement conçue pour transmettre des données par l'entremise de modems. Des expressions comme « Ring Indicator » ou « Data Terminal Ready » en témoignent. Au fil du temps, l'interface s'est aussi avérée de plus en plus utile dans d'autres domaines. Le raccordement de la souris à l'interface RS-232 a constitué une percée particulièrement significative. Lors de l'introduction de l'interface RS-232, on était loin de s'imaginer que des appareils possédant leur propre microcontrôleur fonctionneraient un jour avec cette interface qui leur fournirait même le courant de fonctionnement. Mais bien des années plus tard, ce point a été à la base du concept de la nouvelle interface sérielle USB.

Figure 3. Programme de zapping automatique.

Mais revenons au bon vieux port RS-232 qui a été développé pour transférer des données par 2 lignes (TXD et RXD). Pour ce faire, les données sont convertis en un flux sériel de bits par le matériel de l'interface et envoyées vers leur destination. Le deuxième interface, celle du récepteur, reconstruit à partir du flux sériel les octets de données qui peuvent être traités par le PC. Le tout est un bel exemple de division du travail entre logiciel et matériel. L'envoi d'un caractère par l'interface sérielle nécessite un temps considérable par rapport à d'autres opérations. Alors qu'on se bat d'ordinaire pour des microsecondes, c'est à des millisecondes qu'on a affaire ici. C'est pourquoi l'interface du PC est équipée d'un composant matériel spécial, l'émetteur-récepteur asynchrone universel (*Universal Serial Receiver Transmitter*, UART), une sorte de station télégraphique pour PC. Les informations y sont remises pour être envoyées de façon autonome. Inversement, l'UART reçoit les informations de sa propre initiative et les place de façon à ce qu'elles puissent être retirées en une fois par un programme.
Une fonction typique de l'interface sérielle consiste à transférer des données d'un PC à un autre. Les données peuvent être de n'importe quel type : texte, programmes, images, etc. En fin de compte, les données sont envoyées par octets, donc par morceaux de 8 bits. Mais il existe aussi des réglages de l'interface pour lesquels un caractère ne possède que 7 bits ou même 5. Les caractères 5 bits sont un héritage du temps des téléscripteurs mécaniques qui envoyaient ces caractères sur la ligne de façon sérielle exactement selon le même principe qu'une interface RS-232, mais beaucoup plus lentement et plus bruyamment.

Liaison par faux modem

Un premier essai permettra à 2 PC reliés d'échanger des caractères. Il faut pour cela croiser les lignes TXD et RXD. On désigne ce type de liaison par « câble de faux modem » car aucun modem n'est utilisé. Alors que les liaisons à grande distance doivent être effectuées par lignes téléphoniques équipées de modems, un câble normal (**figure 4a**) suffit si la distance ne dépasse pas quelques mètres. On peut aussi tenter l'expérience avec un seul PC qui s'envoie des données par l'entremise de sa propre interface sérielle (**figure 4b**).
La transmission sérielle des données est assurée par la procédure SENDBYTE et la fonction READBYTE du fichier PORT.DLL déjà mentionné dans l'article précédent. Pour que l'échange de données s'effectue correctement, il faut aussi que les interfaces des 2 partenaires soient initialisées avec les mêmes paramètres. OPENCOM ("COM2,1200,N,8,1") définit une vitesse commune de 1 200 bits par seconde (1 200 bauds), pas de bit de parité (bit de test), 8 bits de données et 1 bit d'arrêt. Nous reviendrons sur la signification de ces paramètres.
La **figure 5** reproduit l'oscillogramme du caractère « 1 » envoyé en série à 1 200 bauds. Il faut à 1 bit 1s/1 200 = 833 μs lorsque la vitesse de transmission est de 1 200 bauds. On reconnaît à gauche de l'image le bit

Tableau 1
Adresses des appareils

Adresse de l'appareil	Appareil
0	Téléviseur
2	Télétexte
5	Magnétoscope
7	Expérimental
16	Préamplificateur
17	Récepteur/Tuner
18	Magnéto-cassette
19	Expérimental

Tableau 2
Quelques codes de touches pour TV et magnétos importants

Code	Funktion
0...9	Touches des chiffres 0 à 9
12	Standby
13	Silencieux
16	Volume +
17	Volume –
32	Programme +
33	Programme –
48	Pause Vidéo
53	Lecture Vidéo, Son TV en fonction
54	Arrêt Vidéo
55	Enregistrement Vidéo

Listage 2

Les procédures du temporisateur Vidéo les plus importantes (VCRtimer.vbp)

```
Private Sub RecordOn()
  For n = 1 To 10
    RC5tx Control, 5, 55
    DELAY 100
  Next n
  If Control = 1 Then
    Control = 0
  Else
    Control = 1
  End If
  ONOff = 1
End Sub

rivate Sub RecordOff()
  For n = 1 To 10
    RC5tx Control, 5, 54
    DELAY 100
  Next n
  If Control = 1 Then
    Control = 0
  Else
    Control = 1
  End If
  ONOff = 0
End Sub

Private Sub Timer1_Timer()
  Text1.Text = Time$
  If (Time$ > Text2.Text) And (Time$ < Text3.Text) And (ONOff
= 0) Then RecordOn
  If (Time$ > Text3.Text) And (ONOff = 1) Then RecordOff
End Sub

Private Sub Command1_Click()
  RecordOn
End Sub

Private Sub Command2_Click()
  RecordOff
End Sub
```

de départ qui dure 0,833 ms. Il est suivi de 8 segments dont chacun dure aussi 0,833 ms. Seul le premier bit suivant le bit de départ se distingue du flux de données par son niveau nul. Comme il s'agit d'envoyer le chiffre « 1 », on voit que les bits de poids faible sont transmis en premier et que le niveau des bits de données envoyés est inversé. On a donc 7 états de niveau haut durant eux aussi 0,833 ms. Ce sont en effet les 7 bits nuls suivants. L'octet reconstitué sera donc 00000001 = 1. Vient ensuite, même si on ne le voit pas, un bit d'arrêt de niveau nul. Le bit d'arrêt n'est rien d'autre que l'arrêt le plus court possible avant le caractère suivant, s'il y en a un. Sinon, l'arrêt peut durer aussi longtemps qu'il est nécessaire.
La suite des opérations dans l'UART lors de la réception d'un caractère sériel est basée sur la structure que nous venons d'examiner. Le composant doit continuellement surveiller sa ligne d'entrée dans l'attente d'un flanc montant. Il s'agit du début d'un bit de départ. L'UART met alors en marche une horloge interne pour attendre la fin du bit de départ. Il s'y

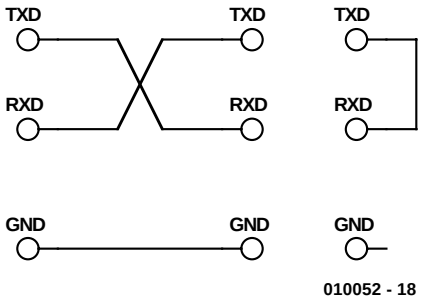


Figure 4. Liaison par faux modem (a) et test avec une seule interface (b).

ajoute un délai d'un demi-bit avant que le niveau soit interrogé. Voilà donc le premier bit lu. Chaque bit suivant est lu après un délai d'un bit. Le processus de lecture se termine donc théoriquement un demi-bit avant le début du bit d'arrêt.
On voit que la synchronisation par rapport à la trame de données n'a lieu qu'une fois, au début du bit de départ. Le reste du bloc de temps doit être connu exactement et ne pas subir de déviations importantes. Des erreurs de lecture sont inévitables si le nombre de bauds d'un émetteur diffère de celui du récepteur. Les UART modernes ne se content d'ailleurs pas d'une seule valeur de chaque bit de données, mais lisent par exemple 3 états vers le milieu du bit qui doivent normalement être égaux. Si la présence de parasites a conduit à des valeurs différentes, une décision majoritaire est utilisée, ce qui améliore la sécurité des données. Le premier exemple de programmation (**listage 4**, voir l'annexe en fin d'article) d'une transmission de données ne transfert qu'un octet par commande. Il s'agit en l'occurrence de la position d'un curseur de défilement dont la valeur entre 0 et 255 est transmise à l'autre ordinateur. Ce dernier affiche chaque octet reçu dans une fenêtre de texte (**figure 6**).

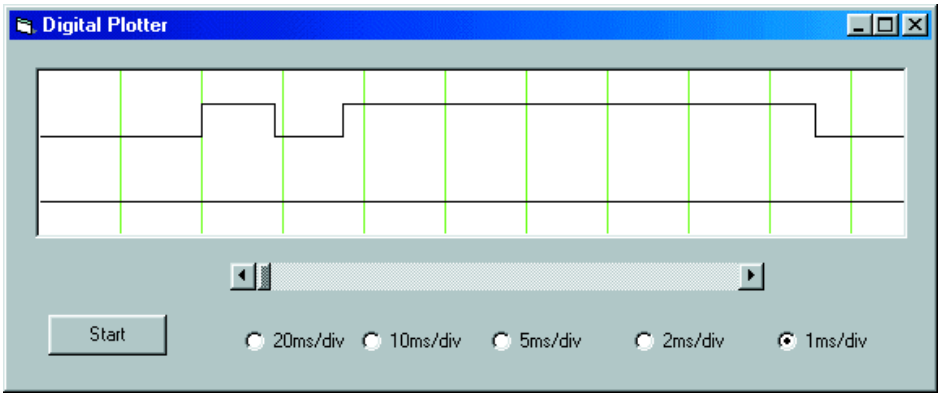


Figure 5. Envoi de l'octet « 1 » à 1 200 bauds.

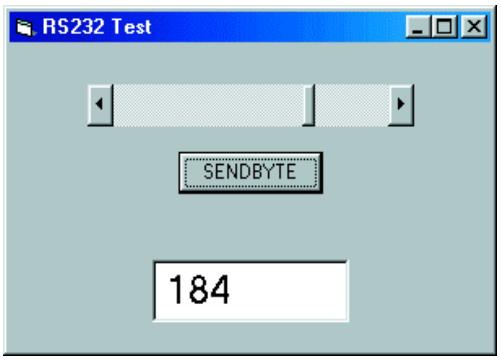


Figure 6. Émission et réception d'octets individuels.

Un autre listage (**listage 5**) et une description dans le fichier de téléchargement vous offrent un programme d'émulation de terminal qui se prête particulièrement bien au développement et aux essais.

Transmission de données par signaux lumineux

Pour effectuer une expérience simple de transmission optique de signaux, il suffit d'une diode d'émission IR et d'un phototransistor (voir la **figure 7**).
Alors que la LED d'émission est attaquée directement par TXD, le récepteur a besoin d'une tension auxiliaire que l'on peut trouver par exemple sur la ligne DTR positionnée à l'état haut. Le courant passe et TXD est positionné à l'état haut dès que la lumière de la diode d'émission atteint le phototransistor. On obtient ainsi une copie conforme du signal de TXD également sur RXD.
Cet essai peut être effectué par exemple avec le programme d'émulation de terminal du fichier de téléchargement. Les données peuvent être échangées entre 2 PC, mais l'essai peut tout aussi bien avoir lieu sur un seul PC en y réinjectant les données comme le montre la figure 4b. Il faut seulement se rappeler d'activer la ligne DTR du récepteur.

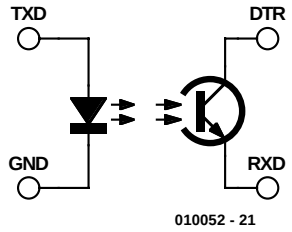


Figure 7. Le trajet de transmission lumineuse le plus simple ne comporte qu'une LED et un phototransistor.

Listage 3

Changement de programme automatique (Zapper.vbp)

```
Private Sub Chan1()
  For n = 1 To 3
    RC5tx Control, 5, 1
    DELAY 100
  Next n
  If Control = 1 Then
    Control = 0
  Else
    Control = 1
  End If
  Chan = 1
End Sub
Private Sub ChanPlus()
  For n = 1 To 3
    RC5tx Control, 5, 32
    DELAY 100
  Next n
  If Control = 1 Then
    Control = 0
  Else
    Control = 1
  End If
  Chan = Chan + 1
End Sub

Private Sub Timer1_Timer()
  If Chan = 30 Then Chan1
  If (Chan > 0) And (Chan < 30) Then ChanPlus
  If Chan = 0 Then Chan1
  Text1.Text = "Chan " + Str$(Chan)
End Sub

Private Sub Command1_Click()
 Timer1.Enabled = True
End Sub

Private Sub Command2_Click()
 Timer1.Enabled = False
End Sub
```

La portée de la transmission est limitée à quelques millimètres, mais l'emploi de fibres optiques la fait déjà passer à quelques mètres. Il existe des émetteurs et des récepteurs de lumière qu'il est facile de monter avec des câbles à fibres optiques. La diode d'émission SFH250 et le phototransistor SFH350 possèdent les perforations nécessaires au montage de câbles à fibres optiques. Ce trajet par fibres optiques permet de transférer des données à n'importe quel débit jusqu'à 115 200 bauds.
Pour augmenter la porté dans l'espace, on peut avoir recours à des signaux modulés comme ceux qui sont utilisés dans les télécommandes IR. On peut utiliser l'émetteur-récepteur IR (partie 1 de cet article) qui inverse le signal de réception par un étage à transistor pour que 0 corresponde à l'état de repos. En fait, les récepteurs tels que SFH506 ou TSOP1836 conçus pour les télécommandes IR ne se prêtent à la transmission de signaux RS-232 qu'à condition de tenir compte de certaines particularités. En effet, la portée lors de la réception de signaux de télécommande doit être aussi grande que possible, même lorsque la lumière ambiante est forte. Cela nécessite un filtre de bande et un réglage automatique du gain (AGC, voir la **figure 8**) comme dans les récepteurs hertziens. Ce circuit de réglage possède toujours une constante de temps qui doit être optimisée pour le but poursuivi.
Comme cette constante de temps est dimensionnée pour les signaux de télécommande, il faut éviter que les impulsions émises soient trop courtes ou trop longues. Ce type de circuit ne se prête pas au traitement d'un niveau constant car il provo-

Téléchargement :
Vous trouverez sur le site Internet d'Elektor (www.elektor.presse.fr), sous le numéro 010052-2-11 de la liste de téléchargement de ce fascicule, un fichier zip prêt à être téléchargé. Après l'avoir dézippé, on obtient un dossier de tous les fichiers de programmes et un fichier contenant les listages des programmes et la description du programme d'émulation de terminal.
Adresse du site de l'auteur : *http://home.t-online.de/ home/B.Kainka*

Pour en savoir plus :
Le livre « Petites expériences d'électronique avec mon PC », Publitronic (ISBN 2-86661-126-8) de Burkhard Kainka paru vers le milieu de l'année contient un grand nombre d'autres applications, d'un analyseur logique numérique à des oscilloscopes simples, en passant par la génération de sons au moyen de l'interface sérielle. Ce livre donne aussi un aperçu du fonctionnement interne d'une DLL commandant une interface RS-232. Le lecteur se trouve confronté à des faits nouveaux sur les appels d'API Win32, leur utilisation directe avec Visual Basic et la réalisation de sa propre DLL avec Delphi. La DLL, réalisée spécialement pour ce livre, permet entre autre d'utiliser simultanément plusieurs interfaces COM dans un programme

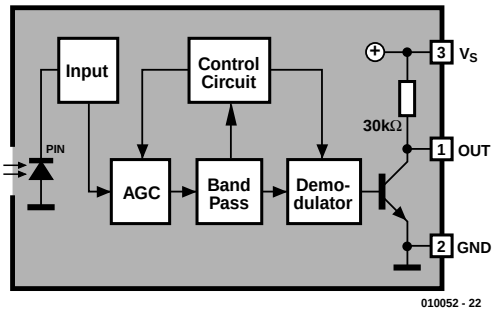


Figure 8. Schéma fonctionnel d'un récepteur IR TSOP18xx.

querait une baisse trop sensible du gain par le réglage automatique.
La fiche de données mentionne les restrictions suivantes :
une salve de signaux (*Burst*) doit comporter au moins 6 impulsions ; elle doit donc durer au moins 167 µs à 36 kHz.
il faut un délai d'au moins 9 impulsions (250 µs) après chaque burst.
il faut un délai d'au moins 15 ms toutes les 90 ms.
Si l'on veut que ces conditions, et tout particulièrement la dernière, soient remplies, il faut garantir le délai de 15 ms par les bits d'arrêt d'un signal sériel. Le cas le plus défavorable est celui de l'envoi d'une séquence continue d'octets nuls. Dans ce cas, en effet, le bit de départ et les 8 bits de données sont au niveau haut. Seul le bit d'arrêt produit un délai. Pris littéralement, cela signifie que le nombre de bauds le plus élevé d'un flux de données continu ne peut dépasser 130 bauds lorsque 2 bits d'arrêt sont utilisés. La situation s'améliore quelque peu lorsque le flux d'octets n'est pas continu mais sporadique, comme c'est par exemple le cas pour la frappe au clavier. Mais même le mélange aléatoire d'octets différents dans un flux permanent de données suffit à assurer en moyenne un nombre suffisant de pauses.
Le programme d'émulation de terminal du fichier de téléchargement permet de tester facilement différents paramètres de transmission (**figure 9**). Un test exhaustif au moyen de caractères tapés à la main révèle que la transmission s'effectue sans erreur pour tous les nombres de bauds de 50 à 2 400. La longueur d'impulsion et de pause la plus courte à 2 400 bauds est de 417 µs, ce qui est compatible avec les caractéristiques du récepteur. La distance maximale entre l'émetteur et le récepteur dépend avant tout de la diode d'émission utilisée et de son courant. Toujours est-il que la portée indiquée par la fiche de données du TSOP18xx est de 35 mètres.

(010052-2)

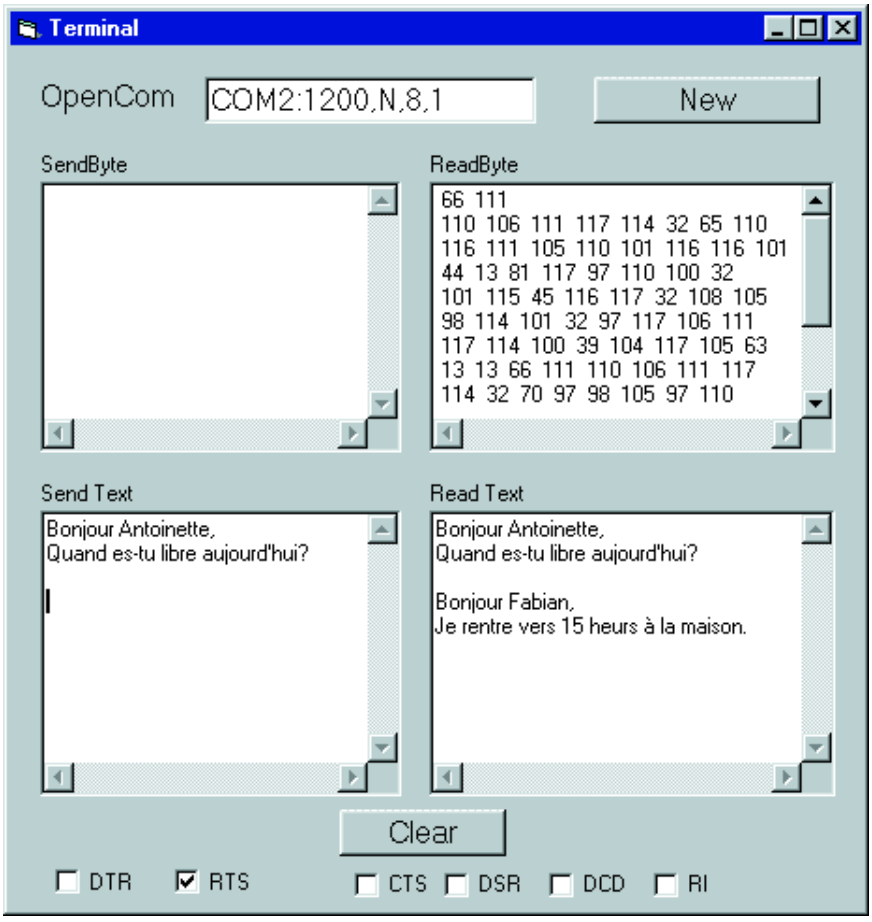


Figure 9. Salon de « clavardage » infrarouge basé sur le programme d'émulation universelle de terminaux.

Annexe

Programme de terminal universel

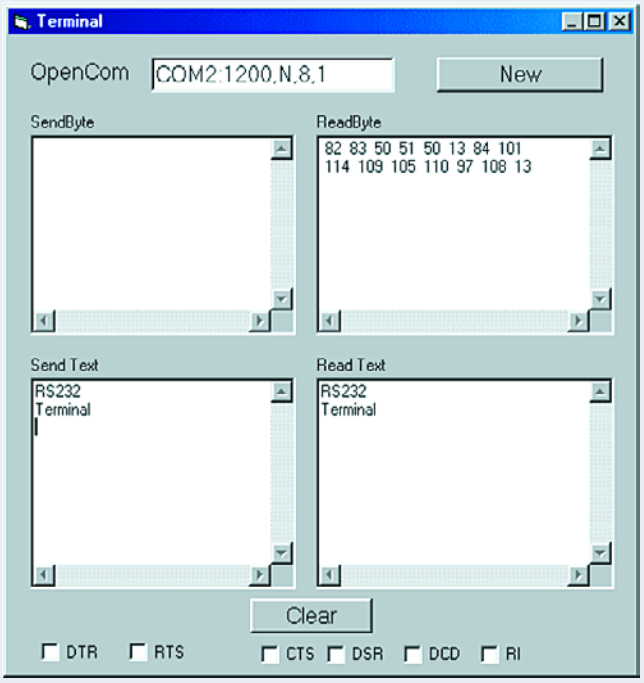


Figure 1 : transmission de texte par le biais du programme de terminal.

On a souvent besoin, lorsque l'on travaille avec l'interface RS-232, d'un programme de terminal, permettant d'une part de visualiser les données reçues et l'émission de données de l'autre. Si l'on travaille sous Windows on dispose de Hyper Terminal qui permet, par exemple, de transmettre des données par le biais d'un modem. Nous vous proposons ici un programme de terminal conçu spécialement pour les développements et qui a été écrit de façon à être le plus universel possible et servir aux applications les plus diverses. L'utilisateur peut entrer directement la chaîne de caractères (Open String) et ouvrir l'interface à l'aide des données souhaitées. Dans certains cas, en particulier lorsque l'on ne connaît pas les paramètres de transmission d'un périphérique, cette approche permet des essais rapides. On aura, après chaque modification de la chaînes de caractères et activation du bouton « New », fermeture de l'interface et sa réouverture avec le nouveau paramétrage.

Les mêmes données sérielles se laissent interpréter de 2 façons totalement différentes. Dans le premier cas il s'agit de données de commande prenant la forme d'un octet, c'est-à-dire qu'il s'agit de valeurs numériques, de nombres, pouvant aller de 0 à 255. On parle dans ce cas de format binaire. Un exemple parlant est la transmission de la position d'un potentiomètre à glissière de l'article précédent. Nombre d'appareils de mesure, d'interfaces et de périphériques tels que souris pour ordinateur et autres instruments de pointage utilisent ce mode.

L'autre mode d'interprétation des données transmises est de les considérer comme du texte. Les caractères alphanumériques prennent eux aussi la forme d'octets. La **figure 1** de cette annexe montre la transmission de caractères alphanumériques. Pour ce test, les lignes TxD et RxD ont été reliées directement de sorte que le texte entré soit reçu par la même interface. Le set de caractères ASCII définit très exactement la valeur numérique de chaque caractère. Le nombre 65 représente ainsi le A majuscule, un 49 le chiffre 0. Lorsque l'on envoie du texte on ne se soucie guère du code ASCII. Il arrive cependant qu'il soit intéressant de procéder à un examen plus critique. S'il vous faut, par exemple, interroger un appareil de mesure à interface sérielle par le biais d'un programme, le manuel ne donne souvent que des informations incomplètes. Il n'est pas rare que l'on ait mélange de données de texte et de données binaires.

Le programme de terminal du **listage 5** (proposé sur Internet) supporte d'origine les 2 interprétations de caractères. Les données entrantes sont visualisées simultanément sous la forme d'octets et de caractères alphanumériques. On voit ainsi rapidement de quoi il retourne. Lors de l'émission on a également la possibilité d'utiliser les 2 formats. Dans la fenêtre du bas on entre le texte qui sera transmis caractère après caractère.

Dans la fenêtre du haut au contraire on entre des valeurs de caractère comprises entre 0 et 255, entrée confirmée par une action sur la touche « Enter ».

Lors de l'émission de texte dans la fenêtre de saisie inférieure on a transmission des caractères de texte correspondant aux différentes touches actionnées sans la moindre modification. Visual BASIC fournit en outre la valeur du caractère dans la variable KeyAscii, qui est transmise directement à SENDBYTE. Une action sur la touche « Enter » se traduit elle aussi par un caractère, à savoir le caractère Retour à la ligne (*Line Feed*, LF de valeur numérique 10).

L'interprétation de valeurs numériques de la fenêtre de texte supérieure est un peu plus complexe. Le programme doit attendre que l'utilisateur clôture la saisie d'un nombre par une action sur la touche « Enter ». Le texte comporte de ce fait en dernière position le caractère LF 10. Les 4 derniers caractères sont alors séparés et convertis d'un texte vers une valeur numérique. Celle-ci devrait se situer entre 0 et 255. En guise de protection contre des erreurs de saisie de nombres de valeur élevée, les grandes valeurs se voient

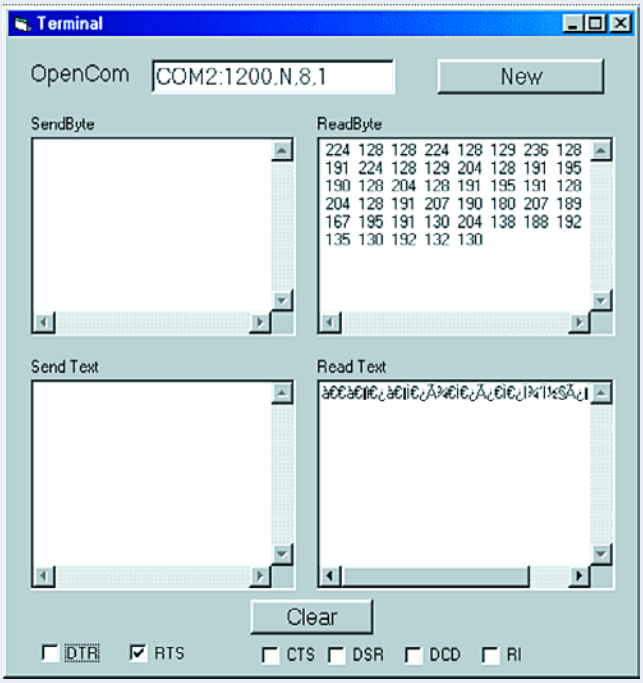


Figure 2.: Réception de données binaires en provenance d'une souris sérielle.

limitées par le biais d'une fonction logique « AND 255 ».
La réception des caractères se fait dans le cadre d'une procédure de temporisateur. On s'assure, toutes les 100 ms de l'arrivée ou non de caractères. Pour éviter tout engorgement, tous les caractères reçus sont à chaque fois lus et traités, jusqu'à ce que READBYTE fournisse un –1 comme résultat, ce qui traduit un tampon vide. Les caractères reçus sont visualisés dans les fenêtres de visualisation.
Lors de la visualisation sous format numérique dans la fenêtre de visualisation de texte, les valeurs numériques sont ordonnées en lignes de 8 octets chacune. Chaque ligne est terminée par les caractères spéciaux Retour à la ligne (LF, ASCII 10) et Retour Chariot (*Carriage Return*, CR, ASCII 13). La visualisation de texte de la fenêtre inférieure doit elle aussi traiter de façon spécifique les retours à la ligne. Le caractère LF est intercepté et remplacé par un LF + CR. Tous les autres caractères sont collés directement un texte. Le résultat de ce traitement est la visualisation d'un texte en sortie qui correspond très exactement à celui entrée dans le fenêtre de saisie.

Le programme pourra être utilisé comme outil universel pour le test de liaisons sérielles. La possibilité d'accès direct à toutes les lignes d'acquittement dans le cadre de ce même programme constitue une aide additionnelle précises. Nombre d'appareils requièrent la mise des lignes DTR ou RTS à un état donné. Ainsi, la souris PC sérielle dérive sa tension d'alimentation de la ligne TRS. Il est possible ainsi, à l'aide du programme de terminal, d'examiner le transfert de données d'une souris. Le résultat de cette opération est visualisé en **figure 2**.

Listage 4

Émission et réception d'octets individuels (Rsbyte1.vbp)

```
Private Sub Form_Load()
 OPENCOM "COM2:1200,N,8,1"
End Sub
Private Sub Form_Unload(Cancel As Integer)
  CLOSECOM
End Sub

Private Sub Command1_Click()
  d = HScroll1.Value
  SENDBYTE d
End Sub

Private Sub Timer1_Timer()
  d = READBYTE
  If d > -1 Then Text1.Text = Str$(d)
End Sub
i
```

IL300

Optocoupleur linéaire

ELEKTOR INFOCARTE I/2002

Caracteeristiques électriques (à 25 °C)

Symbole	Paramètre	Conditions	Min.	Typ.	Max.	Unité
Émetteur à LED						
V_F	Tension de seuil	I_F = 10 mA	-	1,25	1,50	V
$\Delta V_F/\Delta°C$	Coefficient de température V_F		-	-2,2	-	mV/°C
I_R	Courant inverse	V_R = 5,0 V	-	1,0	10	µA
C_J	Capacité de la couche de jonction	V_F = 0 V, f = 1 MHz	-	15	-	pF
$\Delta V_F/\Delta I_F$	Résistance dynamique	I_F = 10 mA	-	6,0	-	Ω
t_r,t_f	Temps de commutation	ΔI_F = 2,0 mA, I_{Fq} = 10 mA	-	1,0	-	µs
Détecteur						
I_D	Courant de noir	V_{det} = -15 V, I_F = 0 µA	-	1,0	25	nA
V_D	Tension à vide	I_F = 10 mA	-	500	-	mV
I_{SC}	Courant de court-circuit	I_F = 10 mA	-	70	-	µA
C_J	Capacité de la couche de jonction	V_F = 0 V, f = 1,0 MHz	-	12	-	pF
NEP	Bruit	V_{det} = 15 V	-	$4\cdot10^{14}$	-	W/√Hz
Caractéristiques de couplage						
K1	Gain de servo (I_{P1}/I_F)	I_F = 10 mA, V_{det} = 15 V	0,0050	0,007	0,011	-
I_{P1}	Courant de servo	I_F = 10 mA, V_{det} = 15 V	-	70	-	µA
K2	Gain dans le sens direct (I_{P2}/I_F)	I_F = 10 mA, V_{det} = 15 V	0,0036	0,007	0,011	-
I_{P2}	Courant direct	I_F = 10 mA, V_{det} = 15 V	-	70	-	µA
K3	Gain de transfert (K2/K1)	I_F = 10 mA, V_{det} = 15 V	0,56	1,00	1,65	K2/K1
ΔK3	Linéarité du gain en sens direct	I_F = 1...10 mA	-	±0,25	-	%
		I_F = 1...10 mA, T_A = 0...75 °C	-	±0,5	-	%
Photocouplage						
BW (-3 dB)	Réponse en fréquence	IFq = 10 mA, MOD = ±4 mA, $RL_>$ = 50 Ω	-	200	-	kHz
-	Comportement de phase à 200 kHz	V_{det} = -15 V	-	-45	-	degré
t_r,t_f	Temps de montée/descente	-	-	1,75	-	µs
Boîtier						
C_{IO}	Capacité d'entrée/ de sortie	V_F = 0 V, f = 1 MHz	-	1,0	-	pF
C_{cm}	Capacité en mode commun	V_F = 0 V, f = 1 MHz	-	0,5	-	pF
CMRR	Réjection en mode commun	f = 60 Hz, R_L = 2,2 kΩ	-	130	-	dB

IL300

Optocoupleur linéaire

ELEKTOR INFOCARTE I/2002

Fabricant :

Infineon Technologies *www.infineon.com*

Caractéristiques :

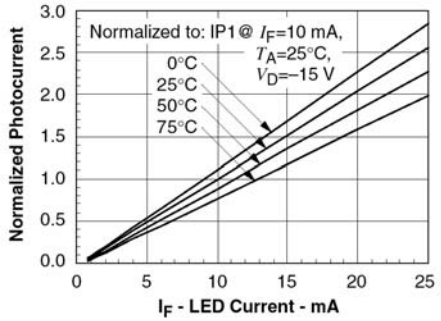
– Couplage de signaux CA et CC
– Erreur de linéarité de 0,01%
– Bande passante étendue > 200 kHz
– Stabilité élevée du gain (±0,05%/° C)
– Faible capacités de l'entrée et de la sortie
– Dissipation faible <15 mW
– Tension d'isolation 5,5 kV_{RMS} pendant 1 s
– Espace d'isolation interne selon normes VDE > 0,4 mm

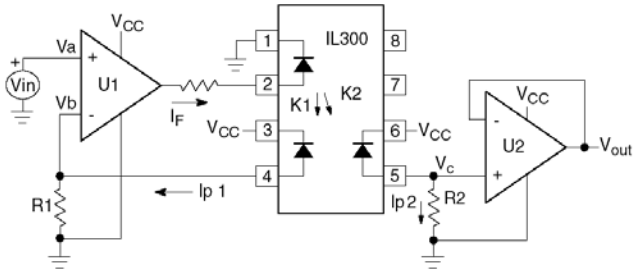
Applications :

– Réinjection de courant/tension dans alimentations
– Isolation de capteurs médicaux
– Transmission de signaux audio
– Transmetteurs isolés pour pilotage de processus
– Centraux numériques isolés

Description :

L'opto-coupleur linéaire IL300 se compose d'une diode infrarouge à l'arséniure de gallium (AlGaAs) qui pilote simultanément une diode PIN de réinjection et une diode PIN de sortie. La photodiode de contre-réaction reçoit une partie du rayonnement IR émis par la LED et produit un signal de commande (IP_1), que l'on pourra utiliser pour le pilotage du courant de commande de LED et compenser ainsi la non-linéarité. La diode PIN de sortie produit un signal de sortie (IP_2) linéaire au rayonnement de la LED. La stabilité en température dans le temps de la jonction de transfert de l'opto-coupleur (K3) est garantie par la mise en oeuvre de diodes PIN appairées. Une application typique utilise un amplificateur opérationnel à l'entrée chargé de piloter les LED. La photodiode de contre-réaction gère, à travers R1, un courant en direction de l'entrée de l'amplificateur opérationnel. Le photo-courant IP_1 répond à la formule $IP_1 = V_{IN} / R1$. Le courant est de ce fait directement proportionnel à K1 et au courant de LED ($V_{IN}/R1 = K1 \cdot I_F$). L'amplificateur opérationnel alimente la LED de manière à garantir la présence d'un photo-courant suffisant en vue de garder la tension nodale V_b égale à V_a. La photodiode de sortie est connectée à un suiveur de tension non inverseur. La résistance de charge R2 convertit le courant de sortie en une tension correspondante qui répond à la formule suivante : $V_O = I_F \cdot K2 \cdot R2$. On obtient ainsi un rapport de transfert général de V_O/V_{IN}, = (K2· R2)/(K1·R1), qui est partant totalement indépendant du courant de la LED. K3, le rapport de transfert du IL300 est fonction du rapport du gain de sortie K2 sur le gain d'entrée K1. Dans ces conditions, on peut également exprimer le rapport de transfert à l'aide de la formule suivante : V_O/V_{IN}, = K3·(R2/R1).





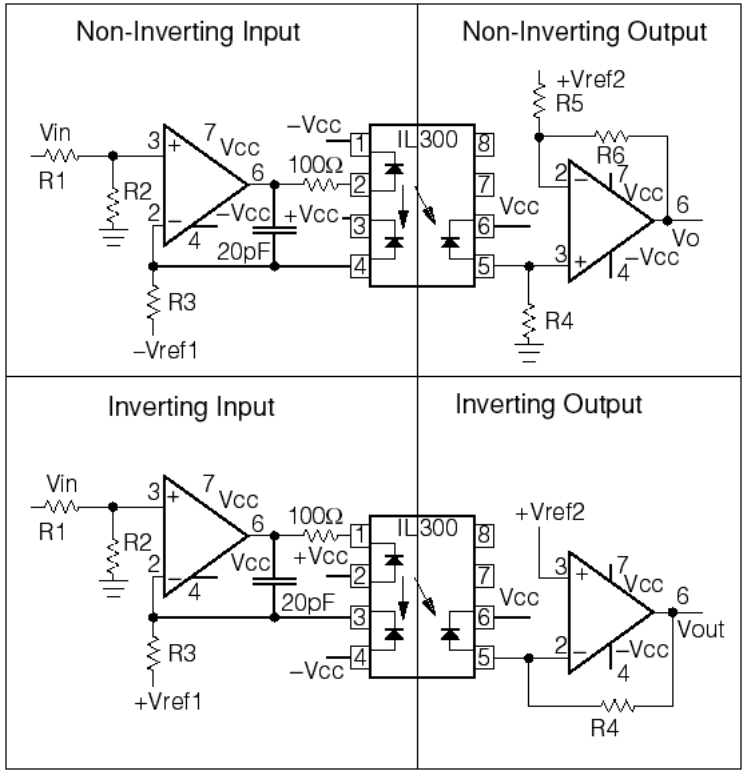
IL300

Optocoupleur linéaire

ELEKTOR INFOCARTE I/2002

Amplificateur opto-linéaire :				
Amplificateur	**Entrée**	**Sortie**	**Gain**	**Dérive** (offset)
non inverseur	inverseuse	inverseuse	V_{OUT}/V_{IN} = [K3·R4·R2] / [R3·(R1 + R2)]	Vref2 = Vref1·R4·K3 / R3
	non inverseuse	non inverseuse	V_{OUT}/V_{IN} = [K3·R4·R2·(R5 + R6)] / [R3·R5·(R1 + R2)]	Vref2 = – [Vref1·R4·K3·(R5 + R6)] / [R3·R6]
inverseur	inverseuse	non inverseuse	V_{OUT}/V_{IN} = – [K3·R4·R2·(R5 + R6)] / [R3·R5·(R1 + R2)]	Vref2 = – [Vref1·R4·K3·(R5 + R6)] / [R3·R6]
	non inverseuse	inverseuse	V_{OUT}/V_{IN} = – [K3·R4·R2] / [R3·(R1 + R2)]	Vref2 = – Vref1·R4·K3 / R3

Configuration en amplificateur non-inverseur et amplificateur inverseur



IL300

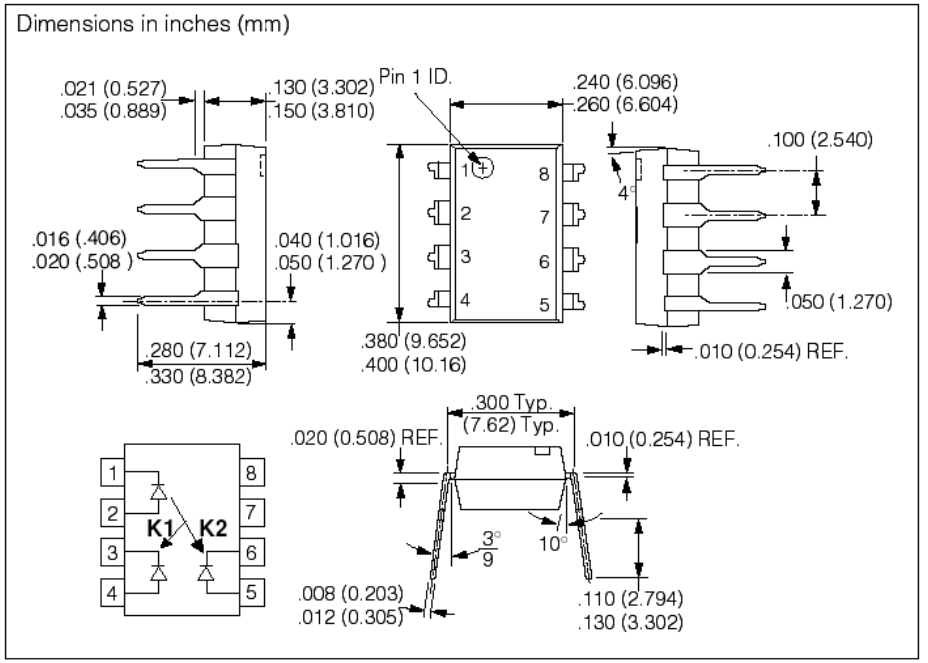
Optocoupleur linéaire

ELEKTOR INFOCARTE I/2002

Exemple d'application

Amplificateur de séparation
Elektor n° 283, janvier 2002

Brochage et types de boîtiers :



Alimentation numérique

3ème partie : le logiciel

Projet : Ralf Pagel

C'est à un microcontrôleur programmé en PIC-BASIC qu'est confié le pilotage de l'alimentation numérique, l'environnement utilisateur tournant sur le moniteur du PC étant lui un programme écrit en Visual BASIC.

La **figure 1** donne l'ordinogramme du programme présent dans le microcontrôleur. Il se traduit par la mise à zéro des valeurs de consigne et le paramétrage de quelque-unes des broches du contrôleur.
L'étape suivante, la mesure effective des valeurs de tension et de courant, fait déjà partie de la boucle principale du programme. Tous les autres pas de programme se font successivement dans cette boucle. On ne rencontre le premier embranchement qu'au niveau de la scrutation (interrogation) des touches. Vu sous la forme d'un ordinogramme, le processus de scrutation paraît plus imposant qu'il ne l'est en réalité; il était tout simplement impossible de le représenter différemment. En principe, on effectue la scrutation d'une touche après l'autre et s'il rencontre une touche enfoncée le microcontrôleur se la note dans l'arrière-plan de sa cervelle sous la forme d'un numéro d'identification. Tout au bas de l'ordinogramme, à la fonction « *Action sur bouton* », on détermine s'il y a, oui ou non, effectivement action sur l'une ou l'autre touche.
Si la réponse à cette question est affirmative on a embranchement vers une partie de programme chargée de diminuer au d'augmenter la valeur de consigne correspondante, et ce dans les limites minimum et maximum admissibles. On a ensuite affichage de la nou-

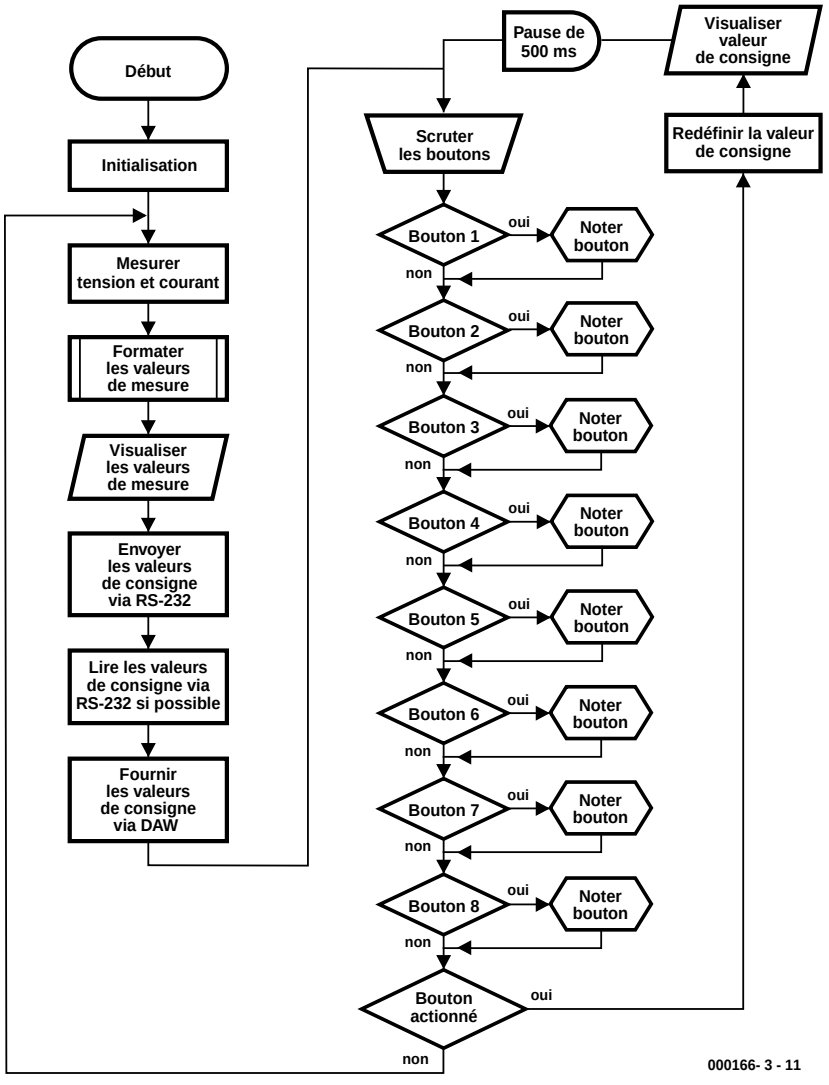


Figure 1. L'ordinogramme du programme du microcontrôleur.

```
'D-PSU 25V, 2.5A or 20V, 1A

'attention: modifications to the program require that register
      numbers
'in the assembler subroutines are checked for changes!!!

'_____________________________
'declaring the variables
VarB Lh1, Lh2, Lh3, Lh5, Lh6, Lh7, Uvalue, Ivalue, y
VarB Buttonnumber, Accu, Callcounter, Bitpattern
VarW Meas_Voltage, Meas_Current

'_____________________________
'Main program
Init:
CV Uvalue, Ivalue  'set to 0 on each start'
Low A3           'ADC output at 0
High B2          'CTS: not ready to receive

Start:
'Measure voltage and current
'Using value 5??? allows ADC scale factor to be adjusted
'                              + - 20 equals approx. 1 digit
Low A4   'Mux to U
ADW A2, 5380, 0, Meas_Voltage  'Voltage measurement
Meas_Voltage = Meas_Voltage Shr 1   'equals / 2
High A4   'Mux to I
ADW A2, 5380, 0, Meas_Current  'Current measurement
Meas_Current = Meas_Current Shr 1   ' equals / 2   'line for
      2.5A
'Meas_Current = Meas_Current Shr 2   ' equals / 4   'line for
      1A

'Format measured values
Call Format

'Display measured values on LCD
LCD B5, " ", Lh1, Lh2, ",", Lh3, "V    ", Lh5, ",", Lh6, Lh7,
      "A "

'Send measured values over RS232
SerOut B3, 9600, "D", #Meas_Voltage, #Meas_Current, 13

'Allows a new target value to be received over RS232
Call RS232E

'Send target values over DAC
PWM A1, Uvalue, 64  'Set voltage (200 = 20V)
PWM A0, Ivalue, 64  'Set current (200 = 2A or 200 = 1A)


'Scan buttons

Entry:

Accu = %00010000  'Bit 4 High (reset by pressed button)
CV Callcounter, Buttonnumber
Call ButtonScan
Call ButtonScan
Call ButtonScan
Call ButtonScan
Call ButtonScan
Call ButtonScan
Call ButtonScan
Call ButtonScan
Branch Buttonnumber, Start, Button1, Button2, Button3, Button4,
      Button5, Button6, Button7, Button8


Button1:
If Uvalue > 240 Then Skip   'Line for 2.5A
'If Uvalue > 190 Then Skip   'Line for 1A
Uvalue = Uvalue + 10
Goto Display_Uvalue

Button2:
If Ivalue > 240 Then Skip   'Line for 2,5A
'If Ivalue > 190 Then Skip   'Line for 1A
Ivalue = Ivalue + 10
Goto Display_Ivalue

Button3:
If Uvalue < 10 Then Skip
Uvalue = Uvalue - 10
Goto Display_Uvalue

Button4:
If Ivalue < 10 Then Skip
Ivalue = Ivalue - 10
Goto Display_Ivalue

Button5:
If Uvalue > 249 Then Skip   'Line for 2,5A
'If Uvalue > 199 Then Skip   'Line for 1A
Inc Uvalue
Goto Display_Uvalue

Button6:
If Ivalue > 249 Then Skip   'Line for 2.5A
'If Ivalue > 198 Then Skip 2  'Line for 1A
Inc Ivalue
'Inc Ivalue  'Line for 1A (omit for 2.5A version)
Goto Display_Ivalue

Button7:
If Uvalue < 1 Then Skip
Dec Uvalue
Goto Display_Uvalue

Button8:
If Ivalue < 1 Then Skip   'Line for 2.5A
'If Ivalue < 2 Then Skip 2  'Line for 1A
Dec Ivalue
'Dec Ivalue  'Line for 1A (omit for 2.5A version)


Display_Ivalue:
'y = Ivalue Shr 1 'equals / 2  'Line for 1A (omit for 2.5A ver-
      sion)
LCD B5, "     ", #Ivalue, "0mA"  'Line for 2.5A
'LCD B5, "     ", #y, "0mA"  'Line for 1A
Pause 500
Goto Entry


Display_Uvalue:
LCD B5, "     ", #Uvalue, "00mV"
Pause 500
Goto Entry


'_____________________________

'Subroutines

'Depending on value in Callcounter, ButtonScan shifts one of
'eight bitpatterns to the pins of the HC164.
'Only the button at the pin with the 0 on it
'can pull PB4 Low. PB4 then indicates if a button was pressed
      or not,
'while Callcounter reveals the button identity


Sub ButtonScan
   LookUp Callcounter, %11101111, %11011111, %10111111,
      %01111111, %11111011, %11110111, %11111110, %11111101,
      Bitpattern
   EXPo B5, Bitpattern, 0  'only Button 0 of bit pattern can
      pull B4 Low
   Inc Callcounter
   PBI %00010000 = Accu        'read only bit 4 of Port B
   If Accu <> 0 then Skip        'skip when no button pressed
   Buttonnumber = Callcounter       'mark Button number
EndSub


'The Basic subroutine Read is called from
'assembler subroutine RS232E

Sub Read
   SerIn B0, 9600, #Uvalue, #Ivalue
   Uvalue = Uvalue Min 250           'limit to 25 Volt  'Line
```

Figure 2. Listage en PIC-BASIC.

```
      for 2.5A
   'Uvalue = Uvalue Min 200          'limit to 20 Volt  'Line
      for 1A
   Ivalue = Ivalue Min 250          'limit to 2.5 Ampere
      'Line for 2,5A
   'Ivalue = Ivalue Min 200          'limit to 1 Ampere  'Line
      for 1A
   Y = 1          'Leave loop immediately
Endsub


'Assembler sub-routine Format employs the already available
'Resources for PB. It load the number registers Lh1-Lh8 with
      the
'ASCII values for Numbers 0-9 according to the values in the
'variables Meas_Voltage and Meas_Current.

'The auxiliary subroutine called Packer saves 8 bytes of pro-
      gram memory
'Packer calls machine code program SOSS°, which is contained in
      the
'PB compiler output, when the commands SerOut - #WordVar
'or LCD - #WordVar" was employed.
'It returns the decimal number equivalent of a wörd variable.
'It divides te value contained in HWERT2/R21 by the value from
      the
'jump table SOTT° (also contained in compiler output).
'The value(!) in the FSR has to be the ADD value
'of the jump table (Pos. 5 = 0, 4 = 2, 3 = 4, 2 = 6, 1 = Rest
      in R21).
'LWERT1 contains the ASCII code (characters 0-9) as the result.

Ass Format
      ;format voltage
   MOVF 24,W
   MOVWF HWERT2
   MOVF 23,W
   MOVWF 21
   MOVLW 2
   Call Packer
   MOVWF 27
   MOVLW 4
   Call Packer
   MOVWF 28
   MOVLW 6
   Call Packer
   MOVWF 29
      ;format current
   MOVF 26,W
   MOVWF HWERT2
   MOVF 25,W
   MOVWF 21
   MOVLW 2
   Call Packer
   MOVWF 30
   MOVLW 4
   Call Packer
   MOVWF 31
   MOVLW 6
   Call Packer
   MOVWF 32
   Return

Packer:      ;no repeating of lines; saves 8 bytes of program
      memory
   MOVWF FSR
   CALL SOSS°
   MOVF LWERT1,W
EndAss


'RS232E controls data reception at the interface. Each time it
      it called, the CTS line is pulled High for 1.5ms.
'If a character arrives via RxD within this period, the D-PSU
      goes into Receive mode i.e.
'subroutine Read is called. Next, 2 values with terminating CRs
'have to arrive at the interface before the controller is allo-
      wed
'to leave the subroutine

Ass RS232E
   CLRF 35        ;Clrf Y (= R35)
 RS232:
   BCF PB,2       ; CTS: ready to receive
   BTFSS PB,0     ; RxT pin test
   Call Read
   DECFSZ 35,F    ;
   GOTO RS232     ; repaet loop 256 times
   BSF PB,2       ; CTS: not ready to receive
EndAss
```

velle valeur de consigne. Il se passe ensuite une pause d'une demi-seconde avant que ne prenne place une nouvelle scrutation des boutons. On crée ainsi une fonction de répétition. En l'absence d'action sur une touche quelconque le programme retourne dans la boucle principale où il effectue les mesures de tension et de courant.

Programme BASIC

Le listage du programme du contrôleur est donné en **figure 2**. Le programme destiné au contrôleur a été écrit en PIC-BASIC-1.3; il est disponible au téléchargement à l'adresse Internet d'Elektor (*www.elektor.presse.fr*).
PIC-BASIC permet une écriture rapide et confortable de programmes pour contrôleur. Il est en outre facile, par son intermédiaire, de compiler le programme terminé et de programmer le composant. On pourra trouver des informations additionnelles concernant PIC-BASIC à l'adresse Internet *www.pic-basic.de*. On y trouvera des informations supplémentaires quant à la version la plus récente de PIC-BASIC.
La **figure 3** montre l'alimentation de type 1 A lors du processus ICP (*In Circuit Programming*) avec le programmateur PIC-BASIC. On commence par déclarer toutes les variables de programme utilisées. Il y en a au total 15, 13 variables du type octet (*Byte*) et 2 du type mot (**Word**). Ces variables occupent partant un total de 17 octets de mémoire de RAM du contrôleur (auxquels s'ajoutent 12 octets de mémoire de travail toujours occupés par PIC-BASIC). On trouve ensuite la première partie du programme. Il s'agit de la partie représentée par le module « *Initialisation* » de l'ordinogramme. « *Début* » représente l'étiquette de démarrage de la boucle principale. Les commentaires qui accompagnent le programme sont suffisamment exhaustifs pour que nous n'ayons pas à entrer dans son détail. Nous nous limiterons à quelques remarques :

Le convertisseur A/N

L'instruction

```
ADW A2, 5380, 0, Meas_Voltage
```

démarre une conversion analogique/numérique et stocke la valeur de mesure dans les variables baptisées `Meas_Voltage`. Il est possible, par modification de la valeur `5380`, d'ajuster le facteur d'échelle. Le dimensionnement du circuit est cependant conçu de telle façon que cela ne soit pas, normalement, nécessaire.

Routines en assembleur

Au niveau de l'étiquette « `Format measured values` » il est fait appel à une routine (sous-programme) en assembleur baptisée « `Format` ». Cette routine de 30 octets seulement

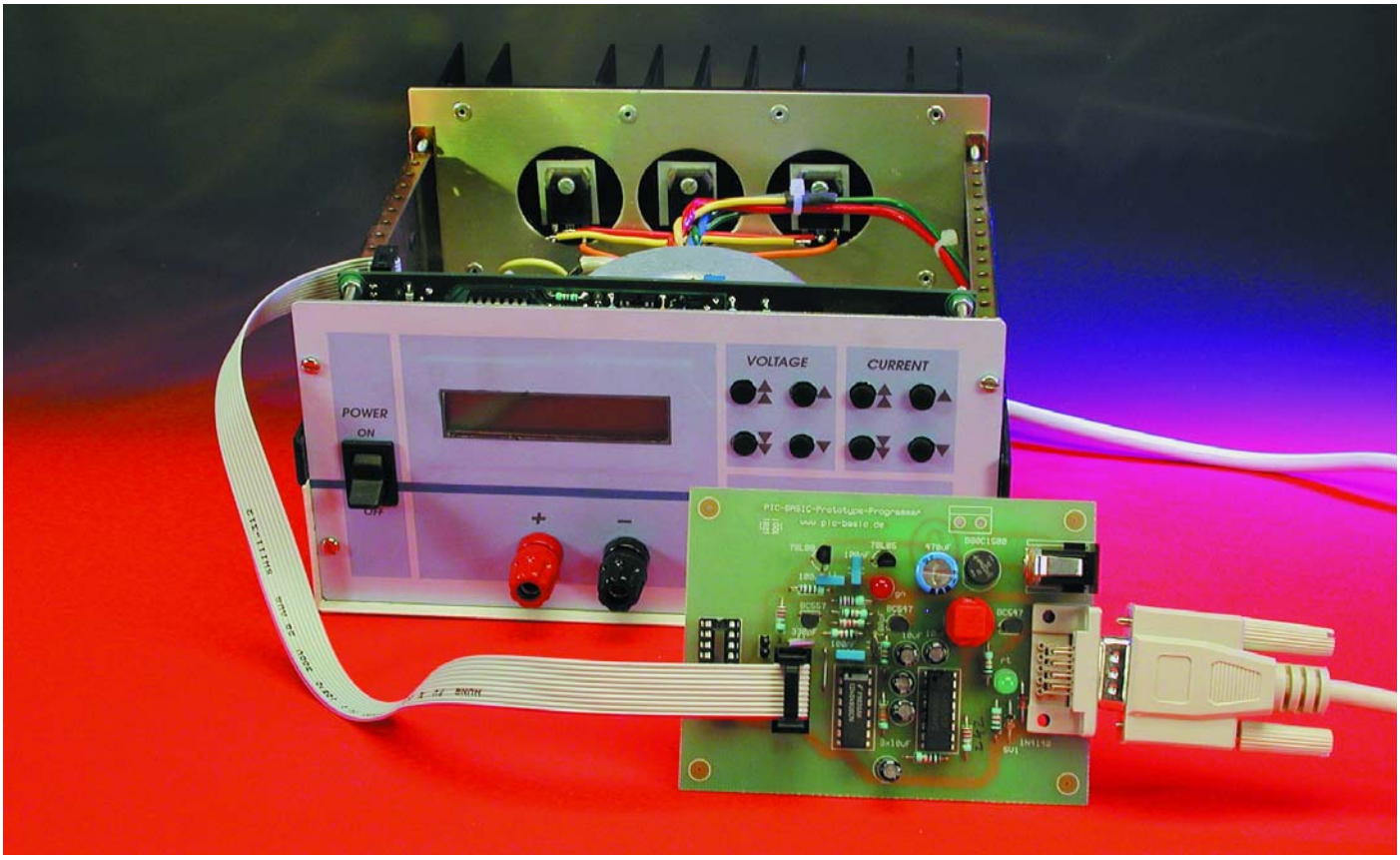


Figure 3. Programmation en circuit (ICP) du microcontrôleur de l'alimentation.

fait appel à quelques routines (en PIC-BASIC) qui existent de toutes façons déjà par le biais d'autres instructions Basic en vue de formater la valeur de mesure pour qu'elle puisse être visualisée. Cette technique astucieuse permet d'économiser de l'espace en mémoire de programme toujours précieux.
La routine `RS232E` de 8 octets est elle aussi écrite en assembleur. Elle force la ligne CTS au niveau haut et attend ensuite un certain temps l'émission de données par le PC. Dès que cela est le cas, il arrive partant des données en provenance du PC, la routine en assembleur appelle une routine en Basic baptisée `Einlesen` qui se charge de la prise en compte des données proprement dite.
Le reste du programme est écrit en Basic. Une fois compilé, l'ensemble du programme a une taille de 1 009 ou de 1 021 octets (selon qu'il s'agit de la version 2,5 ou 1 A), ce qui lui permet tout juste de trouver place dans la mémoire du PIC16F84. Les lignes de programme spécifiques à chacune des 2 versions de l'alimentation numérique sont identifiées comme telles dans le listage Basic.
Si l'on tient à ce que le microcontrôleur travaille avec chien de garde (*watchdog*) activé il faudra modifier en conséquence le mot de configuration (*Configword*) du listage assembleur et le remplacer par :

```
CONFIG B'11111111110101'
```

Il faudra en outre, dans le listage assembleur,

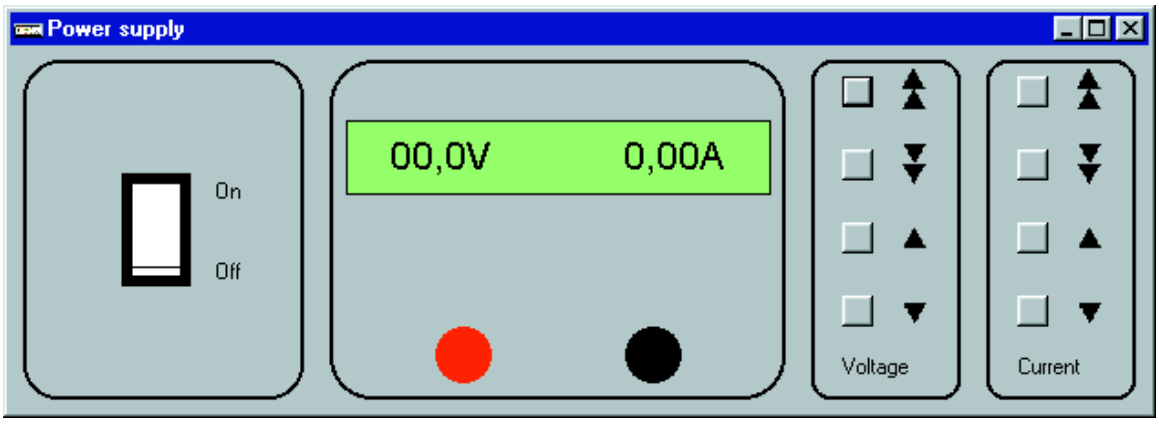


Figure 4. Recopie d'écran de l'information fournie par l'alimentation.

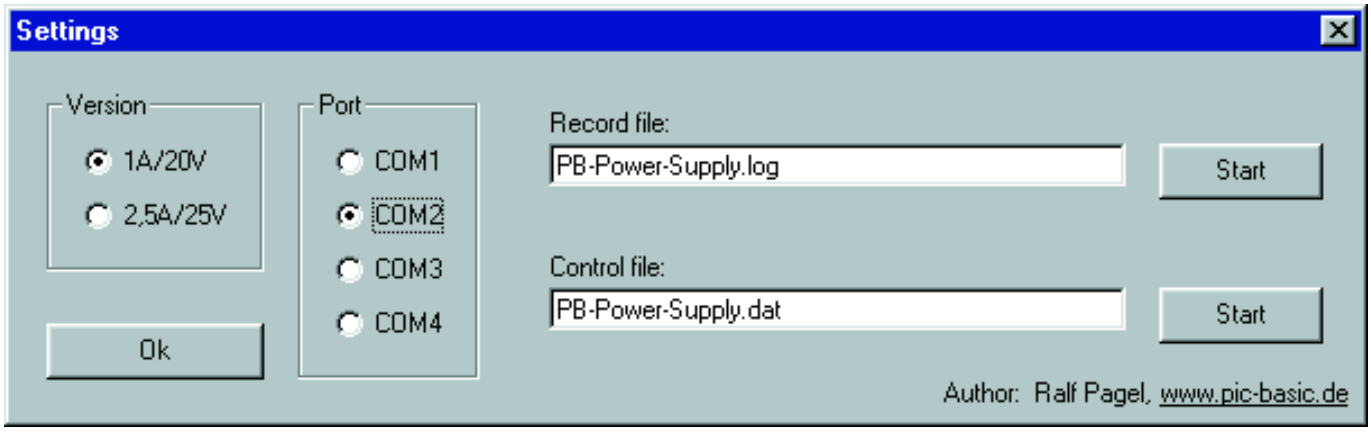


Figure 5. La fenêtre de paramétrage de l'alimentation numérique.

intercaler l'instruction `CLRWDT` (*Clear Watch-Dog Timer*) à un endroit donné qui se trouve à la fois dans la boucle principale et dans la boucle de scrutation des boutons. Cette dernière boucle est parcourue en un peu plus de 500 ms, la durée d'exécution de la boucle principale étant elle de l'ordre de 780 ms.
Avec les valeurs présentes, après activation, dans le registre Option on a déclenchement, après 2,3 s d'une réinitialisation du chien de garde. Cette durée est largement suffisante pour les 2 boucles évoquées plus haut (même une temporisation de durée 2 fois plus faible suffirait également). L'option la plus pratique est de positionner l'instruction `CLRWDT` directement après l'étiquette « `Einsprung` ». Cela a pour conséquence de décaler l'ensemble du code d'un octet, mais ce déplacement n'a, dans le cas présent, rien de critique.

Le protocole de l'interface

Le paquet de données transmis par l'alimentation par le biais de son interface possède le format suivant :

```
Duuuuuiiiii↵
```

On commence par envoyer un D, puis une valeur à 5 positions pour la tension (u) et autant pour le courant (i), le tout suivi par un caractère de retour chariot (CR). La dernière position de la valeur de tension correspond à 10 mV, celle de la valeur de courant à 1 mA. On remarque de ce fait que la résolution de ces 2 valeurs est 10 fois supérieure à celles visualisées par l'afficheur LCD.
La première position de la valeur de tension et du courant est toujours un zéro. Lorsque l'on envoie des valeurs de consigne vers l'alimentation il faut à chaque fois envoyer immédiatement l'une après l'autre les 2 valeurs (tension et courant). Ces 2 valeurs peuvent, à l'émission, comporter un maximum de 3 caractères numériques (chiffres)

```
uuu iii↵
```

Chaque valeur doit être suivie par un caractère non-numérique, un caractère de retour chariot par exemple. Quelle que soit la version de l'alimentation réalisée, il faudra envoyer le nombre `200` pour une valeur de tension de 20 V. Pour une valeur de courant de 1 A au contraire, les valeurs à entrer sont différentes, `100` dans le cas de l'alimentation de 2,5 A, et `200` pour la version 1 A de l'alimentation.

Le programme en Visual BASIC

Le programme de pilotage écrit spécialement à l'intention de ce projet tourne sous Windows '95/'98. Il possède un environnement de commande similaire à la face avant de l'alimentation (**figure 4**). Il est possible, par le biais de 8 boutons, d'ajuster, comme si l'on se trouvait devant l'alimentation « en chair et en os », la valeur de la tension et celle du courant. Lors du relâchement du bouton de la souris, les valeurs sont transmises vers l'alimentation. De même, l'affichage LCD factice visualise la valeur de consigne et la valeur effective. Tout clic sur une plage quelconque à l'intérieur de la fenêtre ne comportant pas de bouton se traduit par l'ouverture d'une fenêtre de paramétrage (**Figure 5**).
Par son biais on définit la version de l'alimentation (2,5 ou 1 A) et le port COM (1 à 4) utilisé pour la transmission des données. La présence d'une souris sur le port COM1 peut entraîner des problèmes de fonctionnement du programme si l'on opte pour le port COM3, mais cette situation problématique est classique dans le cas du PC.
On pourra donner, dans la partie droite de la fenêtre de paramétrage, le nom d'un fichier d'acquisition (*log*) ou de pilotage. Après un clic souris sur l'une des touches de lancement situées à proximité se traduira par, selon le cas, la création d'un fichier ou son écrasement, sachant que l'on aura, à chaque fois, stockage des valeurs visualisées et d'un tampon chronodateur, si tant est qu'elles aient changé ou que l'on pilote l'alimentation par le biais d'un fichier défini auparavant. Dans le cas le plus simple, le fichier de pilotage pourra être un fichier d'acquisition dont on aura modifié les informations horodatrices.
L'exemple ci-après est celui d'un set de données tel qu'il est utilisé dans les fichiers pour l'acquisition et la commande :

```
2000-08-20          14:35:53
,"04.9V","0.97A"
```

On trouve, intercalé entre les 2 caractères « # », la date au format international et l'heure, instant auquel a eu lieu ou devra avoir lieu le passage à la valeur suivante. Dans le set de données, le tampon horodateur et les 2 valeurs électriques sont séparées par une virgule. Il est partant possible de traiter ces fichiers à l'aide de programmes tableurs pour, par exemple, obtenir un affichage sous forme graphique des valeurs relevées.

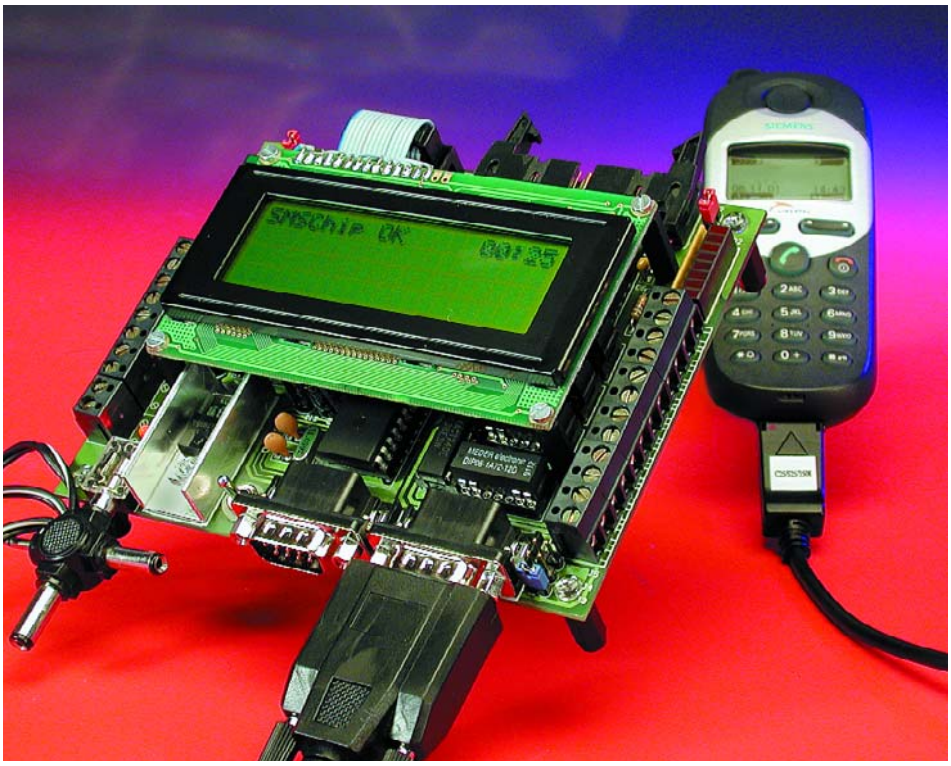
(000166-3)

Télécommande par portable et SMS (I)

Pour les modèles de la série 35 de Siemens

Prof. Dr. Bernd vom Berg & Dipl.-Ing. Peter Groppe

Les possibilités des téléphones mobiles vont bien plus loin que le simple envoi de messages. Il est devenu possible, à l'aide de téléphones portables, de piloter et de surveiller des processus complexes.



La TFH (*Technischen FachHochschule*) Georg Agricola de Bochum (RFA) a mis au point, en collaboration avec la société Engelmann & Schrader, une petite carte d'expérimentation SMS baptisée SMS ExBo (pour *Experimental Board*) qui, une fois connectée à un téléphone portable, permet le suivi et le pilotage de processus partout dans le monde.

Transferts entre SMS et processus

Côté émetteur, l'utilisateur entre sur son portable l'instruction requise sous la forme d'un texte en clair comme il le ferait pour un SMS tout ce qu'il y a de plus classique. Les séquences d'instructions autorisées au cours de ce processus ont une structure simple :

`set 10` = mettre la sortie de la puce SMS au numéro 10. Un relais, moteur ou une LED connecté à cet endroit sera activé, une instruction

`reset 10` se traduisant par sa désactivation.

Le SMS constitué d'un nombre quelconque d'instructions de commande est envoyé comme d'habitude par le biais du portable « émetteur ».

Côté « réception » on aura à nouveau besoin d'un portable dont l'interface sérielle est connectée à la carte SMS ExBo. Cette carte additionnelle comporte un circuit intégré (que nous avons baptisé « **noyau SMS** ») qui assure, une fois la tension d'alimentation appliquée, la reprogrammation du portable, par le biais de son interface sérielle de manière à être en mesure de traiter les SMS en externe. Cette reprogrammation a des conséquences très importantes : lors de la réception d'un SMS le portable produit le même signal sonore que d'habitude et le symbole signalant la réception d'un SMS s'affiche sur l'écran du téléphone. À partir de là c'est à la puce SMS de prendre les affaires en main : il a, auparavant, reprogrammé le portable de manière à ce que les SMS « entrants » soient

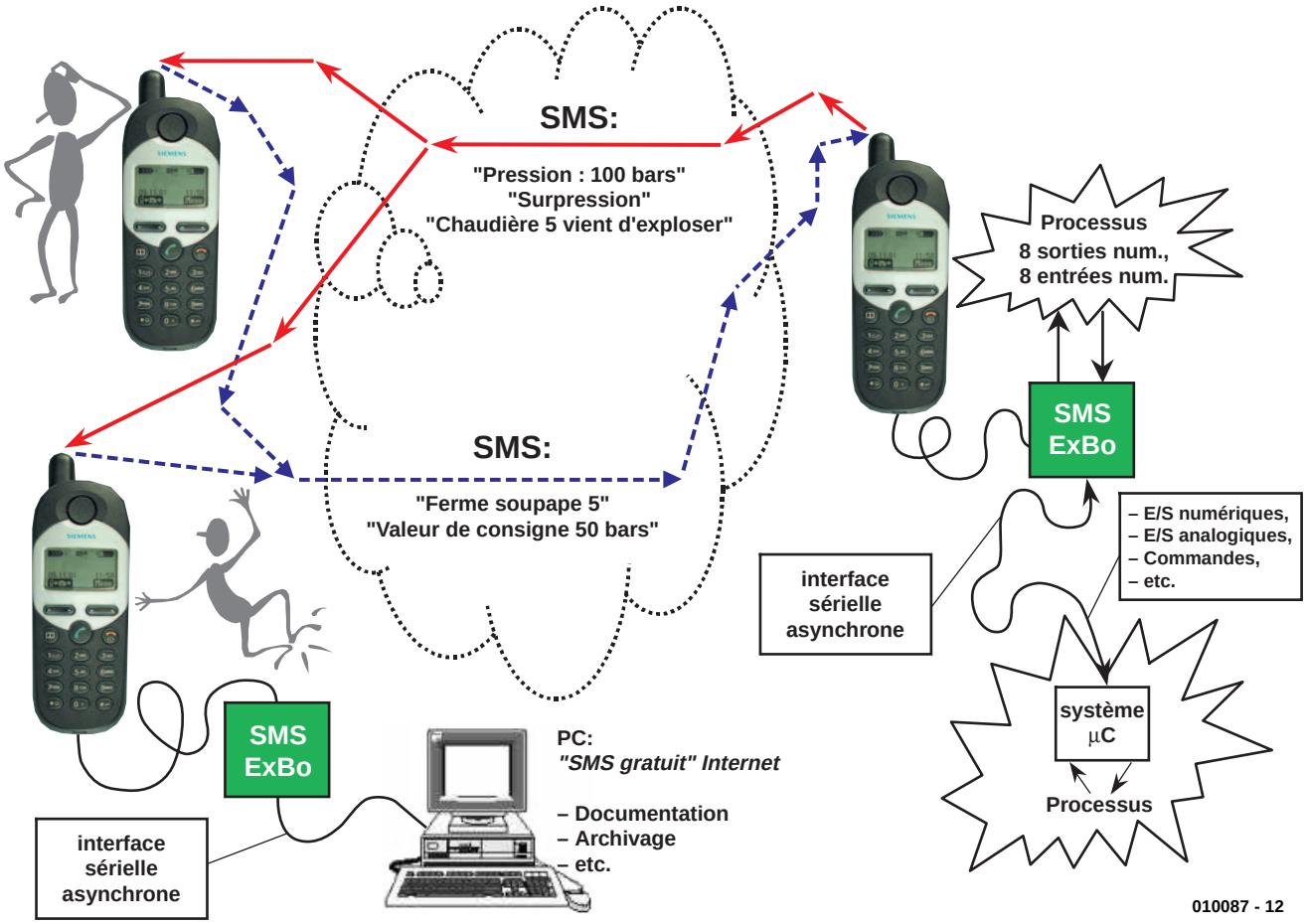


Figure 1. Télécommande de processus par le biais d'un portable et de message SMS.

directement transmis vers l'interface sérielle du portable. Ceci permet au circuit SMS de passer immédiatement à leur traitement.
Si le noyau SMS reconnaît dans le message SMS des instructions valides protégées par mot de passe il en effectue le décodage et les exécute.
Dans l'autre sens, le noyau SMS peut procéder à l'acquisition de données de processus et les transmettre, sous la forme d'un SMS, au portable connecté à la carte SMS-ExBo par le biais de l'interface sérielle. Depuis ce téléphone le dit SMS de données est transmis tout normalement à un quelconque téléphone situé n'importe où sur terrer. Le « récepteur » de ce message verra ainsi, par exemple, s'afficher sur son portable le message SMS suivant : *Surpression, Chaudière 5 vient d'exploser* et pourra prendre les dispositions requises (se mettre à paniquer !). La **figure 1** illustre l'ensemble de ce processus.
Mais notre noyau SMS peut encore bien plus : il comporte une seconde interface sérielle asynchrone par le biais de laquelle il peut transmettre, tels quels et directement, les SMS reçus à un système micro-informatique externe. Dans ces conditions l'utilisateur pourra, à l'aide de son propre set d'instructions, piloter et surveiller un système encore plus complexe; on pourra imaginer, par exemple, la télécommande d'un automate par le biais d'un téléphone portable et de messages SMS.
De son côté, cet automate de processus plus complexe (cette PLC) pourra à son tour générer son propre SMS (valeurs de mesure, données concernant le déroulement de processus etc.) et le transmettre au noyau SMS par le biais de la seconde interface sérielle. À partir de ce niveau le SMS est transmis, sans avoir subi la moindre modification, au portable d'où il sera immédiatement envoyé dans les éthers vers le destinataire.
Il est possible ainsi, sans le moindre problème, d'interconnecter un PC à la paire constituée par la carte SMS-ExBo et un portable.

Le noyau SMS

Le noyau SMS est un microcontrôleur du type AT89LS8252 ou AT89S8252 d'Atmel, un dérivé du 8051 doté de 8 Koctets de mémoire de programme et de 2Koctets de mémoire de données, toutes 2 du type Flash programmées en conséquence. Il remplit en fait en fait 3 fonctions :

– Reprogrammation du portable et communication avec ce dernier par le biais de l'interface sérielle.

– Réception et traitement des SMS : interrogation et pilotage des entrées et des sorties, génération de propres SMS.

– Transmission et réception de SMS en provenance/en direction d'un autre système micro-informatique.

Au nombre des ports dont dispose le AT89S8252, les ports 1 et 2 (soit un total de

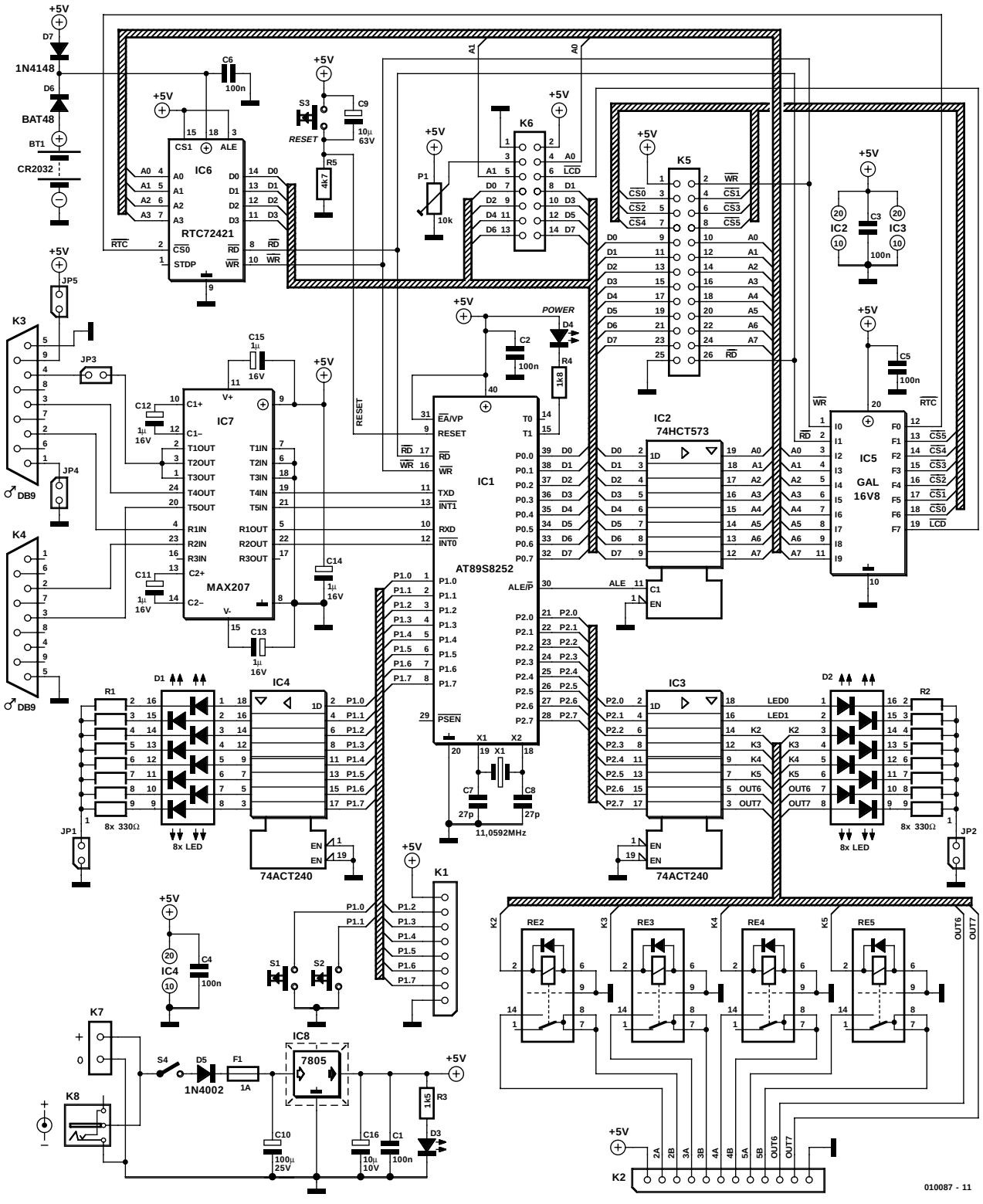


Figure 2. L'électronique de la carte d'expérimentation SMS.

16 lignes de port programmables individuellement soit en entrée soit en sortie) sont à la disposition de l'utilisateur pour la commande de processus. L'une des interfaces sérielle est prévue pour la communication avec le portable, l'autre, un UART (*Universal Asynchronous Receiver/Transmitter* = Émetteur/Récepteur Asynchrone Universel) logiciel travaillant à 9 600 bauds, pour la communication avec un système à microcontrôleur quelconque (un PC ou un automate programmable par exemple).

On pourra connecter au port 0 du noyau SMS d'éventuelles extensions telles qu'horloge en temps réel externe à alimentation de sauvegarde par pile, un affichage LCD alphanumérique pour la visualisation des messages de texte.

Le noyau SMS communique avec le

Tableau 1. Répartition des lignes de port programmables		
Port	**Broche**	**Fonction**
P1.0	0	ENTRÉE : Connexion de la touche S1
P1.1	1	ENTRÉE : Connexion de la touche S2
P1.2	2	ENTRÉE : niveau TTL, non protégé
P1.3	3	ENTRÉE : niveau TTL, non protégé
P1.4	4	ENTRÉE : niveau TTL, non protégé
P1.5	5	ENTRÉE : niveau TTL, non protégé
P1.6	6	ENTRÉE : niveau TTL, non protégé
P1.7	7	ENTRÉE : niveau TTL, non protégé
P2.0	8	SORTIE : Connexion de la LED0 du réseau de LED D2
P2.1	9	SORTIE : Connexion de la LED1 du réseau de LED D2
P2.2	10	SORTIE : Relais Re2; 200 V_{CC} max., 1 A max., 15 W max.
P2.3	11	SORTIE : Relais Re3; 200 V_{CC} max., 1 A max., 15 W max.
P2.4	12	SORTIE : Relais Re4; 200 V_{CC} max., 1 A max., 15 W max.
P2.5	13	SORTIE : Relais Re5; 200 V_{CC} max., 1 A max., 15 W max.
P2.6	14	SORTIE : niveau TTL par circuit de commande 74AC/ACT240
P2.7	15	SORTIE : niveau TTL par circuit de commande 74AC/ACT240

portable par le biais d'un set d'instructions on ne peut plus connu et respecté, connu sous la dénomination de standard Hayes (instructions AT). Nous lui consacrons un encadré en fin d'article.

La « cour » matérielle du noyau SMS

La mise en oeuvre pratique du noyau SMS est on ne peut plus simple et pourra se faire, dans le cas d'une configuration minimale, sous la forme d'un petit système autonome ne comportant rien de plus que le noyau SMS, un quartz et quelques résistances et condensateurs.

Nous avons réalisé, avec le SMS ExBo, un petit système expérimental étendu grâce auquel on peut tirer meilleur parti des caractéristiques du noyau SMS. Un coup d'oeil à son schéma représenté en **figure 2** permet de constater qu'il s'agit d'une carte à microcontrôleur classique dotée d'un certain nombre d'extensions.

Le noyau SMS (IC1, en version DIL) se voit doté de la circuiterie d'horloge standard à base de quartz, X1, et de 2 condensateurs, C7 et C8, et de l'électronique de réinitialisation (Reset) classique à base de S3, C9 et R5, éléments typiques dans le cas des microcontrôleurs de la famille 8051. La LED faible consommation D4 prise dans la ligne de port P3.5 (Power) visualise l'état de la ligne de transmission de données vers le portable. Il n'en faut pas plus pour constituer notre système minimum.

Le **tableau 1** donne les 16 lignes de port numériques mises à notre disposition par le circuit SMS.

Les entrées numériques (P1.2 à P1.7)

sont reportées au bornier à vis K1. Le niveau de chaque entrée est en outre visualisé à l'aide d'une LED (circuit de commande IC4, réseau de LED D1, réseau de résistances R1). On pourra, si l'on veut travailler en mode économie de courant, désactiver le réseau de LED en sortant le cavalier JP1. Les entrées numériques P1.0 et P1.1 sont reliées une fois pour toutes aux boutons-poussoirs S1 et S2.

Les sorties numériques (P2.0 à P2.7)

sont reliées aux bornes de sortie correspondantes au travers du circuit de commande IC3. À nouveau, les états des sorties sont visualisés à l'aide de LED (réseau de LED D2 protégées par le réseau de résistances R2). En l'absence du cavalier JP2 les LED sont désactivées. Les contacts de commutation des relais et les 2 sorties TTL OUT6 et OUT7 sont reportées au bornier à vis K2.

Les interfaces sérielles

L'adaptateur de niveau IC7, un MAX207, se charge de la conversion des niveaux TTL vers des niveaux RS-232 (V24), dans un sens ou dans l'autre et cela pour les 2 interfaces. Il génère en outre à ses sorties T1OUT, T2OUT et T3OUT une tension de l'ordre de +10 V utilisée pour l'alimentation du câble de transfert de données (*Data Link*) vers le portable, tension amenée à la broche 4 de l'embase K3. On pourra interrompre le passage de la tension d'alimentation en n'implantant pas le cavalier JP3.

La tension de charge pour son accu est transmise au portable par le biais de l'embase sub D K3 et le cavalier JP5, de manière à permettre la recharge du téléphone portable au travers de la platine d'expérimentation. L'embase sub-D K4 fait office de sortie pour la seconde interface sérielle du noyau SMS en vue de permettre la communication SMS avec d'autres systèmes micro-informatiques.

Périphériques d'extension pour le noyau SMS

Comme il est possible de connecter au noyau SMS des circuits intégrés périphériques externes, il faut, comme cela est courant dans le cas des microcontrôleurs de la famille 8051, démultiplexer le bus de données/d'adresses multiplexé du port 0. C'est là la tâche de l'octuple bascule D (flip-flop), IC2. Le bus d'adresses et de données ainsi démêlé est reporté, de même que d'autres signaux de commande du 8051 ($\overline{RD}$, $\overline{WR}$) et les signaux de sélection $\overline{CS}$ et la tension d'alimentation (+5 V, GND), sur l'embase K5.

Les périphériques additionnels se trouvant sur la carte ExBo sont, en premier lieu, une horloge en temps réel (dite RTC pour *Real Time Clock*) et un affichage LCD alphanumérique, ces 2 périphériques étant pilotés adéquatement par le noyau SMS.

L'horloge en temps réel

IC6 est alimenté par la pile au lithium BT1 au travers de la diode D6 ou de la diode D7. Ce circuit intégré met à la disposition du système SMS ExBo les informations de tampon horodateur (heure et date), même en cas de disparition de la tension d'alimentation au niveau du jack d'alimentation K8.

L'affichage LCD alphanumérique

se branche à l'embase K6. Il est possible d'y connecter différents types d'affichages, si tant est qu'ils soient compatibles avec le contrôleur HD44780 d'Hitachi. L'illustration en

Communication entre le noyau SMS et le portable

Le noyau SMS communique par le biais du set d'instruction modem compatible avec le standard Hayes et correspondant aux normes GSM07.07 et GSM07.05. Le pionnier américain du modem, la société Hayes, a défini, vers la fin des années 70, un set d'instructions servant à la commande de modems utilisés pour la transmission de données, devenu aujourd'hui le set d'instructions quasi-standard intégré de nos jours dans tout modem mais aussi dans chaque téléphone portable.
Les instructions débutent par la paire de caractères ASCII AT, les instructions normées proprement dite commençant elles par les caractères AT+C et se terminent par un retour chariot (CR = Carriage Return). Ceci explique la dénomination attribuée à cette série de commande pour modem, à savoir set d'instruction AT ou AT+C. Ces instructions ont été reprises pour la télécommande de portables par le biais d'interfaces sérielles (liaison câblée RS-232 ou liaison infrarouge établie selon le standard IrDa). Ceci explique que les spécifications GSM07.07 et GSM07.05 intègrent un set d'instruction comportant pas moins de 55 instructions AT, instructions qu'un portable se doit de « comprendre », telles que, par exemple, commandes pour l'écriture et la lecture de son annuaire téléphonique, le traitement de SMS, la sélection de sonneries et le réglage de leur volume etc. etc.
De par cette normalisation il est partant possible de coupler un portable à un PC, par le biais d'un câble Data Link sériel ou de leur interface infrarouge et de procéder, depuis le PC à l'ensemble du traitement des manipulations à effectuer par le portable, approche bien plus confortable. Outre ce set minimum, les fabricants connaissent également leur propre set d'instructions qui varie d'un fabricant à l'autre et auquel ne réagissent que les portables de la dite marque. Il existe ainsi, dans le cas des portables de la série 35 de Siemens, les S35i, C35i, M35i, 25 instructions additionnelles qui commencent toutes par la série de caractères AT ^ S.
Et c'est bien là que commencent les problèmes : chaque fabricant de téléphones portables peut choisir de supporter toutes les instructions AT des normes GSM, voire quelques-unes d'entre elles seulement et d'ajouter sa propre collection d'instructions. En d'autres termes : il faut donc, si l'on veut pouvoir télécommander un portable par le biais de son interface sérielle (ce qui est notre intention avec le noyau SMS), disposer d'informations détaillées sur le set d'instructions que le portable est en état de « saisir ».
Ceci explique qu'il est, avec la présente version du noyau SMS, possible uniquement de travailler avec les portables de la série 35 de Siemens. Le portable situé à l'autre « bout du fil » pourra en ce qui le concerne, être n'importe quel portable capable de recevoir des messages SMS.
L'utilisateur n'a pas besoin de plus d'informations pour l'instant sachant que le programme intégré dans le noyau SMS programmé se charge du déroulement correct de la communication avec le portable par le biais des commandes AT, de sorte qu'il n'y a pas le moindre réglage ni la moindre programmation à entreprendre. La réception et le traitement des SMS se fait ainsi automatiquement en accord tacite entre le portable et le noyau SMS. Le transfert des SMS par la voie de la seconde interface sérielle se déroule lui aussi sans le moindre problème. La seule tâche de l'utilisateur sera de créer les messages SMS de commande.

début d'article montre un modèle d'affichage de 4 lignes de 20 caractères chacun. L'ajustable P1 permet de régler le contraste de l'affichage.

La GAL,
IC5, une 16V8, génère les signaux $\overline{CS}$ nécessaires à l'horloge en temps réel et à l'affichage LCD. Ce circuit produit en outre 6 signaux $\overline{CS}$ additionnels $\overline{CS0}$ à $\overline{CS5}$ utilisables pour la commande d'autres circuits de périphériques externes. Ces 6 signaux $\overline{CS}$ supplémentaires sont également reportés sur l'embase K5.

La tension d'alimentation
requise par l'ensemble de la carte d'expérimentation est régulée à +5 V par un régulateur de tension intégré, IC8. Le régulateur de tension tire sa tension d'entrée (9 à 12 V/800 mA, lorsqu'il faut recharger le portable) d'un adaptateur secteur que l'on branchera, selon le cas, soit au bornier K7, soit à l'embase jack d'alimentation K8. La diode D5 protège contre une inversion malencontreuse de polarité de la tension d'alimentation, la LED D3 visualise la présence de la tension d'alimentation et S4 fait office de commutateur marche/arrêt général.

Les cavaliers
Le **tableau 2** récapitule les fonctions des cavaliers présents sur la carte d'expérimentation SMS ExBo.

(010087-1)

Dans la seconde partie de cet article nous examinerons de plus près la connexion du portable à la carte, nous intéresserons à la configuration de base et au set d'instructions du noyau SMS. Nous vous proposerons en outre le dessin de la platine d'expérimentation et verrons les aspects intéressants de sa réalisation.

Tableau 2. Signification des cavaliers

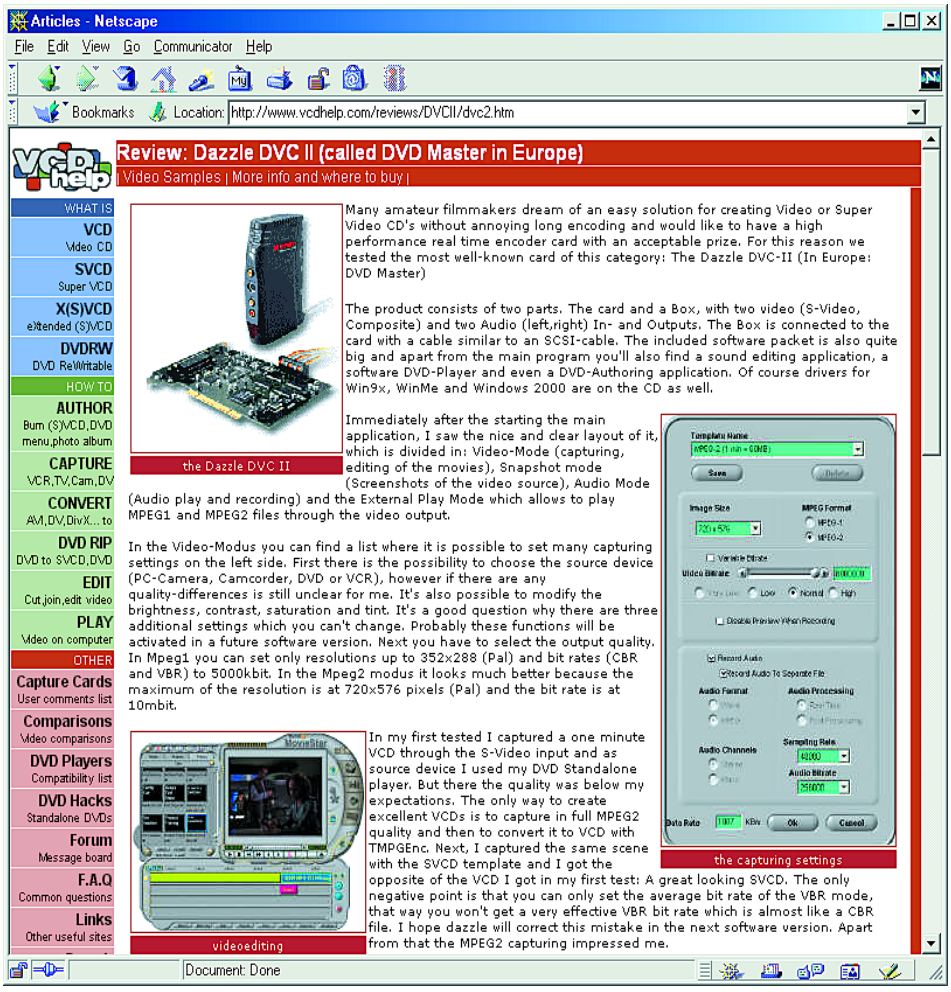
J1	Activation/désactivation du réseau de LED D1 (Visualisation du niveau d'entrée)	
J2	Activation/désactivation du réseau de LED D2 (Visualisation du niveau de sortie)	
J3	Tension d'alimentation positive pour le câble Data-Link :	
	En cas d'utilisation d'un câble Data-Link du commerce : enficher, sinon enlever.	
J4	Niveau de commande de la logique de chargement de l'accu du portable : N'est pas utilisé dans le cas des portables de Siemens	
	Non implanté (High-Z):	Charge rapide avec 5 V, 400 mA
J5	Tension de charge positive pour l'accu du portable : N'est pas utilisé dans le cas des portables de Siemens	
	Enfiché :	à n'enficher que si l'accu doit être chargé par le biais d'un câble Data-Link de fabrication maison.
	Non implanté :	dans tous les autres cas.

Numérisation d'images vidéo

Un site pour tout (savoir)

Harry Baggen

Créer ses propres CD-vidéo est un sujet très populaire dans le monde des possesseurs d'ordinateurs. Qu'il s'agisse de ses propres enregistrements ou de convertir des bandes vidéo ou des DVD au format CD, il n'en reste pas moins qu'il faut un certain niveau de connaissances et de logiciels pour mener cette opération à bonne fin.

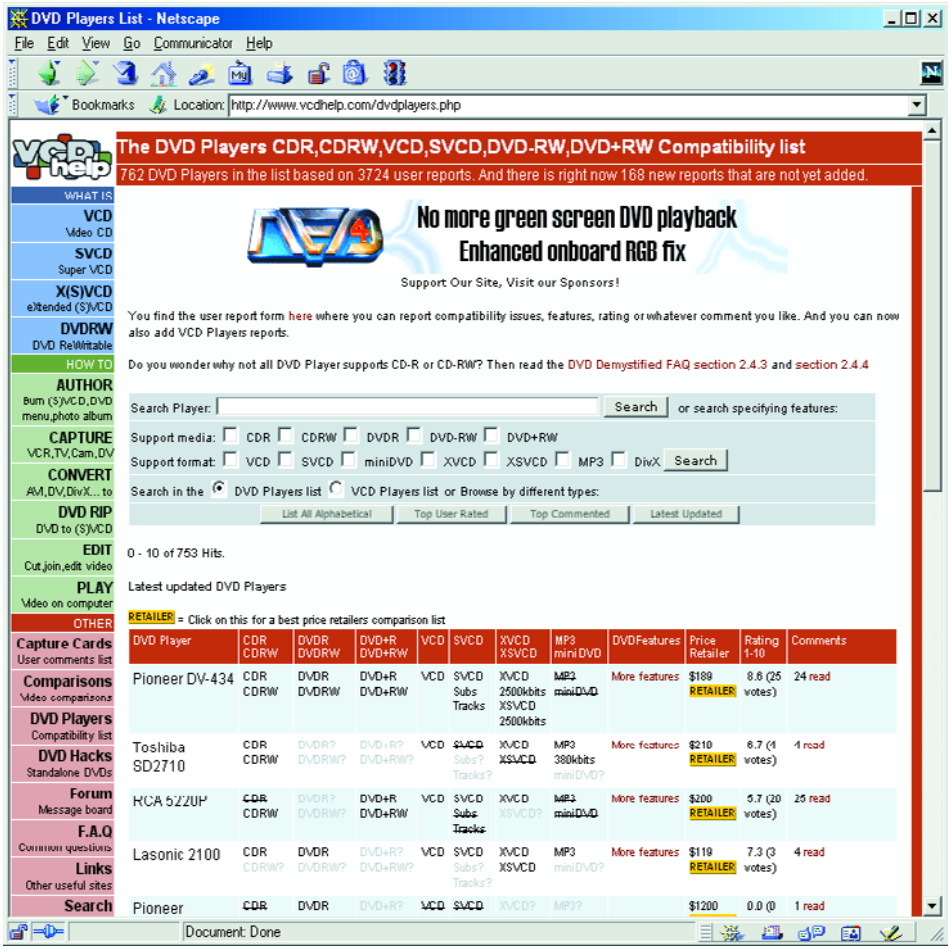


La numérisation de vidéos ou le transfert d'images numériques d'un support à un autre sont des activités très populaires de nos jours ne serait-ce que parce que la grande majorité des possesseurs d'ordinateurs ont une machine rapide dotée d'un graveur de CD. Mais même si l'on possède une machine très rapide la numérisation d'images vidéo analogique ou la conversion d'un format dans un autre peut durer de longues heures, voir plusieurs jours. Pas très rapide tout cela. L'existence d'un certain nombre de format implique l'utilisation de plusieurs programmes différents pour mener toute l'opération à bonne fin. Sans même parler de tous les paramétrages (dont il souvent difficile de saisir la fonction).

Il existe, sur Internet, nombre de sites consacrés à ce sujet. Même les magazines (sérieux) n'hésitent pas à aborder régulièrement ce sujet en lui consacrant des articles copieux. Telle n'est pas notre prétention... Nous nous limiterons ici à signaler un seul et unique site sur lequel on trouve tout ce dont peut rêver tant l'amateur débutant que le (quasi)-professionnel) :

VCDHelp.com.
Comme le suggère sa dénomination, ce site offre des informations et une aide précieuses lors de la création de CD vidéo (VCD = Video CD comme on peut se l'imaginer), mais cela dans le sens le plus large de l'acception de ce terme. Il comporte des sections consacrées aux CD Vidéo (VCD), aux CD Super Vidéo (SVCD), aux CD Vidéo Extended (XVCD) ainsi qu'aux DVD réinscriptibles (DVD RW). Des modes d'emploi exhaustifs décrivent, point par point les étapes à suivre pour créer, à partir d'images vidéo, un VCD, SVCD voire XVCD et quels programmes chaque type d'opération requiert. On y apprend également comment convertir un DVD en VCD, SVCD ou SVCD (nous supposons qu'il s'agit d'une copie d'un DVD légalement en votre possession et strictement limitée à votre usage personnel telle que l'autorise la législation de nombreux pays européens).
Nous le disions plus haut, la numérisation et la conversion de vidéos requiert une puissance de calcul importante. Les tableaux comparatifs indiquant la qualité obtenue et la vitesse de conversion des programmes de conversion les plus connus, de AVI vers MPEG1 ou MPEG2, permettent de se faire une bonne idée des durées de ces opérations.
La liste des programmes de traitement vidéo est extrêmement longue. VCDHelp ne peut pas prétendre disposer de tout mais ce site décrit les plus importants d'entre eux et fournit des liens vers des sites où l'on pourra les trouver.
Ce site propose bien d'autres choses. S'il est dans vos intentions, par exemple, d'acheter une carte de capture vidéo, ce site en décrit un certain nombre et en donne les caractéristiques les plus importantes. On y trouve même les tests d'un certain nombre d'entre elles !
On y trouve également une liste de compatibilité de lecteurs de DVD autonomes qui indique lesquels sont en mesure de lire certains types de CD Vidéo.
On y découvre en outre un panorama des possibilités de modification des lecteurs de DVD pour leur permettre de lire plusieurs codes de région.
La présence d'un forum et d'une section FAQ très riche, les amateurs peuvent échanger ou chercher des informations sur les sujets les plus divers.
Si le sujet ne vous intéresse pas (pour le moment), pourquoi ne pas jeter un coup d'oeil aux nouveautés données sur la page d'accueil. Cela vous permettra de rester au courant de ce qui se trame dans le monde de la vidéo (numérique).

(015108)



Adresse Internet :

www.vcdhelp.com

Commande de feux et de boîte de vitesse

Pour véhicules radiocommandés

Projet : David Dzida

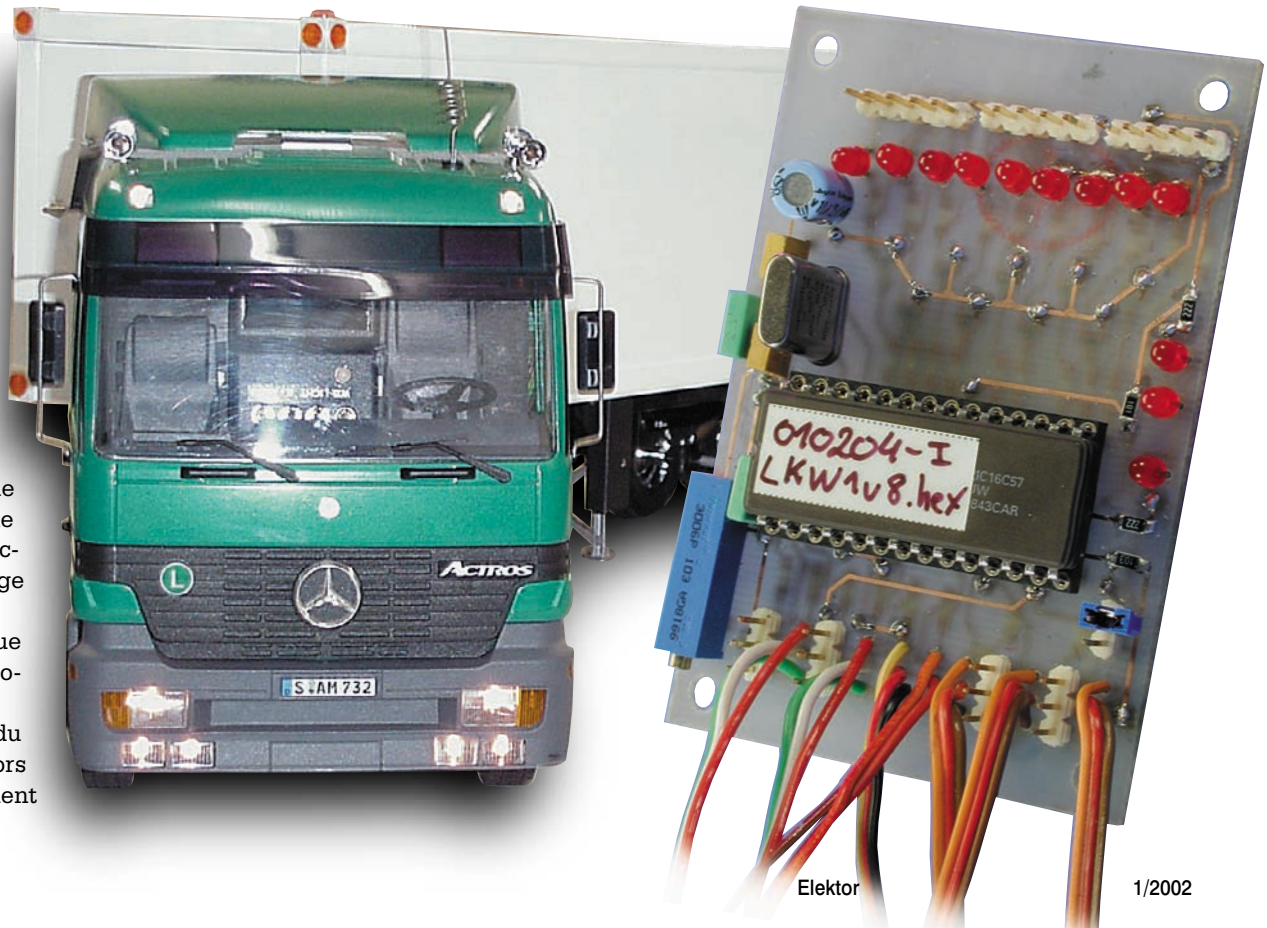
Le montage à base de microcontrôleur PIC proposé ici prend à son compte l'ensemble du pilotage de l'installation des feux et de la boîte de vitesses d'un camion, mais rien n'interdit non plus de l'utiliser pour d'autres « Poids lourds » tels que grues, bulldozers ou autres véhicules similaires.

Le problème majeur auquel on se trouve confronté lorsque l'on se trouve en présence de véhicules télécommandés de ce type est la multiplicité des fonctions qui requiert, normalement, une radiocommande pilotée par micro fort coûteuse. Le projet objet de cet article prend la forme d'une petite platine dotée d'un microcontrôleur de la famille on ne peut plus populaire des PIC de Microchip qui dote le modèle réduit de fonctions additionnelles. Cette extension se fait en partie sous la forme d'une gestion automatique de fonctions et de l'autre par des fonctions de décodage additionnelles.

Passons en revue les fonctions automatiques :

– Mise en route du clignotant lors d'un changement de direction du véhicule,
– Allumage des feux stop lorsque l'on réduit les gaz,
– Allumage des feux de recul lorsque le véhicule se met et roule en marche arrière.

Il est possible, par le biais de commutateurs additionnels sur la télécommande, de mettre en et hors-fonction un certain nombre d'accessoires tels que les feux de croisement, les feux de route, les feux de détresse, les clignotants au niveau de la cabine conducteur, de com-



Caractéristiques techniques :

– Alimentation sous tension de 5 V (par le récepteur)
– Consommation de courant (hors courant d'ampoule) inférieur à 10 mA
– Tension d'alimentation externe entre 5 et 12 V
– Fréquence de clignotement de 0,5 Hz environ
– Déplacement de la servo de boîte réglable par ajustable
– Sens de déplacement (gauche/droite) de la servo d'accouplement de remorque sélectable par JP1
– Courant maximal par sortie 0,3 A
– Sortie d'échantillonnage pour pilotage uniquement (10 mA max.)
– Possibilité de connexion de relais (doté de diode de protection !)

Note concernant le courant d'ampoules maximal :
Si l'on prévoit (cf. figure 5) d'alimenter les ampoules par le biais de la tension d'alimentation interne (récepteur-régulateur BEC) il faudra veiller à ne pas dépasser le courant de sortie maximal du régulateur-BEC !

mander le klaxon et d'autres canaux supplémentaires.
Il est en outre possible, par le biais d'un commutateur ou par l'intermédiaire du manche de commande, de procéder à une commutation séquentielle de la servo de la boîte (de vitesse), dans le sens montant ou descendant (comme sur une Porsche, mais sur 3 vitesses seulement...).
Il reste, pour finir, une fonction pour la remorque, fonction avec laquelle une unique servo-commande assure l'accouplement au tracteur et commande en outre la descente et la remontée des vérins de soutien de la remorque. Cette double fonction est prévue principalement pour les modèles de remorques pour poids lourds de chez Tamiya.

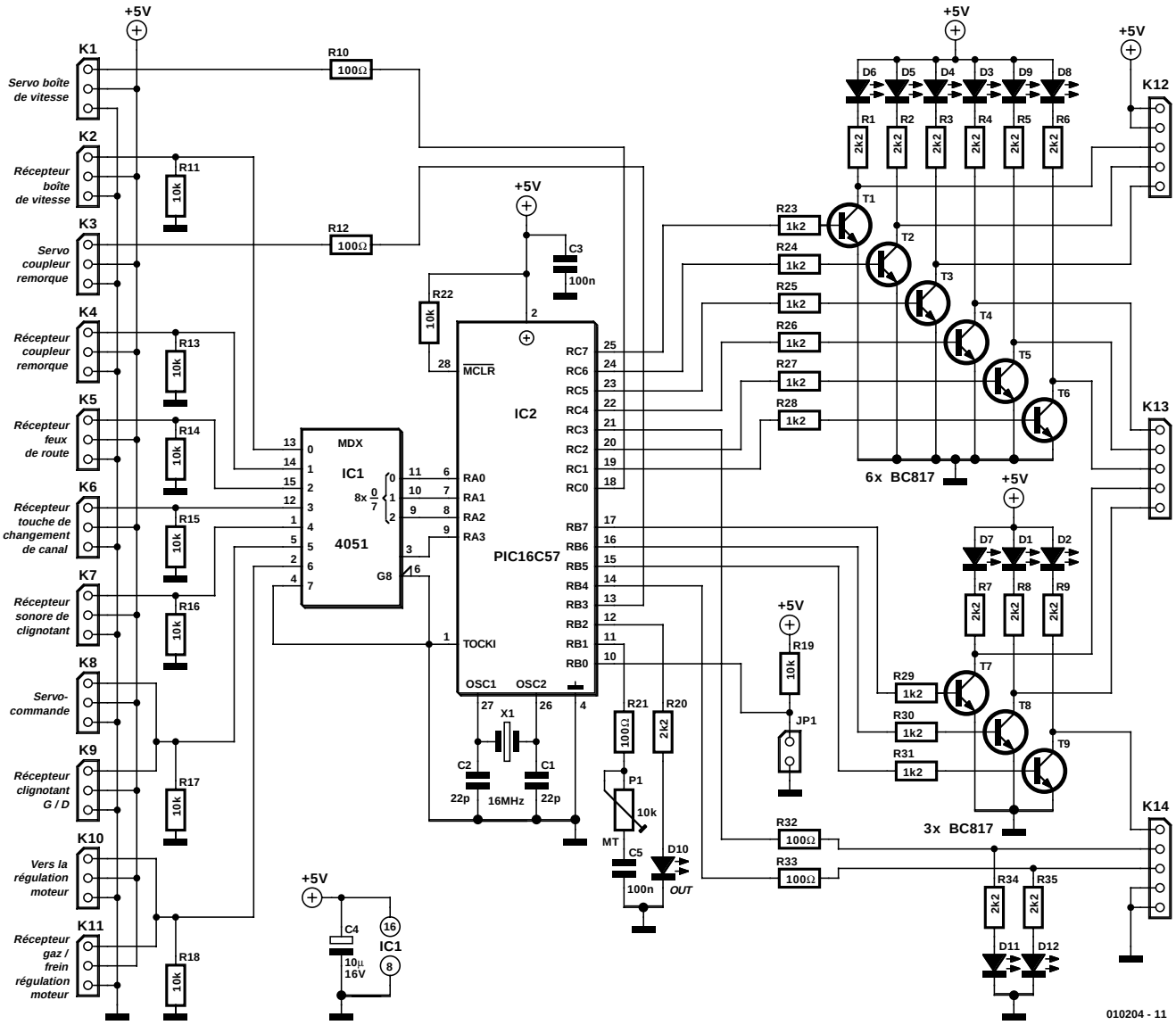


Figure 1. L'électronique de système de pilotage des feux et de la boîte. Le microcontrôleur traite les signaux impulsionnels en provenance du récepteur de la télécommande et génère à partir de ceux-ci les signaux de commande des ampoules et de la boîte de vitesses.

BEC

La signification de l'acronyme « BEC » est tout simplement Battery Elimination Circuit. Il désigne partant un circuit qui supprime la nécessité d'un accumulateur distinct réservé à l'alimentation de la télécommande et qui permet, suite à une préparation adéquate (régulation, découplage), de dériver l'alimentation du récepteur de l'accu d'alimentation du moteur. Certains modèles réduits de poids lourds intègrent le BEC directement dans leur régulateur de moteur (du tracteur). Ces types de régulateurs ont toujours des BEC fournissant une tension de sortie de 5 V. Dans le cas de ces régulateurs spécialement prévu pour les poids lourds, la sortie 5 V accepte une charge sensiblement plus importante que celle supportée par leurs homologues pour modèles réduits légers (voitures). Le courant de sortie atteint, en fonction du type de régulateur concerné, entre 1 et 3 A

Courant d'ampoules

La consommation de courant du montage est, en l'absence d'ampoules, inférieur à 10 mA. Les transistors présents aux 9 sorties peuvent commuter un maximum de 300 mA chacun, soit au total 9 x 0,3 = 2,7 A au maximum ! Seule une alimentation externe (câblée comme le montre la figure 6) peut fournir un courant aussi important. Si l'on se contente d'une tension d'alimentation interne de 5 V (câblage selon le schéma de la figure 5) le régulateur de moteur du véhicule doit fournir à sa sortie BEC 5 V non seulement le courant nécessaire au récepteur de la télécommande et aux servos, mais également l'ensemble du courant requis par les ampoules. Si l'on prévoit d'alimenter un nombre relativement important d'ampoules il faudra opter pour un régulateur capable de fournir un courant de 1,5 A voire plus.

Exemple de choix des ampoules : toutes les ampoules sont du type 5 V/40 mA :

Feux de croisement : 6 x 40 mA =	240 mA
Feux de route : 2 x 80 mA =	160 mA
Feux de stop : 4 x 40 mA =	160 mA
Feux de recul : 2 x 40 mA =	80 mA
Clignotant : 6 x 40 mA =	240 mA
Gyrophare : 2 x 40 ma =	80 mA
Canal de commutation : 1 x 80 mA =	80 mA
Consommation de courant d'ampoules total :	1,04 A

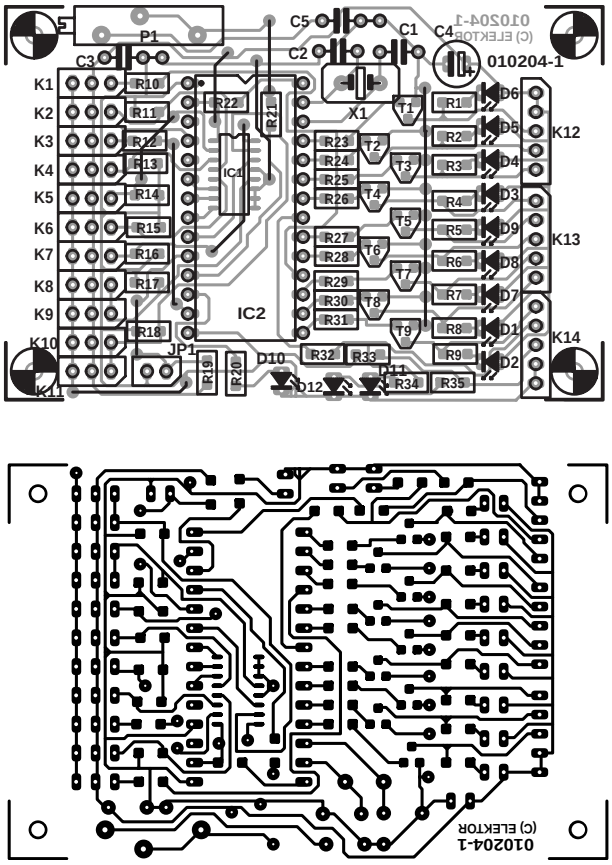


Figure 2. Dessin des pistes et sérigraphie de l'implantation des composants des côtés « pistes » et « composants » de cette platine simple face.

Le montage pourra également être utilisé sur d'autres modèles de poids lourds, où seule la fonction d'accouplement de remorque est requise.

Fonction de commutation

Le coeur du système de commande est un PIC16C57 de l'écurie Microchip tournant à une fréquence d'hor-

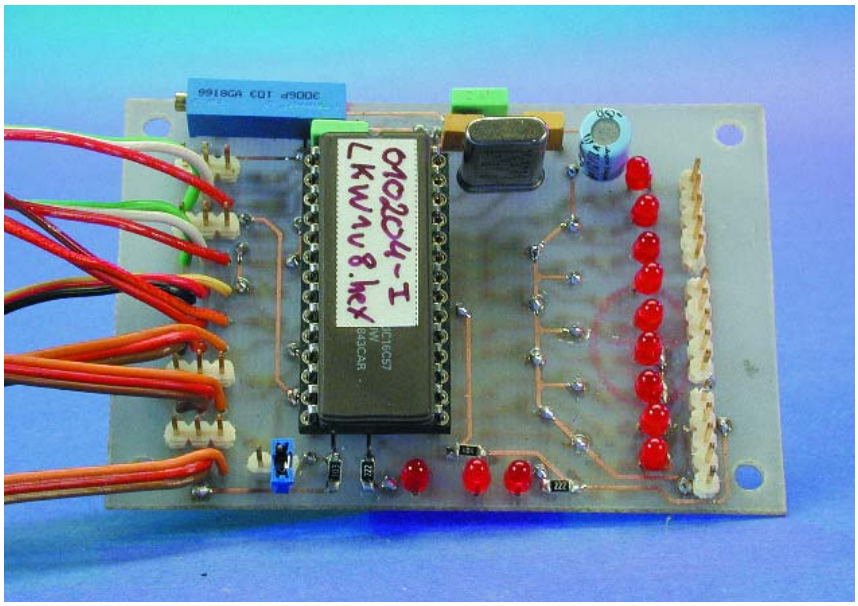


Figure 3. Vue plongeante sur le prototype.

Liste des composants

Résistances :
R1 à R9,R20,R34,R35 = 2kΩ2
R10,R12,R21,R32,R33 = 100 Ω
R11,R13 à R19,R22 = 10 kΩ
R23 à R31 = 1k2Ω
P1 = ajustable 10 kΩ 10 tours horizontal

Condensateurs :
C1,C2 = 22 pF
C3,C5 = 100 nF
C4 = 10 µF/16 V vertical

Semi-conducteurs :
D1 à D9,D11,D12 = LED 3 mm à haut rendement rouge (faible courant)
D10 = LED 3 mm à haut rendement verte (faible courant)
T1 à T9 = BC817
IC1 = 4051 CMS
IC2 = PIC16C57 (programmé **EPS 010204-41**)*

Divers :
JP1 = embase mâle à 1 rangé de 2 contacts + cavalier de court-circuit
K2,K4 à K7,K9,K11 = câble de connexion pour servo-commande
K1,K3,K8,K10 = embase mâle à 1 rangé de 3 contacts
K12 à K14 = embase mâle à 1 rangé de 5 contacts
X1 = quartz 16 MHz

loge de 16 MHz. Nous avons opté pour le modèle en boîtier DIL à 28 broches sachant qu'il facilite une éventuelle réactualisation du programme ou son extension. L'électronique représentée en **figure 1** requiert l'utilisation d'un BEC (circuit éliminateur de pile) fournissant une tension d'alimentation régulée de 5 V. Le microcontrôleur traite les signaux en provenance de 7 canaux de sortie du récepteur de la télécommande. Sur chacune de ces sorties, le récepteur fournit un signal MLI, fameux s'il en est, signal modulé en largeur d'impulsion caractérisé par une largeur d'impulsion comprise entre 1 et 2 ms (1,5 ±0,5 ms). Ces signaux correspondant aux fonctions boîte, accouplement de la remorque, feux de croisement/feux de route, canal de commutation, signal d'alarme, action sur la direction (qui se traduit en fait par un changement d'orientation des roues avant) et moteur (« gaz ») sont connectés respectivement, dans cet ordre, aux embases K2, K4, K5, K6, K7, K9 et K11. Depuis ces connecteurs, ils attaquent les entrées X0 à X6 du multiplexeur IC1 pour aller ensuite aux broches de ports RA0 à RA3 du microcontrôleur PIC IC2. Il aurait été possible, en principe, de se passer de ce multiplexeur CMOS de type 4051, mais cela se serait traduit par l'obligation de passer à un PIC de la taille au-dessus, qui coûte également plus cher.

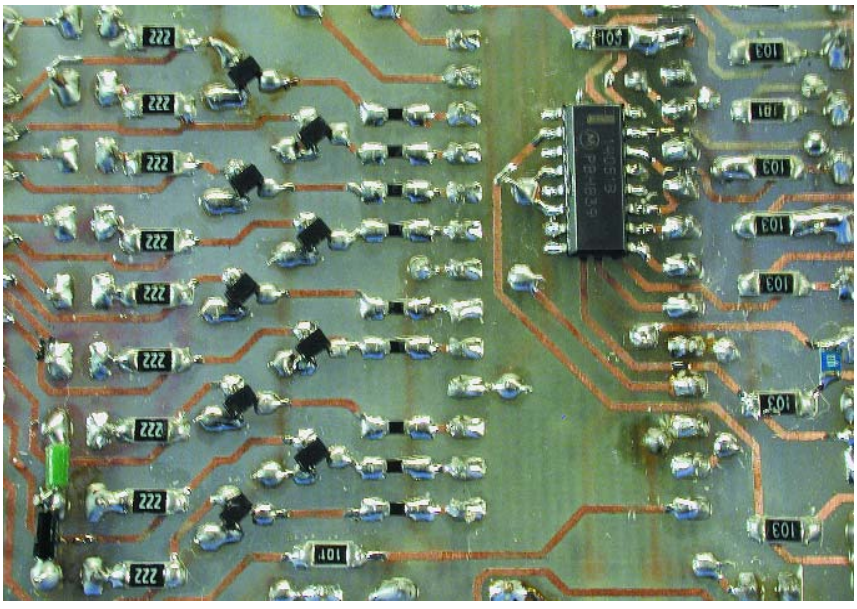


Figure 4. Les composants CMS prennent place sur le dessous du circuit imprimé.

Les signaux des fonctions pilotées directement par la télécommande (moteur, direction) sont transmis non seulement au microcontrôleur mais aussi relayées directement vers des embases de sortie auxquelles sont connectés la servo de direction (K8) et la régulation de vitesse (K10).
La ligne de port RB0 du microcontrôleur est reliée à l'embase (pour cavalier de court-circuit) JP1. Ce cavalier sert à définir le sens de rotation de la servo d'accouplement de remorque. La sortie de port RB1 permet, par le biais de l'ajustable P1, de définir le débattement de la servo de boîte de vitesse.
À la suite du traitement programmé des différents signaux d'entrée, le microcontrôleur fournit des signaux de commande pour 2 sorties de servo-commandes (K1/servo de boîte et K3/servo d'accouplement de remorque) et 11 sorties de commutation. Ces derniers signaux sont disponibles sur les embases K12 à K14. 9 des sorties de commutation prennent la forme de sorties à collecteur ouvert à transistors externes (T1 à T9), 2 broches de sorties (sur K14) sont, elles, reliées directement aux broches de port RC3 et RB4. Le **tableau 1** récapitule les fonctions des différentes broches et leur dénomination. Si le progiciel de traitement de signal stocké dans le microcontrôleur PIC vous intéresse, vous pourrez en télécharger gratuitement le code-source depuis le site Internet d'Elektor (www.elektor.presse.fr).
Chacune des sorties de commutation est dotée d'une LED, D1 à D9 + D11, qui en visualise l'état de sortie. La LED D10 remplit une fonction spécifique. Après l'application de la tension d'alimentation le microcontrôleur commence par calibrer certains signaux, tels que les signaux allant vers le clignotant, les feux de stop et de recul. Il ne faut pas, au cours de ce processus de calibration, toucher partant aux manches de commande. Ce n'est qu'à la fin de ce processus d'initialisation qui dure jusqu'à 4 secondes, s'il s'est fait normalement, que la LED D10 s'allume signalant ainsi que le système est prêt à être utilisé. La LED D12 est reliée à un canal de commutation non utilisé pour le moment (K14/3). De ce fait, cette LED n'a pas, pour le moment, de fonction.
L'auteur a l'intention de réaliser, par le biais de cette sortie, une liaison infrarouge entre le tracteur et la remorque.

Réalisation et mise en oeuvre

L'implantation des composants sur la platine représenté en **figure 2** requiert un minimum d'attention et de soin. Bien qu'il s'agisse

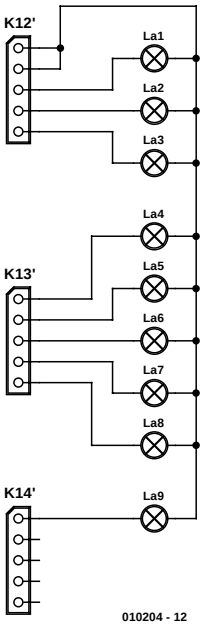


Figure 5. Branchement des ampoules en cas d'alimentation par le biais de la tension d'alimentation interne.

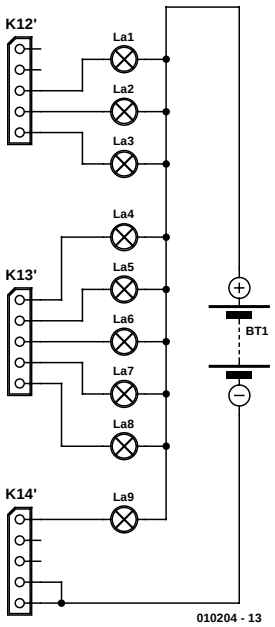


Figure 6. Branchement des ampoules en cas d'alimentation externe sous une tension comprise entre 5 et 12 V.

d'une platine simple face, elle comporte des composants sur ses 2 faces : IC1, un 4051 de type CMS prend place côté « pistes ». L'implantation des transistors T1 à T9 requiert, elle aussi, une attention particulière vu qu'ils prennent place eux aussi côté « pistes ». En

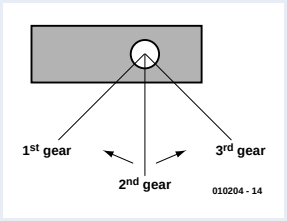
Tableau I. Branchements et fonctions :

Boîte de vitesses à K1/K2

K1 : Branchement pour la servo de boîte de vitesses (pour circuit à 3 vitesses uniquement)
K2 : Borne pour le câble venant de la sortie « Boîte » du récepteur. Fonction sur la télécommande : inverseur à 3 positions « Montée-Rien-Descente »

Fonctionnement :

P1 permet de régler le débattement vers la gauche ou la droite de la servo par rapport au neutre. La valeur donnée à P1 est prise en compte lors de la première nouvelle mise sous tension suivant le changement de réglage. Après l'application de la tension d'alimentation la servo vient toujours en position centrale (seconde vitesse).
Touche bascule 1x Montée = la servo monte d'un cran (passe de la seconde à la troisième vitesse)
Touche bascule 1x Descente = la servo descend d'un cran (passe de la seconde à la troisième vitesse)

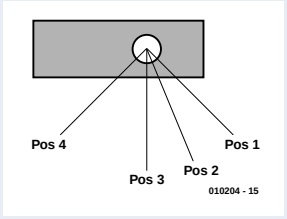


Accouplement remorque K3/K4

K3 : Connexion pour la servo d'accouplement (pour poids lourd Tamiya avec vérins d'étaiement à commande électrique cf. texte)
K4 : Connexion destinée au câble venant de la sortie « accouplement remorque » du récepteur.
Fonction sur la télécommande : inverseur à 3 positions « Montée-Rien-Descente »

Fonctionnement :

Touche bascule 1x Montée = la servo passe en position 1 (commande, via un contact de fin de course de la remontée des vérins d'étaiement de la remorque)
Touche bascule 1x Rien = la servo passe en position 3 (impossibilité, tant que les feux de recul ne sont pas allumés, d'ouvrir le système d'accouplement !)
Touche bascule 1x Descente et feux de recul en fonction = la servo passe en position 2 (commande, via un contact de fin de course de la descente des vérins d'étaiement de la remorque)
Une fois que la touche bascule est relâchée (elle se trouve alors en position centrale) la servo passe en position 4. Le dispositif d'accouplement de la remorque s'ouvre et le tracteur peut se désaccoupler de la remorque. Le système d'accouplement reste ouvert jusqu'à une nouvelle activation vers la position Montée de la touche bascule (remontée des vérins et fermeture du dispositif d'accouplement).



Feux de croisement/feux de route K5

K5 : connexion pour le câble en provenance de la sortie « feux de croisement/de route » du récepteur
Fonction au niveau de la télécommande : interrupteur à bascule à 3 positions « Montée-Rien-Descente »

Fonctionnement

Interrupteur sur Montée = extinction de toutes les ampoules (K12/3/4)
Interrupteur sur Rien = activation des feux de croisement (K12/3)
Interrupteur sur Descente = activation des feux de croisement et des feux de route (K12/3/4)

Canal de commutation et d'échantillonnage K6

K6 : connexion pour le câble en provenance de la sortie « canal de commutation et d'échantillonnage » du récepteur
Fonction au niveau de la télécommande : interrupteur à bascule à 3 positions « Montée-Rien-Descente »

Fonctionnement :

Interrupteur 1 x sur Montée = activation du canal de commutation (K14/1)

Interrupteur à nouveau sur Rien = désactivation du canal de commutation (K14/1)
Interrupteur sur Descente = activation du canal d'échantillonnage (K14/2
Interrupteur sur Rien = désactivation du canal d'échantillonnage (K14/2

Note importante :
Le canal d'échantillonnage n'est utilisable qu'à des fins de commande (10 mA max. !) - ne pas y brancher d'ampoule ou de relais sans étage de commande (driver) adéquat !

Clignotant - Gyrophare K7

K7 : connexion pour le câble en provenance de la sortie « Feux de détresse - Gyrophare » du récepteur
Fonction au niveau de la télécommande : interrupteur à bascule à 3 positions « Montée-Rien-Descente »
Fonctionnement :
Interrupteur 1 x sur Montée = activation des feux de détresse (clignotant G/D (K13/4/5)
Interrupteur à nouveau sur Montée = désactivation des feux de détresse
Interrupteur sur 1 x sur Descente = activation du gyrophare (K13/2/3)
Interrupteur à nouveau sur Descente = désactivation du gyrophare

Clignotant G/D et servo de direction K8/K9

K8 : Branchement de la servo de direction
K9 : connexion pour le câble en provenance de la sortie « Direction » du récepteur
Fonction au niveau de la télécommande : manche de commande de direction
Fonctionnement :
Après application de la tension d'alimentation on a mise en mémoire de la position de neutre du manche de commande – ne pas toucher au manche avant que la LED D10 ne se soit allumée !
Manche vers la gauche = activation du clignotant gauche (K13/5)
Manche au centre = extinction de tous les clignotants
Manche vers la droite = activation du clignotant droite (K13/4)

Feux de stop/de recul et régulateur de moteur K10/K11

K10 : Connexion du régulateur de moteur
K11 : connexion pour le câble en provenance de la sortie « Régulateur de moteur » du récepteur
Fonction au niveau de la télécommande : manche de commande avant/arrière
Fonctionnement :
Après application de la tension d'alimentation on a mise en mémoire de la position de neutre du manche de commande – ne pas toucher au manche avant que la LED D10 ne se soit allumée !
Manche au centre = activation des feux de stop (K12/5)
Manche vers l'avant = extinction des feux de stop/de recul
Manche vers l'arrière = activation des feux de recul (K13/1)
Passage du manche d'une position arrière vers le neutre = activation des feux de stop et de recul.

Adresses Internet :

www.lrp-electronic.de/	régulateurs, chargeurs, moteurs (anglais)
www.graupner.com	catalogue complet (anglais/français)
www.robbe.de	catalogue complet (en anglais)
www.multiplex-rc.de	catalogue complet (en allemand)
www.schulze-elektronik.com/index_uk.htm	régulateurs, chargeurs (anglais)
hwww.hacker-motor.com/english/englisch.html	moteurs (anglais)
www.wedico.de/index_eng.html	modèles de poids lourds (anglais)
www.madrat.de/	Trucs et astuces concernant les servos (allemand)

ce qui concerne l'implantation des composants sur le côté « composants » on commencera de préférence par la mise en place des ponts de câblage. Une note à l'intention des Sherlock Holmes parmi nos lecteurs : il est exact que la disposition des composants de la sérigraphie n'est pas identique à 100% avec celle de la platine utilisée pour les prototypes que l'on voit sur les photos des **figures 3** et **4**. Le dessin des pistes a en effet été modifié quelque peu mais d'un point de vue électrique, les 2 platines dont identiques.
Le tableau 1 fournit toutes les informations importantes en ce qui concerne le brochage, le câblage et la mise en oeuvre des platines, sachant cependant que nous avons consacré les **figures 5** et **6** à la connexion des ampoules. Les sorties RB5 à RB7 et RC0 à RC7 du PIC attaquent directement les ampoules par le biais des transistors BC817. Comme les sorties sont du type à collecteur ouvert, il est possible de piloter les ampoules tant par le biais de la platine de commande (figure 5) qu'au travers d'une tension d'alimentation externe (accu distinct de la figure 6).
Pour cela, on relie, dans chacun des cas, l'une des bornes de l'ampoule à la ligne positive (+) de la tension d'alimentation externe, tension qui doit se situer entre 5 et 12 V. L'important est de veiller à ne pas créer, lors du branchement des ampoules, de court-circuit entre la tension d'alimentation de +5 V sur les broches 1 et 2 de K12 et la tension d'alimentation externe. Un tel court-circuit pourrait entraîner l'endommagement voir la destruction non seulement du PIC mais également de la télécommande !
Si l'on opte pour une alimentation interne (figure 5), cela signifie que la totalité du courant nécessaire à l'alimentation des ampoules devra être fourni par le régulateur de déplacement. Certains régulateurs spécialement conçus pour les véhicules lourds peuvent fournir un courant plus important à la sortie BEC (5 V) que celui qu'est capable de fournir un régulateur pour voiture. Il faudra, quelle que soit la solution adoptée, veiller à dimensionner les ampoules de façon à ce que l'alimentation BEC ne soit pas surchargée (cf. l'encadré « Courant d'ampoule »). On pourra également prendre des relais aux sorties en collecteur ouvert, à condition cependant que ces derniers soient dotés d'une diode de protection (dite de roue libre). La règle définie en ce qui concerne le courant d'ampoule maximal vaut bien évidemment également dans le cas du courant total nécessaire aux différents relais.

(010204)

ISAC (4)

Premières applications

Prof. Dr. Bernd vom Berg, Dipl.-Ing. Peter Groppe & Dipl.-Ing. (FH) Michael Müller-Aulmann

Dans cet article de la série consacrée au projet ISAC nous allons décrire les premières applications réalisées avec ISAC-Cube et vous proposer la version simple de la platine de base, ce qui vous donnera une bonne idée des possibilités d'applications potentielles. Quelques exemples de programmes développés à cette intention et des informations additionnelles (en anglais et en allemand) sont mis à votre disposition sur le site Internet d'Elektor pour un éventuel téléchargement.

Permettez-nous, avant que nous ne passions en revue les différents exemples d'application, de faire quelques remarques préliminaires en ce qui concerne les exemples de programme et les paramètres :

– Les applications et les programmes de démonstration en C51 sont (à une exception près) écrits de manière à pouvoir être traités avec la version limitée de µVision2 (taille maximale de code 2 Koctets, pas d'arithmétique flottante). Il vous faudra partant, si vous envisagez de travailler à des applications plus complexes, soit acquérir la version complète de µVision2, soit utiliser un autre outil pour le 8051.
– De manière à donner quelque chose à « voir » tous les programmes ont soit une fenêtre d'entrée de données utilisateur soit une fenêtre de visualisation pour le moniteur PC. Cela implique de disposer d'un programme de terminal pour le PC. Nous avons supposé que vous avez utilisé Hyper Terminal, sachant que ce programme de terminal accompagne toutes les versions de Windows. Rien ne vous interdit cependant d'utiliser un autre programme de terminal (paramétrage à respecter : 9 600 bauds, 8 bits de données, pas de bit de parité, 1 bit d'arrêt). Il faudra donc, avant de lancer le programme sur le ISAC-Cube, démarrer le programme de terminal et appuyer ensuite sur la touche de remise à zéro (Reset) du ISAC-Cube.
– **Très important** : il vous faudra, si vous avez travaillé avec Hyper Terminal et que vous voulez décharger un nouveau programme vers le ISAC-Cube en vous aidant du Serial Down-Loader, interrompre la liaison établie par le biais de Hyper Terminal (opter pour l'option « *Disconnect* » de Hyper Terminal). En effet, si l'on ne prend pas cette précaution, Hyper terminal continue d'occuper l'interface sérielle (COM) ce qui empêche le Serial Down-

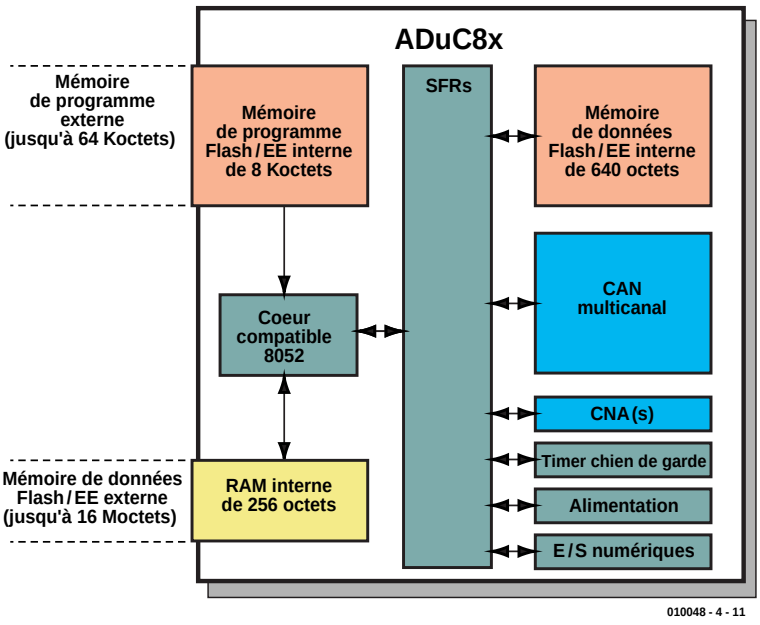


Figure 1. Le concept SFR du microcontrôleur.

Loader d'y accéder lui-même. Cela se traduit par l'apparition d'un message d'erreur « insensé » (dans le sens qu'il n'a pas de sens) pour la simple et bonne raison que le Serial Down-Loader se trouve dans l'incapacité de découvrir un ADuC812 !

Il vous faudra dans ce cas-là, acquiescer le message d'erreur, interrompre la liaison dans Hyper Terminal; vous pourrez ensuite redémarrer le Down-Loader.

La communication entre le coeur du µC et les périphériques embarqués se fait par le biais du concept du SFR (*Special Function Register*) bien connu du 8051 (**figure 1**). Ceux d'entre nos lecteurs qui voudraient de plus amples informations concernant la signification des différents bits des SFR pourrons nous se référer à la fiche de caractéristiques du ADuC812 et à la documentation technique du 8051, tel l'ouvrage proposé en référence [1].

Mise en oeuvre du convertisseur N/A

Mise à disposition en sortie de différents signaux :

Le programme *dac 1.c* permet de faire apparaître en sortie, 2 formes de signal différentes (un signal en dents de scie et un signal rectangulaire) et ce à 2 résolutions différentes (8 et 12 bits). Branchez un oscilloscope aux sorties en question et vérifiez les résultats.

La tension de référence utilisée ici est la tension d'alimentation de la partie numérique VDD (= +5 V), l'excursion des tensions de sortie allant de ce fait de 0 à +5 V.

Utilisation du convertisseur A/N

Saisie des tensions aux canaux d'entrée :

Après le lancement du programme *adc 1.c* il vous est demandé d'entrer le numéro du canal dont il faut effectuer la lecture, en tenant compte du paramétrage suivant :

– Numéro ≡ 0 à 7 : numéro du canal de mesure accessible à l'extérieur correspondant aux convertisseurs ADC 0 à ADC 7.
– Numéro ≡ 8 : canal de mesure relié au capteur de température interne embarqué (*on-chip*), cf. l'exemple ci-après.

Une fois vos données saisies, la tension d'entrée présente au canal choisi est échantillonnée (convertie) une fois par seconde, le résultat de mesure étant affiché sur le moniteur du PC sous la forme d'une valeur décimale et d'une valeur hexadécimale. Une action sur la touche de RAZ (Reset) termine l'opération de conversion; vous pouvez maintenant sélecter un nouveau canal d'entrée.

Attention : le convertisseur A/N possède une résolution de 12 bits et peut être piloté par 2 tensions de référence différentes :

1. Par sa tension de référence interne de 2,5 V. Dans ce cas-là le bit de poids faible (LSB) correspond à un pas de tension de 610 µV.
2. Par le biais d'une tension de référence externe de valeur comprise entre +2,3 et +5 V, que l'on aura connecté à la broche V_{Ref}. Il faudra bien entendu, si l'on opte pour cette solution, veiller à un filtrage et à une régulation corrects (résolution de 12 bits !). Si, par exemple, on applique la tension d'alimentation AVDD (+5 V) à la broche V_{Ref} le bit de poids faible aura une valeur de pas de tension de 1,2 mV.

La tension d'entrée appliquée (c'est-à-dire la tension à mesurer) doit toujours se situer dans la plage comprise entre 0 V et la tension de référence (externe/interne) utilisée, c'est-à-dire entre 0 et +2,5 V en cas d'utilisation de la tension de référence interne ou de 0 à +5 V si l'on utilise une tension de référence externe de +5 V.

Mesure de la température de puce interne et de la température ambiante :

Nous avons parlé en long et en large de ce sujet dans l'annexe prise à la fin de l'article du mois dernier. Nous reprenons son illustration en **figure 2**.

Connexion d'un clavier à une broche d'entrée analogique

Le principe sur lequel repose ce concept de clavier, qui se traduit par la connexion d'un clavier complet à une broche du port d'entrée analogique du convertisseur A/N est celui du « diviseur de tension à étages multiples »

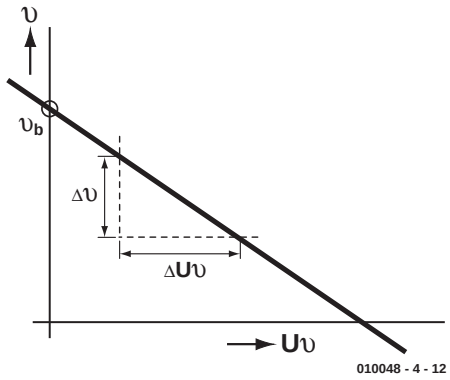


Figure 2. La relation linéaire entre la tension Uϑ et la température ϑ.

Nous commencerons par le diviseur de tension simple tel que représenté en **figure 3**.

Dans le cas de ce circuit on a :

En l'absence d'action sur une touche : U_{ent} = 5 V

Si action sur la touche 0 : U_{ent} = 0 V

Si action sur la touche 1 :

U_{ent} = 5 V * R1/(R0 + R1)

Si l'on donne aux résistances R1 et R0 une même valeur, disons de 1 kΩ par exemple, U_{ent} vaut : 2,5 V.

En fonction de la touche activée on verra apparaître à l'entrée analogique une tension U_{ent} différente parfaitement étagée que le convertisseur A/N est parfaitement en mesure de reconnaître et de traiter. On obtient ainsi, pour chaque touche activée un mot numérique différent qui permet partant une identification indubitable de la touche activée. Cette approche présente un avantage additionnel :

En cas d'action sur plusieurs touches ce sera la touche la plus proche de l'entrée analogique qui sera prise en compte, en d'autres termes : il est extrêmement facile de réaliser un clavier sur lequel les touches sont dotées

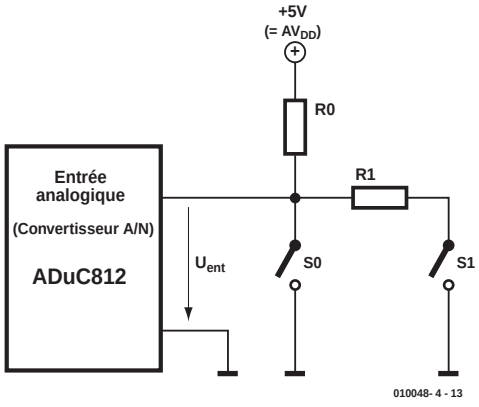


Figure 3. Diviseur de tension piloté par touches pris à l'entrée du convertisseur A/N.

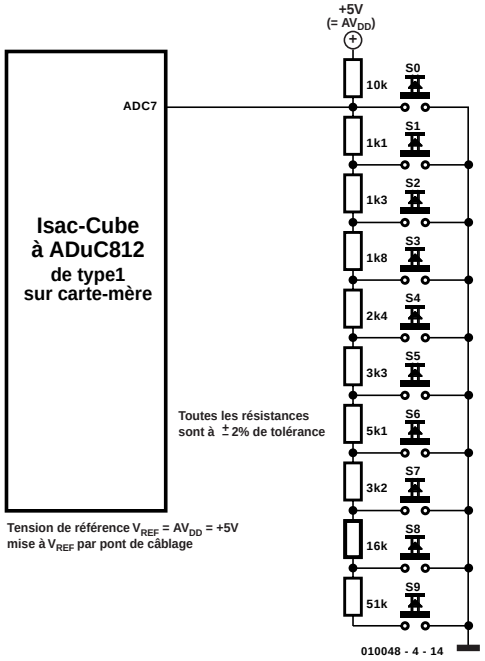


Figure 4. Clavier sériel connecté à une broche du port d'entrée analogique.

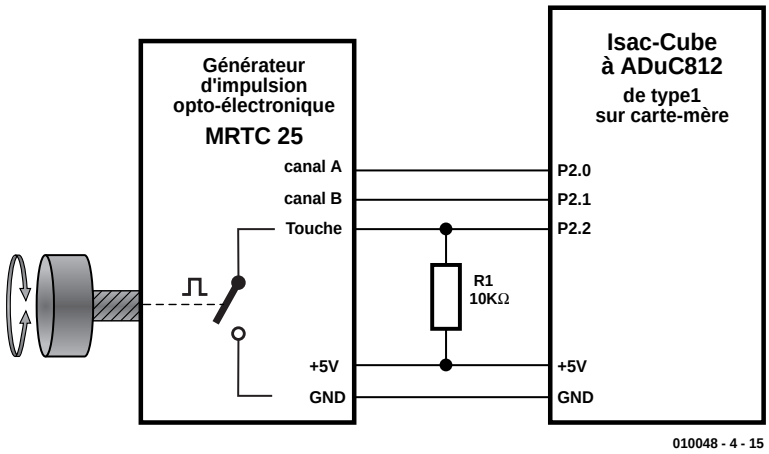


Figure 5. Connexion d'un générateur d'impulsions optoélectronique rotatif.

Tableau 1

Caractéristiques du générateur d'impulsions optoélectronique rotatif du type MRTC25

Caractéristiques électriques :

Résolution (impulsions/360 °)	25
Canaux	2 (A et B; décalés de 90 °)
Tension d'alimentation 5 V CC, ± 0,5 V	
Consommation de courant 20 mA	
Signaux de sortie TTL rectangulaire; pull-up interne de 10 kΩ	
Touche	5 V/10 mA (charge non inductive)
Durée de vie	> 1 000 000 de cycles

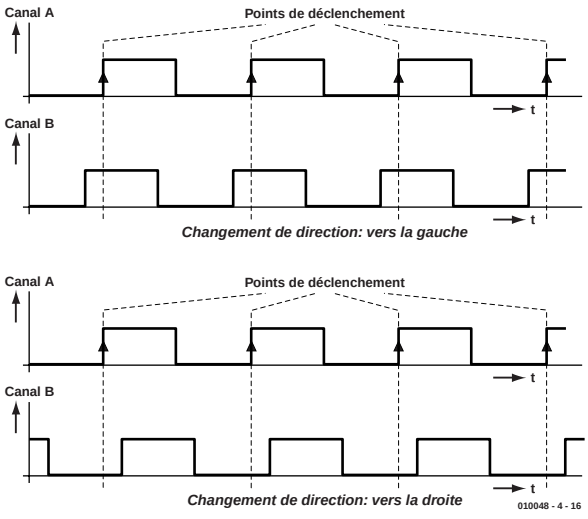


Figure 6. Chronodiagramme des impulsions en fonction du sens de rotation.

d'une priorité différente. Un exemple : si l'on vient d'actionner la touche 1 et que l'on actionne ensuite la touche 0, la tension U_{ent} est instantanément forcée au potentiel correspondant à la touche 0 (0 V), la touche 1 perdant ainsi sa voix au chapitre et n'étant plus reconnue. La touche 0 possède la priorité la plus élevée, de sorte qu'elle sera toujours reconnue en premier. Ce n'est qu'après relâchement de la touche 0 qu'une autre touche pourra être prise en compte.

La condition sine qua non d'un nombre de touches plus important est l'utilisation de résistances à faible tolérance et une alimentation sans parasites du diviseur de tension des touches. Nous vous proposons, en **figure 4**, la solution pratique d'un clavier à 10 touches.

Nous ne pouvons pas ne pas ne pas mentionner un petit inconvénient de cette approche : la reconnaissance des touches se limite naturellement à l'instant de leur activation (et aussi longtemps qu'elles restent enfoncées). Il n'y a pas, en arrière-plan, de mise en mémoire intermédiaire automatique du code des touches. Ce manque est facile à compenser à l'aide d'un programme adéquat.

Nous vous proposons, sous la dénomination *ana tast.c*, un programme fonctionnel permettant l'interrogation d'un clavier à 10 touches connecté à la broche d'entrée analogique ADC7 du ADuC812. Lors du téléchargement de ce programme il vous sera également fourni un tableau donnant la relation entre la touche activée, la tension U_{Ref} correspondante, le mot numérique de conversion central, un intervalle de reconnaissance de touche de longueur suffisante et les niveaux de priorité des touches (0 = priorité la plus élevée).

Mise en oeuvre du port d'E/S numérique

Utilisation d'un générateur d'impulsions optoélectronique rotatif :

Un générateur d'impulsions opto-

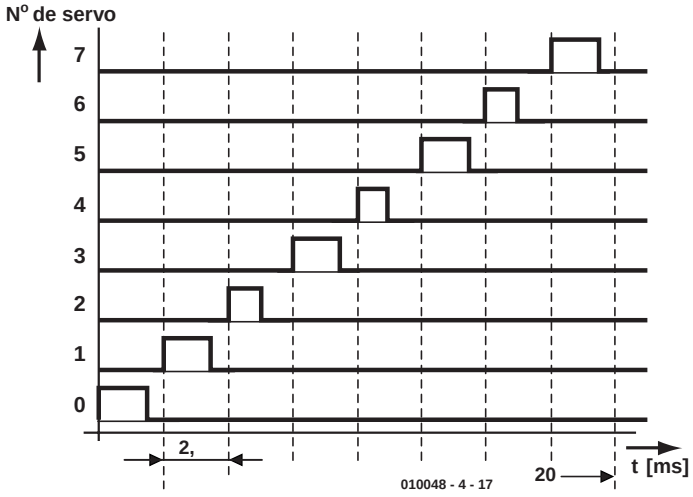


Figure 7. Chronodiagramme pour la commande des servos.

électronique rotatif constitue un organe de saisie tout à la fois très simple et très élégant pour le traitement d'impulsions de comptage et de traitement. Nous avons utilisé ici le modèle MRTC25 de MEGATRON [2]. Il dispose de 2 canaux de traitement (A et B) pour le comptage d'impulsions et la reconnaissance du sens de rotation (droite/gauche). Il comporte en outre un bouton-poussoir qui se fermera lorsque l'on appuie sur l'axe rotatif.
Le schéma de la **figure 5** montre comment le connecter au ISAC-Cube et à la platine de base simple (carte-mère de type I). La **figure 6** donne les chronodiagrammes pour une rotation vers la droite ou vers la gauche. Le **tableau 1** récapitule les caractéristiques électriques du rotacteur. Le programme *drehge 1.c* donne un petit programme utilitaire écrit à l'intention de cet organe de saisie à la fois pratique et intéressant.

Pilotage de 8 servos :
Il est possible, par le biais des 8 sorties numériques du port 2, à l'aide d'un minuscule programme écrit dans le cas présent en assembleur, de piloter, sans le moindre problème 8 moteurs de servocommande. Pour leur pilotage, chaque servo doit se voir appliquer, au cours d'un intervalle de temps de 20 ms, une impulsion d'une longueur comprise entre 0,9 et 2,1 ms (**figure 7**). Dans le cas le plus simple on pourra tout simplement subdiviser l'intervalle de 20 ms en 8 sous-intervalles, et attribuer à chacune des servos, et cela en fonction de la position souhaitée, une des sorties du port 2. Dans le programme SERVO.ASM on utilise, pour générer l'intervalle de 2,5 ms, le temporisateur Timer 2 en mode d'auto-chargement (*auto reload*) et pour la génération des impulsions destinées à chacune des servos le temporisateur numéro 0 câblé en temporisateur à 16 bits.
Intervalle 20 ms/8 = 2,5 ms => Timer 2
Impulsion de servos 0,9 ms – 2,1 s => Timer 0
Dans la routine Timer-2-Interrup-Service on démarre à chaque fois le temporisateur 0 par la largeur d'impulsion d'une servo et on positionne (mise à « 1 ») de la broche correspondante du port 2.
Dans la routine Timer-0-Interrup-Service on remet tout simplement à zéro toutes les broches du port 2 et on arrête le temporisateur.
La routine Timer-2-Interrupt-Service comporte une boucle de temporisation logicielle additionnelle de manière à mettre une chronologie précise à la disposition du programme principal. Il est facile et rapide de tester ce petit programme (figure 8) à l'aide du simulateur ADuC.
Personne ne peut mettre de limite à votre imagination quant aux applications du ADuC812 que vous pourriez envisager. Pour peu que l'on écrive un programme un peu plus complexe il est fort possible de piloter 24 sorties numériques et partant 24 servocommandes. En combinaison avec l'interface I2C on peut même imaginer une solution de remplacement à l'article « interface I²C pour servos » [3], et ce avec une résolution de plus de 1 000 pas.
Il est possible, par le biais des 8 entrées analogiques, de réaliser un pilotage des servos à l'aide de capteurs, raison d'être de cette série, ISAC = *Intelligent Sensor/Actor Controller*, basée sur le MicroConverter AUuC812 d'Analog Devices.

(010048-4)

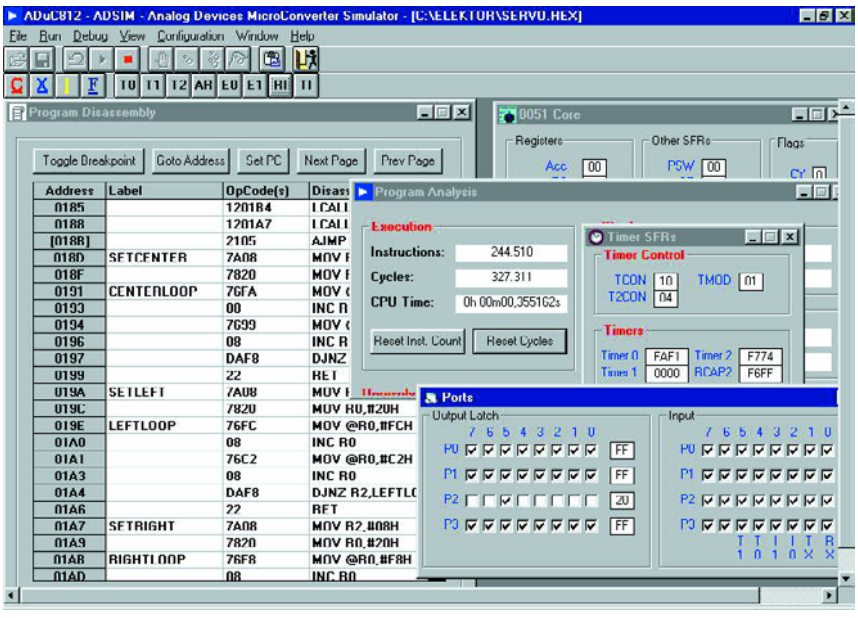


Figure 8. SERVO.HEX au coeur du simulateur ADuC812 (impulsion de sortie à l'intention de la servo 5 en broche P2.5.

Littérature et sources :

[1] Bernd vom Berg, Peter Groppe : « Je programme en Pascal les microcontrôleurs de la famille 8051 (80C537) », Publitronic, ISBN 2-86661-098-9
[2] MEGATRON Composants, Hermann-Oberth-Str. 7, D-85640 Putzbrunn/Munich (RFA), Tél.: (+49) 89/46094-146
[3] Elektor n°279, septembre 2001, page 14 et suivantes

Testeur S/PDIF

Système de test pour l'audio numérique

Projet : Dipl.-Ing. Gregor Kleine

Le petit montage proposé ici pourra être utilisé pour une vérification rapide des sorties d'audio numérique d'une platine CD, d'un lecteur/enregistreur DAT voire d'un lecteur/enregistreur MiniDisc. L'apparition sur le marché d'un nouveau décodeur S/PDIF à CNA (*Convertisseur Numérique/Analogique*) intégré permet la conception d'une réalisation ne requérant qu'un petit nombre de composants connexes.

Nous vous recommandons, si vous n'avez que des notions rudimentaires, voire pas de notion du tout, sur la structure de données d'un signal S/PDIF, de faire un détour par l'encadré qui lui est consacré. Si vous avez passé par cette étape ou que vous en savez suffisamment sur le sujet vous pouvez vous considérer comme un expert en S/PDIF et poursuivre la lecture de cet article.

Le coeur de notre testeur S/PDIF est un nouveau composant de l'écurie de Philips Semiconductors, baptisé du doux nom de UDA1350ATS (voire UDA1351TS pour les taux d'échantillonnage à 96 kHz); il s'agit, comme le dit son fabricant, d'un IEC-958-Audio-DAC, un CNA audio répondant aux normes IEC-958. La **figure 1** en donne le synoptique interne. On y découvre un décodeur IEC-958 et un convertisseur N/A stéréo intégré, de sorte que l'on dispose en sortie d'un signal analogique. Le composant peut être piloté sériellement par un microcontrôleur par le biais d'une interface dite de type L3, mais connaît également un fonctionnement statique aux possibilités plus limitées dès lors que son entrée SELSTATIC est forcée au niveau haut.

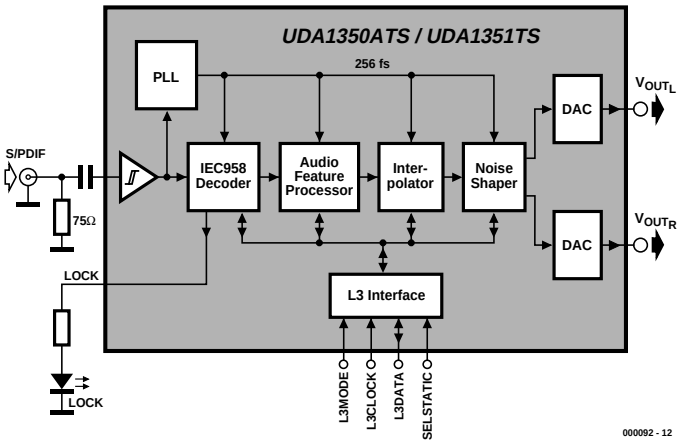


Figure 1. Synoptique de l'UDA 1350/1.

Tableau 1.

Caractéristiques techniques des UDA1350ATS et UDA1351TS

Paramètre		Valeur
Plage des tensions d'alimentation		+2,7 à +3,6 V
Dissipation de puissance		80 mW @ 48 kHz
		110 mW @ 96 kHz
V_{ref}		0,45. à 0,55 V_{DDA}
Tension d'entrée		0,2 à 3,3 V_{ss}
Hystérésis de la tension d'entrée		40 mV
Fréquences d'échantillonnage possibles		
	pour le UDA1350ATS	28 kHz à 54 kHz
	pour le UDA1351TS	28 kHz à 100 kHz
Tension de sortie		900 mV_{eff}
Rapport signal/bruit		100 dB typique
Diaphonie (séparation des canaux)		96 dB typique
Différence entre les niveaux de sortie		0,1 dB typique

Structure de données du signal S/PDIF

Historique

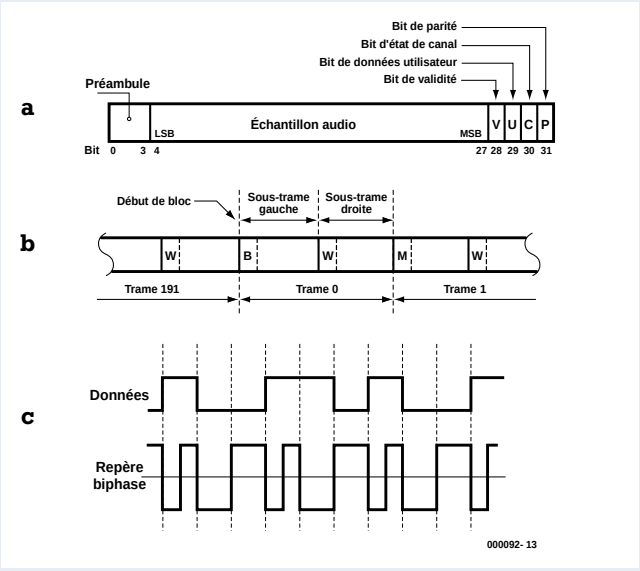
Les interfaces d'audio numérique pour équipements tels que lecteurs de CD ou lecteurs/enregistreurs DAT virent le jour sous la dénomination AES/EBU (*Audio Engineering Society*, *European Broadcasting Union*), sachant qu'elles furent spécifiées, tout au début, pour les émetteurs radio. On a normé, dans le document de l'EBU numéro 3250 de novembre 1985, une interface travaillant à une fréquence d'échantillonnage de 48 kHz (voire 32 kHz) et utilisant 24 bits audio par canal. Avec la multiplication des lecteurs de CD, il fut temps de définir une version grand public de la norme AES/EBU EBU, norme définie par Sony et Philips, d'où sa dénomination de Sony/Philips Digital Interface (S/PDIF). L'interface professionnelle AES/EBU EBU et l'interface grand public S/PDIF furent ultérieurement réunies dans le standard IEC-958 (IEC = *International Electrotechnical Commission*). Les différences majeures entre l'interface professionnelle et son homologue S/PDIF ne se situe pas au niveau du codage des données audio, mais, entre autres, au niveau de la transmission de données additionnelles dans le bloc d'état de canal (*Channel Status*). Le premier bit de ce bloc indique si l'on se trouve en présence d'une structure professionnelle ou S/PDIF. La structure des trames et le codage des données audio est parfaitement identique dans les 2 systèmes.

Sous-trames

Le flux de données d'une interface IEC-958 se subdivise en sous-trames (*subframe*), trames (*frame*) et blocs (*block*). Une sous-trame comporte une seule valeur d'échantillonnage du signal audio codée sur une longueur de 32 bits. Les 4 premiers bits (bit 0 à bit 3) contiennent un mot de synchronisation appelé préambule. Il existe, à ce niveau, 3 modèles de bits : le préambule B identifie le début de bloc, le bloc débutant toujours par la valeur d'échantillonnage du canal gauche. Le préambule M désigne une sous-trame standard du canal audio gauche. Il reste le préambule W qui concerne le canal droit.
Les 24 bits suivants contiennent l'information audio. Si l'on a besoin de moins de 24 bits les premiers bits contiennent des zéros (pour 16 bits, les bits 4 à 11 seront à zéro, les bits 12 à 27 contenant la valeur d'échantillonnage). La sous-trame est clôturée par les bits indicateur de validité (*Validity Flag*), données utilisateur (*User Data*), état de canal (*Channel Status*) et de parité (*Parity Bit*). L'indicateur de validité permet de marquer des valeurs d'échantillonnage invalides qu'il n'est pas question d'utiliser lors du traitement du signal en aval. Le bit « donnée utilisateur » transmet, à raison de 1 bit dans chaque sous-trame, des informations additionnelles telles que du texte. Le bit « état de canal » comporte, à raison de 1 bit par sous-trame lui aussi, des informations additionnelles concernant le trajet de transmission. On y trouve, par exemple, des informations de taux d'échantillonnage, de mode audio ou de mode de données, de mode professionnel ou de mode grand public. En queue de peloton un bit de parité sécurise la transmission des données permettant la détection d'erreurs de transmission. Il n'y a pas de correction d'erreur sachant qu'il est possible de camoufler par interpolation les échantillons audio erronés.

Trames et blocs

La trame qui se situe au niveau au-dessus comporte autant de sous-trames qu'il y a de canaux audio. Dans le cas de signaux stéréo classique une trame comportera partant 2 sous-trames (canaux gauche et droit). Les trames doivent être transmises à la fréquence d'échantillonnage. Dans le cas d'un lecteur de CD travaillant à une fréquence d'échantillonnage de 44,1 kHz et en mode stéréo à 2 canaux cela se traduit par un taux de données de :
2 canaux x 32 bits par sous-trame x 44,1 kHz = 2,822 4 Mbits/s.
On trouve, au niveau au-delà des trames, le bloc, un ensemble de 192 trames, ce qui signifie que toutes les 192 valeurs d'échantillonnage on a un nouveau bloc. Le bloc n'a pas de signification en ce qui concerne les données audio, mais il définit une structure pour les informations additionnelles transmises par le biais des bits de donnée utilisateur et d'état de canal. En marquant le début de bloc à l'aide d'un préambule B il devient possible de décoder sans erreur les informations véhiculées à raison de 1 bit par sous-trame. Ensemble, ils forment ainsi un champ de 384 bits (2 x 192 en stéréo). Comme nous de disions plus haut, la structure au niveau de l'état de canal entre l'interface professionnelle et l'interface S/PDIF présente une différence sensible. Le premier bit indique s'il s'agit du format professionnel ou du format S/PDIF.



Format de données du signal S/PDIF
a) Sous-trame
b) Trame et bloc
c) Signal de repère biphase

Signal biphase

D'un point de vue électrique, l'interface S/PDIF prend la forme d'un câble coaxial de 75 Ω doté à chacune de ses extrémités, source et drain, par une fiche Cinch. La source fournit typiquement une tension de signal de 500 mV_{cc}. Le drain doit avoir une sensibilité d'au moins 200 mV_{cc} pour être en mesure de travailler avec un câble d'une longueur supérieure à 10 mètres. La transmission se fait en code biphase qui utilise, pour chaque « 1 » du flux de données, un double changement de polarité et un seul changement de polarité lorsqu'il s'agit d'un « 0 ». Ce signal est libre de potentiel continu de sorte que l'on peut utiliser un couplage par condensateur.
L'interface professionnelle IEC-958 travaille à d'autres niveaux de signal (3 à 10 V_{cc}), impédance de câble (110 Ω) et requiert des connecteurs symétriques. La source ou le drain doit comporter un dispositif de transmission, ceci pour éliminer tout risque de boucle de masse.

Dans ce mode, un certain nombre de fonctions telles que réglage du volume, des graves, des aigus, des paramétrages de filtres BF, le silencieux et la commande externe de la désaccentuation (*deemphasis*) ne sont plus disponibles. Cependant, si l'on veut uniquement utiliser ce composant pour tester une sortie d'audio numérique, on peut fort bien le mettre en oeuvre en mode autonome.

Tableau 2.

Fréquences d'échantillonnage et taux de données correspondants.

UDA1351TS	UDA1350ATS	Fréquence d'échantillonnage	Taux de données
X	X	32,0 kHz	2,048 Mbit/s
X	X	44,1 kHz	2,8224 Mbit/s
X	X	48,0 kHz	3,072 Mbit/s
X		64,0 kHz	4,096 Mbit/s
X		88,2 kHz	5,6448 Mbit/s
X		96,0 kHz	6,144 Mbit/s

Traitement des données par le UDA1350/1

Le cheminement des données au coeur du UDA1350/1 va de l'entrée S/PDIF au décodeur IEC-958 et à système d'horloge interne (PLL) en passant par un amplificateur intégré chargé d'amener le signal d'entrée à un niveau CMOS. On trouve en aval un processeur de caractéristique audio (un AFP pour *Audio Feature Processor* comme disent les anglophones) chargé de réaliser la désaccentuation lorsque l'on se trouve en mode statique comme cela est le cas ici.

La désaccentuation consiste à atténuer le niveau des fréquences audio les plus élevées en vue de réduire le bruit et partant d'améliorer le rapport signal/bruit. Pour ce faire, côté émetteur, c'est-à-dire sur le support d'enregistrement (CD, DAT, et autre) on rehausse le niveau de la partie supérieure du spectre des fréquences, processus appelé préaccentuation. Tous comptes faits, la courbe de réponse en fréquence est retrouvée, la somme de la préaccentuation et la désaccentuation étant nulle. Dans l'interpolateur qui se trouve à la sortie de l'AFP on procède, pour préparer la conversion N/A, à une multiplication par 128 de la fréquence d'échantillonnage. On utilise pour ce faire un filtre numérique récursif (IIR) suivi d'un filtre numérique FIR. Ce filtre atténue de 50 dB toutes les composantes de signal de fréquence supérieure à la moitié de la fréquence d'échantillonnage. Dans le dernier bloc de la chaîne, celui de la « mise en forme de bruit » (*noise shaper*), on effectue un déplacement du bruit de quantification situé dans la plage des fréquences utiles (10 Hz à 25 kHz) vers le haut de manière à le faire sortir de la plage des fréquences audio, ce qui a pour effet de rehausser le rapport signal/bruit dans la plage des fréquences audio. Le « metteur en forme » convertit en outre les valeurs d'échantillonnage arrivant sous forme parallèle en un flux de données de 1 bit (un train pour le canal gauche un autre pour le canal droit). Le convertisseur N/A intégré dans le UDA1350/1 prend la forme d'un filtre de reconstruction quasi-numérique qui transforme le flux de données de 1 bit en provenance du « formateur de bruit », par filtrage passe-bas, en un signal analogique. De par ce principe de conversion N/A il n'est pas nécessaire de prévoir un filtre passe-bas analogique externe. Le signal de sortie du circuit possède un niveau suffisant pour attaquer l'entrée Ligne (*Line*) d'un amplificateur stéréo.

Dans le cas de notre testeur S/PDIF, la sortie de signal audio sert en fait uniquement à attaquer directement un casque d'écoute stéréo.

Le décodeur IEC-985 dérive du flux de données les échantillons audio de 24 bits pour les signaux audio gauche et droit et les bits d'état du canal central. Il reconnaît le paramétrage de désaccentuation, la fréquence d'échantillonnage, le mode de fonctionnement MCI (PCM = *Pulse Coded Modulation* = **M**odulation par **C**odage d'**I**mpulsion) bicanal ainsi que la précision de l'hor-

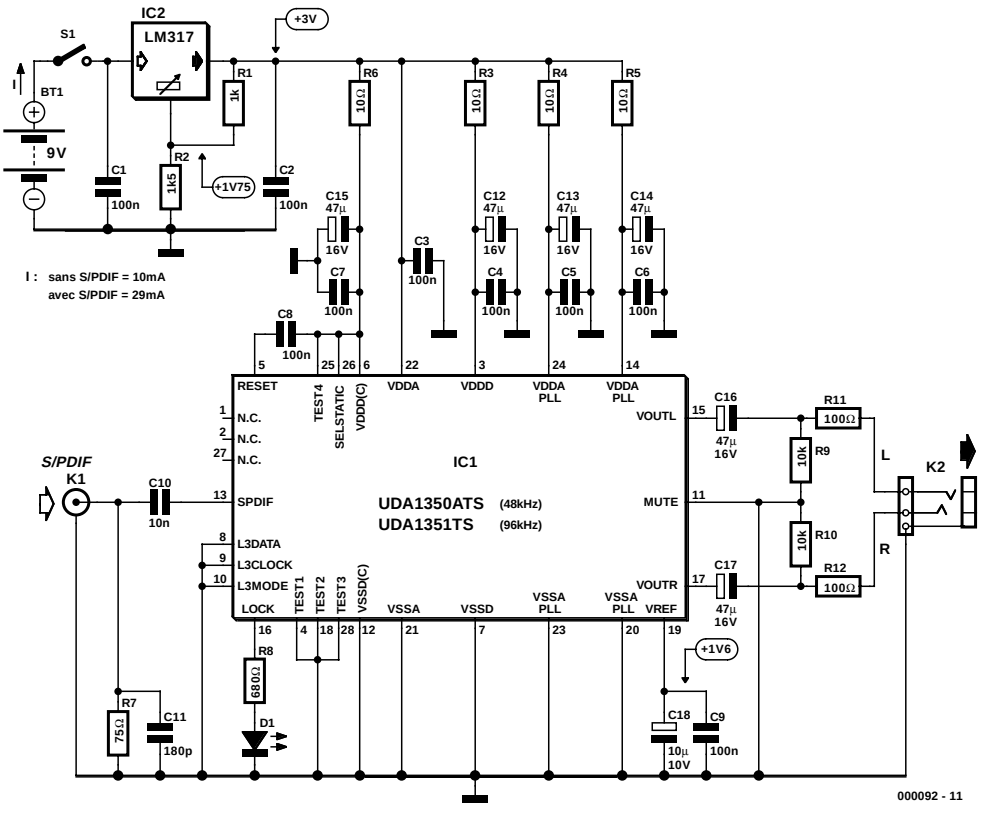


Figure 2. L'électronique du testeur de signal S/PDIF

loge. Le traitement additionnel effectué au niveau de l'interpolateur se fait sous la forme d'un signal de 20 bits.
Une boucle à verrouillage de phase (PLL = *Phase Locked Loop*) interne permet de synchroniser sur des signaux dont la fréquence d'échantillonnage est comprise entre 24 et 54 kHz. Les taux d'échantillonnage courants, à savoir 32, 44,1 et 48 kHz, sont partant couverts.
Si l'on implante dans le testeur S/PDIF non pas un UDA1350ATS, mais un UDA1351TS compatible broche à broche avec le premier la boucle à verrouillage de phase permet même de synchroniser à des taux d'échantillonnage allant jusqu'à 100 kHz. La LED prise à la sortie LOCK signale lorsque le décodeur IEC 958 s'est synchronisé sur le flux de données en entrée. Si le décodeur se trouve dans l'impossibilité de déchiffrer le flux de données d'entrée, la sortie audio est basculée en mode silencieux.
Le **tableau 2** récapitule les fréquences d'échantillonnage classiques et donne les taux de données correspondants. Ce tableau nous apprend quels taux d'échantillonnage chacun des 2 types de UDA135X est capable de traiter.

Une électronique compacte

Le UDA1350ATS et le UDA1351TS travaillent à une tension nominale de +3,0 V. L'électronique dont le schéma est donné en **figure 2** ne consomme, entrée en l'air, que de l'ordre de 10 mA, ce courant passant à moins de 30 mA en présence d'un signal S/PDIF. De manière à permettre au testeur S/PDIF d'être alimenté à partir d'une pile compacte de 9 V, nous avons intégré un régulateur linéaire du type LM317 paramétré à +3,0 V.
Les différents sous-ensembles intégrés dans le composant (PLL, convertisseur N/A, etc.) reçoivent leur courant par le biais de connexions d'alimentation propres.
Le montage est doté, sous la forme d'une triplette constituée d'une résistance de 10 Ω, d'un condensateur électrochimique de 47 µF et d'un condensateur de 100 nF, de réseaux RC de découplage ayant pour fonction de filtrer tout parasite produit par les différents domaines numériques du composant ou risquant d'en gêner le fonctionnement. On évite ainsi que les convertisseurs N/A ne captent de parasites par le biais de leur tension d'alimentation et ne puissent en transmettre.
C8 fait office de condensateur de charge pour l'électronique de remise à zéro chargée de l'initialisation du circuit intégré lors de l'application de la tension d'alimentation. On dispose, pour le convertisseur N/A, en broche 19, de la tension de référence V_{ref}, tension découplée à l'aide d'un condensateur électrochimique et d'un condensateur de 100 nF. La valeur de Vref est proche de la moitié de la tension d'alimentation, soit de l'ordre de +1,5 V.
La ligne de transmission qui véhicule le signal audio numérique (S/PDIF) possède une impédance de 75 Ω et requiert partant d'être close à l'aide d'une résistance de terminaison de ligne, R7, pour éviter toute distorsion du signal par réflexions. C11 court-circuite les signaux parasites de fréquence élevée (HF) que pourrait capter le câble. Le signal d'entrée arrive, par le biais de C10, un condensateur de 10 nF, au décodeur mono-puce. Il faut à ce niveau que le signal ait une valeur crête à crête comprise dans la plage de tension allant de 0,2 à 3,3 V.
La broche Lock commande, au travers de la résistance R8 la LED D1 chargée de visualiser un décodeur IEC-958 verrouillé. Elle s'allumera en fait uniquement en présence d'un signal d'audio PCM. En présence de flux de données qui ne correspondent pas à un signal audio respectant les normes cette LED ne s'allume pas, même si le décodeur est verrouillé.
La broche de silencieux (Mute, broche 11) est, dans le cas du testeur S/PDIF, forcée en permanence au potentiel de la masse, de manière à éviter toute « mise au silence » du signal de sortie. Dans ces conditions, le circuit intégré transmet toujours vers la sortie, et cela sans retard, tous les signaux audio décodables. En l'absence de signaux de donnée ou en présence de signaux non décodables, c'est un circuit de silencieux qui entre en fonction en vue d'éliminer tout risque de production de signaux parasites de fort niveau que pourraient produire des flux de données indéfinies appliquées au convertisseur N/A.
Les 2 canaux sont couplés à l'embase de sortie K2 par l'intermédiaire de condensateurs électrochimiques de 47 µF, C16 et C17. Les résistances de 10 kΩ forçant les sorties à la masse définissent le potentiel en continu de la sortie en l'absence de casque d'écoute. Les résistances série de 100 Ω protègent les sorties du circuit intégré contre une éventuelle mise en court-circuit de la ligne de signal BF.

Implantation des composants et mise en service

Le UDA1350ATS tout comme le UDA1351TS d'ailleurs, est proposé en boîtier SSOP28 (*Shrink Small Outline Package* à 8 broches). Nous n'avons pas dessiné de platine à l'intention de cette réalisation ce qui ne signifie pas non plus qu'il vous faudra impérativement faire appel à votre logiciel de CAO préféré pour dessiner votre propre platine. En effet, vu le faible nombre de composants requis, ce montage pourra fort bien prendre place sur un morceau de platine d'expérimentation à pastilles prévue pour la mise en place de composants CMS (telle que, par exemple, la platine de Conrad proposée sous la référence 527858).
On commencera par la mise en place du régulateur de tension et on vérifiera qu'il fournit bien les 3,0 V prévus. Ce ne sera qu'ensuite que l'on poursuivra la réalisation du montage par la mise en place du UDA1350/1.
Une fois la réalisation physique terminée, on prendra le temps de vérifier les soudures à la loupe. Après s'être assuré qu'aucune des broches de IC1 n'est en court-circuit on appliquera à l'entrée K1, par le biais d'un câble de 75 Ω doté d'un connecteur Cinch, un signal d'audio numérique. L'alimentation prendra la forme d'une pile compacte de 9 V, voire, mieux encore, une alimentation de laboratoire dotée d'une limitation de courant ajustable entre 50 et 100 mA. On devrait entendre alors un signal BF de niveau normal dans les coquilles du casque branché sur l'embase K1. Il vous faudra, si cela n'est pas le cas, revérifier les soudures du UDA1350/1. N'y aurait-il pas à un endroit ou à un autre un court-circuit quasi-indiscernable ? Vérifiez la présence des différentes tensions d'alimentation aux broches correspondantes du UDA1350/1. On doit trouver partout 3,0 V. Si vous mesurez une valeur plus faible la possibilité d'un court-circuit n'en devient que plus probable. V_{ref} se trouve-t-elle bien à 1,6 V ? Si tel est également le cas, il faudra peut-être envisager d'utiliser un autre signal S/PDIF.

(000092)

Cours «Microcontrôleur»

Partie 1 : l'assembleur TASM

Burkhard Kainka

Ce cours, subdivisé en plusieurs parties, s'adresse à tous ceux d'entre nos lecteurs qui ont toujours voulu savoir comment travaille un microcontrôleur et comment s'en servir, mais qui n'ont jamais eu le courage de le demander. Nous allons prendre les choses tout au début. Notre plate-forme de travail sera la toute nouvelle carte à 89S8252 Flash présentée le mois dernier.

Plus personne ne s'étonne aujourd'hui de travailler avec des ordinateurs et dans la plupart des cas avec des machines extrêmement puissantes. Le coeur d'un ordinateur est son processeur qui peut être, par exemple, un Pentium III. Un microcontrôleur est à la fois bien moins et bien plus que le processeur d'un PC typique. Bien moins parce qu'il traite des programmes de taille bien moindre, requiert moins de mémoire et bien souvent travaille bien plus lentement que son grand-frère. Bien plus parce qu'un microcontrôleur intègre déjà sur la même puce un certain nombre de choses que l'on trouve éparpillées sur la carte-mère du PC, à savoir de la mémoire de travail, un (ou plusieurs) temporisateurs, des interfaces et des lignes de port. L'aspect quasi-magique d'un microcontrôleur est qu'il est possible, dans le cas extrême, de remplir une tâche complexe à l'aide d'un seul et unique circuit intégré. Il est possible, à tout un chacun, de se fabriquer ce circuit intégré « spécial » qu'il a toujours souhaité en le programmant tout simplement, et tout cela à un coût, comparativement, très abordable.

Un microcontrôleur est en quelque sorte une sorte de circuit logique doté de nombreuses entrées et sorties. C'est le programme qui détermine la fonction de cette électronique. Faut-il réaliser un compteur numérique ou s'agit-il de faire un chronomètre ? Voulez-vous créer une porte logique très spéciale ou voudriez-vous plutôt réaliser un générateur d'impulsions universel ? A-t-on l'intention de décoder un signal de données complexe ou veut-on piloter un circuit numérique. On pourra, pour toutes ces applications, demander l'aide d'un microcontrôleur. La palette des applications potentielles ne cesse de s'étoffer; si, voici quelques années à peine, il aurait fallu une quantité industrielle de cir-

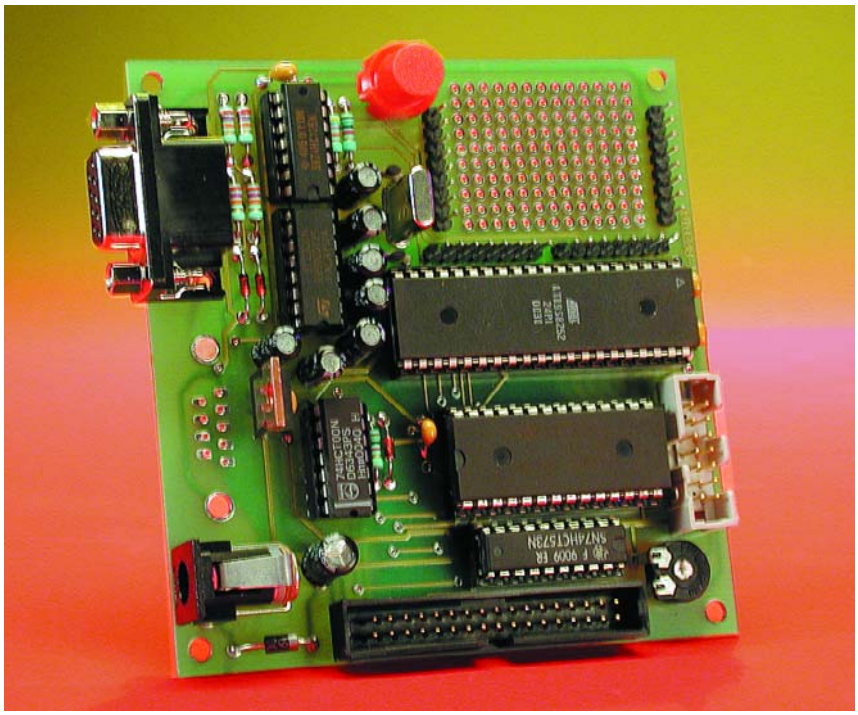


Figure 1. La carte à 89S8252 Flash utilisée comme base de ce cours est également un système à microcontrôleur utilisable pour de nombreuses applications.

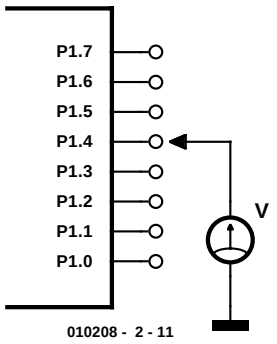


Figure 2. Nous contrôlons le résultat de nos premiers essais à l'aide d'un voltmètre.

cuits intégrés pour réaliser une application, il est possible, aujourd'hui, de lui donner corps à l'aide d'un seul et unique circuit intégré, un microcontrôleur en l'occurrence. Il vaut donc la peine d'en apprendre un peu plus en ce qui concerne la programmation. On peut envisager plusieurs approches.
Comme nous le disions plus haut, ce cours utilise, comme matériel, la carte à 89S8252 Flash décrite dans le numéro de décembre 2001 (**figure 1**). Au niveau du logiciel de programmation nous envisageons d'utiliser, nous le disions, 3 langages de programmation, à savoir l'assembleur, le BASIC et le C. Nos premiers essais se feront en assembleur. Pourquoi spécifiquement l'assembleur ?
N'est-ce pas passablement difficile, voire même trop difficile pour un début ? Non, pour la simple et bonne raison que les premiers exemples seront compacts et facilement compréhensibles. L'accès via l'assembleur présente l'avantage de forcer à travailler très près du matériel et partant de permettre de voir parfaitement ce qui se passe. Les langages de haut niveau tel que le BASIC camouflent énormément de ce qui se passe réellement.
Lors de notre première expérience nous allons nous contenter de faire basculer certaines des sorties du microcontrôleur. L'activation d'un commutateur par un programme est en fait le premier pas d'une automatisation. Il est possible au demeurant de visualiser le résultat (**figure 2**) en branchant un voltmètre à la ligne de port P1.4 de l'embase K4.
Nous allons utiliser un petit programme en assembleur pour faire changer d'état (commuter) la ligne de port. L'assembleur est, résumé en trois mots, une technique d'écriture d'instructions pour un processeur ou un microcontrôleur. Chaque microcontrôleur possède son set d'instructions qui se compose uniquement de valeurs numériques et des fonctions correspondantes. La série de 6 valeurs numériques données ci-après constitue, pour un microcontrôleur du type 89S8252, un petit programme complet; on parle dans ce cas-là d'un programme en langage machine.

```
116,15,245,144,128,254
```

D'habitude, on préfère remplacer les nombres décimaux par leurs équivalents hexadécimaux plus faciles à lire. Le même programme s'écrit alors ainsi :

```
74 0F F5 90 80 FE
```

On écrit ensuite cette série de nombre dans la mémoire de programme du microcontrôleur. On pourra, pour ce faire, mettre à contribution le programme *MicroFlash*. Les nombres constituant le programme se trouvent enfin dans la mémoire de programme du microcontrôleur Flash. Le microcontrôleur lit la série de nombres de sa mémoire et sait à partir de là ce qu'il lui faut faire, à savoir, en clair :

74 : Ah ah, je dois transférer un nombre vers mon accumulateur (c'est ma mémoire), oui mais lequel ?
0F : Ah bon, 0F, OK, j'ai bien noté 0F.
F5 : Ainsi donc, je dois écrire cette valeur dans un registre. Pourriez-vous peut-être, S.V.P., me dire lequel ?
90 : OK, bien compris, l'adresse 90 est celle du registre pour le port 1. J'exécute.
80 : Il me faut exécuter un petit saut maintenant. Mais où dois-je donc aller ?
FE : 2 octets en arrière par rapport à l'emplacement qui aurait dû être le suivant. D'accord, je saute.
80 : Encore le même saut. Bien à partir de maintenant je me mets donc en boucle.

C'est ainsi que « pense » le microcontrôleur parce que des ingénieurs sioux le lui ont appris. À y regarder de plus près, un microcontrôleur n'est en fait rien de plus qu'une circuiterie complexe de portes logiques. Cette électronique réagit aux états de lignes d'entrée, des lignes de données dans le cas présent, qui relient la mémoire de programme à l'unité de calcul (la CPU).
Si l'on voulait saisir le fin du fin du fonctionnement interne du microcontrôleur on aurait de quoi s'occuper un bon moment. On peut en fait fort bien se contenter de connaître les instructions machine d'un processeur et de savoir les utiliser. Le microcontrôleur proprement dit reste alors une sorte de « boîte noire » dont on ne connaît pas le fonctionnement interne mais dont le comportement est lui facilement traçable. Il faut bien s'y résoudre, le temps de l'humanisme (où l'on pouvait prétendre comprendre et savoir tout ce qui se faisait dans le monde) est bien loin derrière nous. Il en est ainsi avec la technologie moderne, il est pratiquement impossible de rester au courant de toutes les facettes des découvertes.
Un microcontrôleur possède donc sa propre langue qui se résume en fait à des nombres.

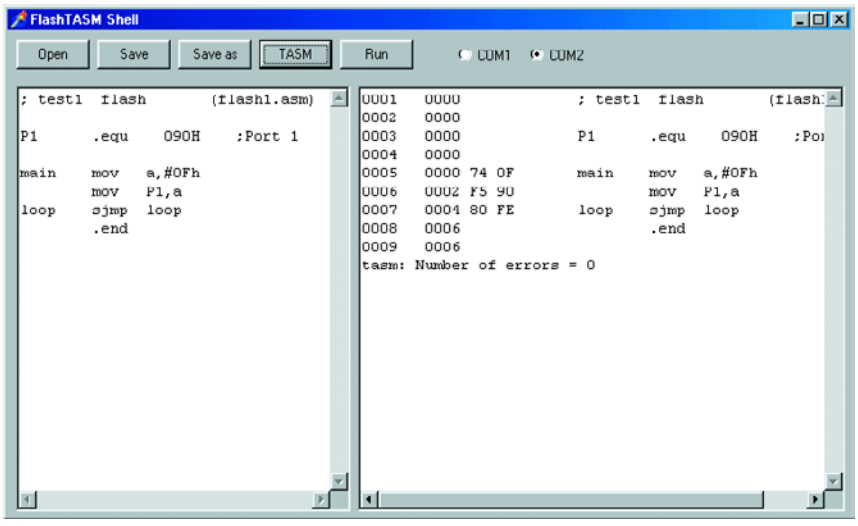


Figure 3. Le premier exemple de programme dans TASMedit.exe.

Il n'en reste pas moins un problème : ce qui ne pose pas de problème de lecture à un microcontrôleur est loin d'être évident pour un être humain. Nous préférons les mots aux chiffres. Ceci explique que l'on ait imaginé des mots plus faciles à mémoriser pour chacune des instructions machine. Le programmeur saisit ces mots à l'aide d'un programme de traitement de texte, un programme spécial les convertira ensuite dans la langue du microcontrôleur. Ce programme s'appelle un assembleur. C'est également là le nom du langage de programmation. L'assembleur est partant une technique d'écriture qui permet de dire au microcontrôleur ce qu'il faut qu'il fasse.

```
main      mov   a,#0Fh
          mov   090H,a
loop      sjmp  loop
```

Les choses deviennent déjà bien plus lisibles. Il suffit en fait de connaître 2 mots spéciaux, à savoir *mov* et *sjmp*. Ce sont ce que l'on appelle des mnémoniques, des mots se substituant aux instructions machine. *mov* (de l'anglais *move*) signifie déplace, transfert ou charge. Cette instruction est suivie par l'endroit où il faut déplacer puis ce qu'il faut déplacer. Dans la première ligne le nombre hexadécimal $0F_{HEX}$, c'est-à-dire 15, (identifié par un « # ») doit être chargé dans l'accumulateur a. L'accumulateur est un registre ou un emplacement de mémoire d'une taille de 8 bits, pouvant de ce fait contenir n'importe quel nombre compris entre 0 et 255.
Dans la seconde ligne, cette valeur d'accumulateur est recopiée vers l'adresse 90_{HEX} (soit 144_D). On trouve à cet endroit un registre dont les lignes sont prolongées vers l'extérieur, à savoir vers les connexions du Port 1. Le mot *sjmp* (*short jump* pour saut court en anglais) requiert de faire un saut de programme à savoir vers l'emplacement appelé *loop*. Le terme de *loop* (*loop* = boucle) a été parfaitement pris au hasard et sert uniquement à désigner une adresse, ici pour la position dans la série d'instructions. L'assembleur traite ces étiquettes (*label*) comme des adresses et utilise pour ce faire les nombres requis. L'instruction *sjmp* peut sauter un maximum de 127 octets vers l'arrière et de 128 octets vers l'avant. Ceci explique que l'on puisse se contenter d'un unique octet pour désigner le but du saut.
Le saut se calcule relativement à la position actuelle dans le programme. On notera que le mot *main* mentionné au début du programme est lui aussi pris au hasard. *mov* et *sjmp* sont les seuls mots-clef d'assembleur. L'esprit humain est parfaitement capable de mémoriser ces 2 termes, il n'est pas nécessaire de disposer de cellules au silicium.

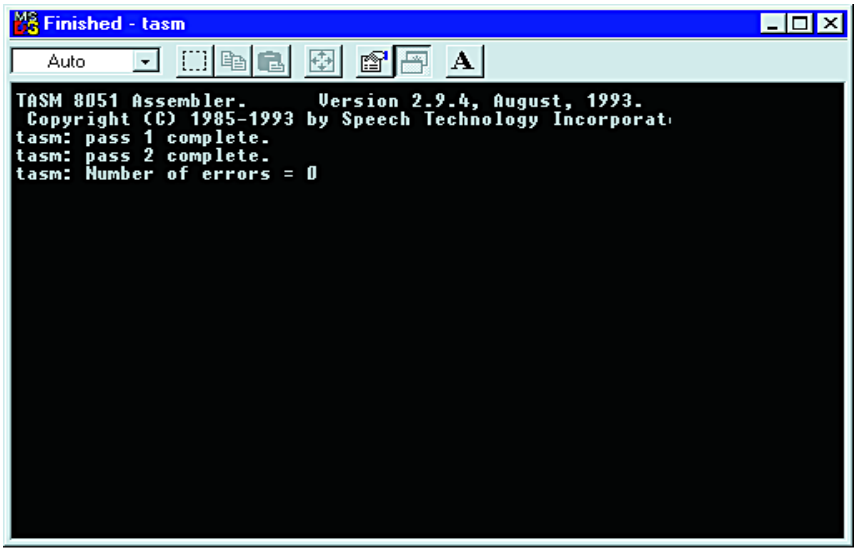


Figure 4. TASM dans une fenêtre DOS.

Il n'en reste pas moins un problème. Est-il vraiment nécessaire de se rappeler qu'un registre donné pour le port 1 se trouve à l'adresse 90_{HEX} ? Il serait plus pratique de pouvoir exprimer cette relation sous forme textuelle. Pas le moindre problème. Il suffit tout simplement d'associer une valeur à un texte. À chaque fois qu'il rencontrera ce texte, l'assembleur utilisera le nombre correspondant auquel il est associé. Il existe pour cela ce que l'on appelle une directive assembleur, à savoir *.equ* (*équate* = est égal à). Ce type de directives assembleur commence toujours par un point. Dans l'exemple qui suit, le mot P1 se voit attribuer la valeur 90_{HEX}. On ne retrouvera plus, partant, dans le programme, la valeur 90_{HEX}, mais le terme P1.

```
; flash1.asm  port output

P1    .equ  090H    ;Port 1

main  mov   a,#0Fh
      mov   P1,a
loop  sjmp  loop
      .end
```

L'examen de ce listage nous apprend autre chose : tous les repères de saut et les mots nouvellement définis se trouvent tout en début de ligne (la gauche de l'écran). Les instructions assembleur se trouvent, elles, légèrement en retrait d'une tabulation vers la droite. On y découvre également l'existence de commentaires qui ne sont d'ailleurs pas inclus lors du processus de conversion.
Ils débutent par un point-virgule (;).
L'écriture du programme peut varier légèrement d'un assembleur à un autre. Nous avons choisi d'utiliser pour ce cours l'assembleur TASM (*Table Driven Assembler*, un programme de Thomas N. Anderson) proposé en shareware. TASM peut se targuer d'être très simple tout en étant capable de convertir des programmes pour un certain nombre de microcontrôleurs différents, sachant qu'il lui faut disposer, pour chacun d'entre eux, d'un tableau d'instructions.
Le programme comporte, tout à la fin, une boucle dans laquelle il saute, à chaque fois et pour l'éternité vers le but du saut, à savoir *loop*. Dans d'autres langages de programmation on parlerait dans ce cas-là d'une boucle fermée fatale qui se comporterait en fait comme un plantage. Le processeur se trouve en effet dans un état dont il est incapable de sortir tout seul. En règle générale il faut toujours que soit parfaitement défini ce qu'un système doit faire lorsqu'il en a terminé avec la tâche en cours. Dans le cas présent, cette boucle revêt une importance capitale. La seule tâche de notre programme est en effet de modifier l'état du port. Si ensuite, on laissait toute liberté de manoeuvre au processeur, il exécuterait au petit bonheur la chance des instructions qui se trouvent encore

dans la mémoire de programme et qui pourraient fort bien provenir d'un programme tout à fait différent. Il faut de ce fait le mettre sur une « voie de garage » par le biais d'une boucle fermée : il peut aller jusqu'à cet endroit et pas plus loin ! Il est vrai que le processeur se trouve enfermé définitivement dans cette boucle. La seule façon de l'en sortir est un Reset (réinitialisation) franc et massif. On pourra ensuite redémarrer le même programme ou charger un nouveau programme et l'exécuter. Le chargement d'un programme se fait également à l'état de réinitialisation, ce n'est pas le processeur qui charge la mémoire de programme mais certains de ses sous-ensembles qui programment la ROM Flash interne du microcontrôleur. Lors de chaque remise sous tension de la carte on a exécution d'un rapide Reset. Le microcontrôleur trouve ensuite de programme ayant été chargé en dernier et le démarre.

Utilisation de l'assembleur TASM

Il est temps de passer aux choses sérieuses ! Nous allons pour cela écrire ce premier programme, le convertir et l'envoyer au microcontrôleur. Il nous faut pour cela un rien de logiciel. Nous allons utiliser l'assembleur shareware bien connu TASM. Le programme se trouve sous la forme d'un fichier .zip sur la disquette de travail accompagnant ce cours mais peut également être téléchargé depuis le site d'Elektor sur Internet (*www.elektor.presse.fr*).
TASM.ZIP devra être décompacté dans un répertoire de travail du disque dur dans lequel seront également placés le programme TASMedit et les exemples de programmes.
La particularité de TASM est qu'il peut être utilisé avec différents types de microcontrôleurs. Il existe, pour chacun d'entre eux, un tableau des instructions machine disponibles qui est activé et transmis lors du lancement du programme. Pour cela on utilise une ligne de commande qui sera, pour l'exemple proposé,

```
TASM -51 -b flash1.asm
  flash1.bin
```

le tableau du set d'instructions *TASM51.Tab* et le format de sortie binaire (.bin). Il est évident que

Le logiciel

Le chargement des programmes dans le microcontrôleur de la carte à 89S8252 Flash requiert l'outil de programmation tournant sous Windows MicroFlash.exe que vous pourrez télécharger depuis le site Internet d'Elektor à l'adresse *www.elektor.presse.fr* dans la liste correspondant au numéro de décembre 2001.
Pour le cours de programmation on pourra se contenter, au début, de l'assembleur TASM, plus tard il faudra disposer du compilateur C READ51 de chez Riegel et du compilateur BASIC BASIC-52. L'assembleur TASM est un shareware très apprécié que vous pourrez télécharger en même temps que TasmEdit du site Elektor. Prenez le temps d'enregistrer ce programme et (s'il vous convient) de payer à son auteur T.N. Anderson son obole méritée.
Le Compilateur-C lui-même est en revanche mis gratuitement à votre disponibilité par la société Rigel pour des utilisations privées et de prise en main. Le compilateur est disponible à l'adresse *www.rigelcorp.com*.
BASIC-52 est un interpréteur Basic produit par Intel qui a été grillé par masque dans la mémoire de programme d'un 8052. Ce contrôleur avait pour dénomination « 80C52-AH-BASIC » et fut, pendant près de 2 décennies, un composant fort apprécié dans le monde des électroniciens et des programmeurs du monde entier.
Il y a quelques années, Intel a cessé la production de ce composant mais a proposé le langage de programmation en code Open Source pour utilisation libre. Le langage a vu son développement se poursuivre; il a même été adapté pour d'autres types de microcontrôleurs. Nous vous avons présenté sa version la plus récente, la version V1.3 dans le numéro de 272 (février 2001, page 45 et suivantes) et existe également sous la forme d'une disquette **EPS000121-11** disponible auprès des adresses habituelles.
La disquette prévue pour ce cours, l'**EPS 010208-11**, comporte l'assembleur TASM, TasmEdit et les premiers exemples tels que BASIC52, MicroFlash.exe et un petit programme de terminal en Basic ainsi que quelques programmes d'exemple.

Format des nombres

Le fait que l'humanité ait un système numérique à base 10 tient sans doute au fait parfaitement aléatoire que nous ayons 10 doigts. Le système de comptage « naturel » d'un ordinateur est le binaire (base 2). Le système hexadécimal est un compromis. Avec ce système, un chiffre peut prendre une valeur comprise entre 0 et 15, sachant que l'on s'est mis d'accord pour représenter les chiffres supérieurs à 9 par les caractères, dans l'ordre, A, B, C, D, E et F.

Décimal	Hexadécimal	Binaire
0	00h	00000000b
1	01h	00000001b
2	02h	00000010b
3	03h	00000011b
...	...	...
10	0Ah	00001010b
11	0Bh	00001011b
12	0Ch	00001100b
13	0Dh	00001101b
14	0Eh	00001110b
15	0Fh	00011111b
16	10h	00010000b
17	11h	00010001b
...	...	...
253	FDh	11111101b
254	FEh	11111110b
255	FFh	11111111b

On a, avec les programmes en assembleur, le choix du mode d'écriture que l'on veut utiliser. Lorsqu'il s'agit de la commande d'un port 8 bits, le mode binaire paraît le mieux adapté. Le chiffre binaire le plus a droite représente, par exemple, la ligne P1.0, celui situé à l'extrême droite, la ligne P1.7. Nos 8 lignes requièrent 8 bits soit 1 octet.
Lorsque TASM convertit un programme on peut indiquer dans quel format il lui faudra sauvegarder le résultat. Si l'on opte pour le format binaire, le fichier ne comportera rien de plus que les octets représentant les différentes instructions machine. IL est impossible à un éditeur de texte de donner à un fichier de ce type quelque signification que ce soit.
Dans le cas du format Intel-Hex on fait appel à des lignes de texte dans lesquelles se trouvent des nombres hexadécimaux. Outre le code proprement dit, chacune de ces lignes comporte également une adresse de début et une somme de vérification. Il est possible de lire et d'examiner ce format sous la forme de texte.

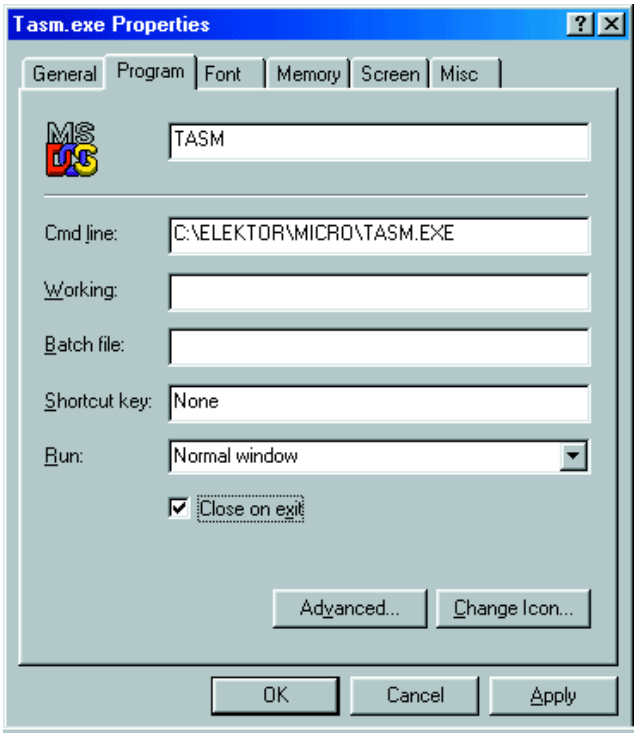


Figure 5. Paramétrage de la caractéristique « Close on exit ». (fermer lors de la clôture).

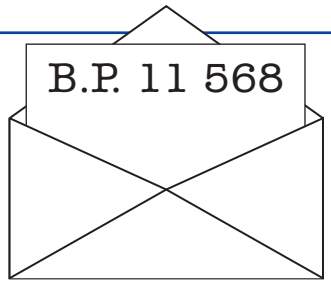
nombre d'entre nous n'avons plus l'habitude de ces lignes de commande, aussi nous ne nous en servirons pas dans ce cours.
En règle générale nous travaillerons sous Windows. L'assembleur TASM n'en reste pas moins un programme DOS pur et dur. Ceci explique la raison de la création de l'environnement Windows, *TASMedit.exe*, écrit à l'intention de ce programme.
Si l'on dispose de son propre éditeur il sera également possible de voir le résultat de la conversion avec d'éventuels messages d'erreur. Ce programme intègre en outre l'outil de téléchargement Flash (*Flash Download tool*) destiné à notre carte à 89S8252 Flash. Nous avons fait de notre mieux pour simplifier la tâche des lecteurs ayant décidé de « suivre » ce cours.
Le programme comporte, comme l'illustre la **figure 3**, 2 fenêtres de texte. Celle de gauche est la fenêtre de l'éditeur du texte-source en assembleur. On pourra y entrer le texte par le biais du clavier ou le charger depuis le disque dur. Le bouton « *TASM* » lance l'assembleur en arrière-plan. La fenêtre de droite donne le fichier du listage assembleur accompagné le cas échéant de messages d'erreur. Elle permet également de suivre le processus de transfert (*download*) vers la carte du microcontrôleur.
Un clic sur le bouton « **TASM** » commence par créer un fichier *Work.asm* à partir du contenu de la fenêtre de l'éditeur. Ce texte de travail subit ensuite une conversion. L'appel de TASM depuis l'environnement Windows se fait à l'aide de la ligne d'instruction

```
TASM -51 -b -
work.asm work.bin
```

On constate que l'on travaille toujours dans ce cas-là en format binaire.
Le résultat de la conversion est sauvegardé dans le fichier *Work.bin*. Et c'est précisément ce fichier que lira le module de téléchargement lorsque l'on cliquera sur le bouton « **Run** ». L'assembleur génère en outre le fichier *Work.lst*, ce que l'on appelle le fichier List qui fournit le résultat de la conversion sous une forme lisible. Le fait d'utiliser à chaque fois la même dénomination de fichier de conversion présente un avantage lors de travaux expérimentaux en assembleur. Il suffira de sauvegarder le texte-source qu'une fois le dernier test réussi. Ainsi, le disque dur ne comporte pas toute une collection de fichiers ratés produits par chaque essai malheureux; on n'y trouvera que le texte-source et les fichiers Work du dernier essais, fichiers sauvegardés en toute connaissance de cause. S'il est arrivé que l'on oublie de sauvegarder la dernière version du texte-source ou que le PC s'est planté au cours de route on pourra toujours réutiliser le fichier *Work.asm* pour sauver les fruits d'un long travail.
Lors du lancement automatique de l'environnement Windows TASM apparaît dans une fenêtre DOS (**figure 4**). Il faudra la fermer avant de se mettre au travail. Au début, il peut encore être très intéressant de voir comment TASM travaille. Plus tard, l'utilisateur « gâté » de Windows en aura sans doute assez au bout du troisième lancement du programme. Il faudra partant qu'à partir de dorénavant la fenêtre se ferme automatiquement.
Pas de problème, Windows connaît en effet une solution. On clique souris droite sur le fichier TASM.EXE et on ouvre le menu *Properties* (paramètres). Sous le point de menu *Properties/Program* on trouve l'option *Close on exit* (fermer lors de la sortie). On activera cette option (**figure 5**). Windows génère alors un lien sous la forme d'un fichier TASM.PIF. Dès lors, la fenêtre DOS se ferme d'elle-même lorsque le travail est terminé.
Une fois que la conversion d'un programme s'est faite avec succès on pourra le transférer, par le bouton « **Run** » dans la Flash système et lancer le programme. Il faudra pour cela paramétrer l'une des interfaces (sérielles) COM du PC et la relier au connecteur de programmation K2 de la carte. Si l'on a remplit l'exigence première du fonctionnement du montage, à savoir appliqué la tension d'alimentation, on pourra procéder au test du programme. Il suffira alors de suivre l'état de sortie du Port 1 pour voir si tout se passe comme prévu. On mesurera, à l'aide d'un appareil de mesure à haute impédance, un niveau très proche de 5 V sur les lignes de port P1.0 à P1,3. Sur les lignes P1.4 à P1.7 en revanche on devrait trouver un niveau pratiquement nul, plus exactement de quelque 30 mV par rapport à la masse.
Au repos et en l'absence de programme, voire après une réinitialisation du processeur, toutes les lignes du port se retrouvent au niveau haut. On pourra mesurer, à l'aide d'un multimètre voire d'un oscilloscope, une tension de +5 V sur chacune de ces lignes. Le programme transféré à nouveau et lancé automatiquement change l'état de 4 des lignes du port; P1.4 à P1.7 présentent maintenant un niveau bas, c'est-à-dire une tension proche de 0 V. Les lignes P1.0 àP1.3 restent elles au niveau haut; la valeur correspondante est un 00001111_B. Il est facile de faire la correspondance entre les niveaux logiques et les différentes lignes du port.

(010208-2)

Après cette introduction succincte au travail avec l'assembleur nous vous proposerons, dans le prochain article de ce cours, quelques petits exemples de programmes qui permettront plus particulièrement d'examiner les caractéristiques du port du microcontrôleur. Il s'agira de sorties, d'entrée et de la vitesse atteinte.



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.

Programmateur Atmel

J'ai une question concernant le projet « Programmateur Atmel » décrit dans le numéro de septembre 2001 d'Elektor. J'ai téléchargé le petit programme tournant sous Windows mais je suis également intéressé par le programme devant être programmé dans le microcontrôleur IC1. J'ai lu que ce circuit intégré était disponible tout programmé (aux adresses habituelles comme vous dites, NdlR : lire pages Publitronic) et qu'il existait également une disquette avec le programme. Pourquoi ne m'est-il pas possible, tout simplement, de le télécharger de votre site ou d'ailleurs ? Cela me paraît beaucoup plus simple.
Corne Daggen

Les programmes téléchargeables gratuitement sont disponibles sur notre site (www.elektor.presse.fr). Il ne nous est malheureusement pas possible de proposer tout gratuitement en raison de droits d'auteur existant sur certains de ces programmes.

Les schémas sur votre site ?

Comment dois-je m'y prendre pour trouver, sur votre site (www.elektor.presse.fr), les schémas correspondant aux platines des montages que vous y proposez ?
Rogier van Cann

Il vous faudra, pour les schémas, liste des composants et description des montages, vous reporter au magazine. Sachant que rester trop longtemps devant son ordinateur et derrière son clavier est la source de problèmes RSI (Repeated Strain Injuries), nous continuons de proposer un magazine sous forme sa forme la plus courante, le « papier ».

Jouvence pour accu

Une petite question concernant le montage « Jouvence pour accu » décrit dans le numéro d'octobre. Je l'ai réalisé et branché sur un vieil accu qu'il était temps de « rajeunir ». La LED rouge du montage s'allume ce qui signifie que l'accumulateur n'est pas en bonne condition. Pour m'en assurer j'ai vérifié la tension aux bornes de la batterie, le montage y étant connecté et ce à l'aide d'un multimètre : tout ce que l'on peut y mesurer sont des impulsions de dépassant pas 0,4 V. Dans votre article vous parlez d'impulsions de plusieurs dizaines de volts. Mon montage fonctionne-t-il correctement ? J'ai utilisé une bobine récupérée sur une alimentation à découpage de la valeur indiquée. Comment puis-je vérifier le bon fonctionnement de ma réalisation ?
H. Voogd

Pour tout vous dire, votre remarque signalant l'allumage de la LED rouge nous donne à penser, avec une quasi-certitude, que votre montage fonctionne correctement. Le problème est que votre voltmètre n'est sans doute pas suffisamment rapide. Il doit en effet mesurer des crêtes de tension extrêmement brèves. Il vaut mieux connecter un oscilloscope à la sortie du montage et vous ne manquerez pas de constater que l'amplitude des impulsions est bien plus importante que ne le « croit » votre voltmètre.

Jouvence pour accu (bis)

Bonjour,
Je voudrais vous faire part d'une remarque technique sur un article paru dans le numéro d'octobre 2001. Il s'agit de la « Jouvence pour Accu ». Il semblerait que le chemin de passage du courant (quand même 1 A !) ne soit pas très explicite lors de la phase de décharge de l'accu correspondant également à la charge de la self L2.
En effet, dans cette phase, le schéma aussi bien que le circuit du typon montre que le seul passage se fait via L1 puis L2 en série. Or la poste vers L1 n'est pas dimensionnée pour 1 A et L:1 non plus (aune mention dans ce sens dans le texte...). Je pense qu'il s'agit d'une erreur que l'on pourra sûrement retrouver dans un prochain « Tort ». Merci de me tenir au courant, directement ou via la rubrique du « Tort ».
Jean-Yves Seyler
(via E-mail)

Rassurez-vous, Mr Seyler, il n'y a pas la moindre erreur de conception en ce qui concerne l'épaisseur des pistes de la platine dessinée à l'intention de ce montage. Il est sans doute vrai que les explications auraient pu être plus explicites. Contrairement à ce que l'on pourrait penser, le courant le plus important ne passe pas par la bobine L1, mais par L2, le transistor T1, sachant que le condensateur C2 « fournit le jus »; L1 est uniquement traversée par le courant moyen qui n'est que de quelques milliampères. Le dessin de la platine correspond ainsi parfaitement à la réalité physique de cette réalisation.

Tort d'Elektor

Alimentation numérique, Elektor n° 282, page 52 et suivantes, n° 283, page 30 et suivantes

Un certain nombre de condensateurs ont refusé de figurer sur la liste des composants. Il s'agit de :
C18 = 100 nF céramique RM5 (comme C4, C7 et C11)
C19 = 10 µF/35 V (au minimum)
C20 = 10 µF/16 V ((au minimum, comme C3, C13, à C17)
Les condensateurs de 10 mF de cette réalisation ont des valeurs de tension de service différentes selon que l'on regarde le schéma ou la liste des composants. Voilà les faits exacts : C3, C13 à C17, C20 doivent avoir une tension de service de 16 V minimum, C19 doit lui avoir une tension de service de 35 V au minimum. Il n'y a pas de danger à opter pour une tension de service plus élevée, une tension plus faible est elle prohibée. (000166)

Carte 89S8252 Flash, Elektor n° 282, page 20 et suivantes

On parle, dans le texte, dans le paragraphe « Test préliminaire », d'un exemple de programme baptisé « **Flashtest1** » dénomination qui est légèrement fausse sachant que le vrai nom de ce programme est « **Flash1** ». On retrouve d'ailleurs ce nom un peu plus loin au niveau du listage 1 qui donne la dénomination correcte de ce programme, à savoir « **Flash1.asm** »; les fichiers correspondants de la disquette et du fichier .zip à télécharger s'appellent eux « **Flash1.hex** » et « **Flash1.bin** ». (010208-1)

Module graphique LCD pour µP 8051, Elektor n° 279, page 8 et suivantes

Il s'est malheureusement glissé quelques erreurs dans la liste des composants. Il faut y supprimer les condensateurs C10 à C12 pour la bonne et simple raison qu'ils n'existent pas (cf. le schéma et la sérigraphie de la platine). (000134-1)

S-VHS/vidéo

Je possède un ordinateur portable doté d'une embase « TV-Out ». Il s'agit en fait d'une sortie S-vidéo, ce qui a pour conséquence que l'image reproduite sur un téléviseur l'est en noir et blanc. Mon revendeur de composants électroniques m'a conseillé de vous demander des informations vu qu'il pensait qu'Elektor aurait peut-être (sans doute ?) une solution.
Tim

Pour tout amateur d'électronique et/ou de micro-informatique, un abonnement à Elektor est toujours rentable, comme vous ne manquerez pas de le constater. La solution à votre problème a été décrite dans le numéro de septembre 2001 d'Elektor en page 79.

SDCC

Je suis très intéressé par le SDCC (Small Device C-Compiler) après avoir lu votre court article à son sujet publié dans le numéro double. Cela fait maintenant plus d'une heure que je me bats avec Internet et j'en ai assez. Open Source - yes! À première cela est réservé aux professionnels des Open Sources. Une fois sur le site en question (NdlR : http://sdcc.sourceforge.net) il est indiqué qu'il faut disposer de CYG-WIN (le full development package), à trouver sur CYG...
Parfait, mais je n'en suis pas devenu sioux (de quoi a-t-on besoin, installation pour Windows etc. ?) Je dois me résigner à continuer à programmer en assembleur (freeware). Dommage.
Wolfgang Thiel
(par E-mail)

Il n'est pas nécessaire, lorsque l'on envisage d'utiliser SDCC, de télécharger CYG. Sur la page d'accueil il est possible de télécharger SDCC.EXE (attention à bien prendre la version pour WIndows !). Ce programme est déjà compilé et fonctionne dans CYG. SDCC suit une politique de Open Source. Ceci signifie que toute personne intéressée par SDCC peut contribuer à son développement. Pour ce faire on a besoin du source de SDCC et de CYG. Ce n'est que dans ces conditions que l'on pourra effectuer la compilation après avoir modifié SDCC. En résumé : on n'a besoin de CYG que si l'on envisage de compiler le compilateur (SDCC) soi-même. C'est simple ? ou peut-être moins simple qu'il n'y paraît à première vue.
Paul Goossens
(Labo Elektor)

Code source du BASIC-52 V1.3

Où et comment puis-je trouver le code source de l'article consacré au BASIC-52 V1.3 du numéro de février 2001 ?
Wlater Kaiserseder
(par E-mail)

Le code-source des modifications caractérisant la version 1.3 du BASIC-52 par rapport au BASIC d'Intel se trouvent sur le site Internet d'Elektor sous l'option de menu « Téléchargements » dans la liste s'affichant après un clic sur le numéro 272.

DOS-Loader pour le AT89S8252

Dans le numéro de décembre 2001 vous proposez un projet à base de microcontrôleur 8051 Flash. Il y a un certain temps déjà j'ai conçu un outil (DOS) simple qui permet de programmer ce composant par le biais de l'interface imprimante d'un PC sans requérir de matériel complexe. Il suffit de 5 lignes :

Port imprimante	ATMEL
Broche 6 (D4)	Broche 9 (RESET)
Broche 7 (D5)	Broche 6 (MOSI)
Broche 8 (D6)	Broche 8 (SCK)
Broche 10 (ACK)	Broche 7 (MISO)
Broche 25 (GND)	Broche 20 (GND)

Le dit outil est freeware et connaît une bonne distribution depuis. J'ai également une version AVR. Le gros avantage de cette version DOS est qu'elle est facile à intégrer dams le compilateur vu qu'elle peut être démarrée depuis un fichier batch .bat. Les programmes tournant sous Windows requièrent souvent d'être pilotés par souris, ce qui pose des problèmes lors du développement d'un programme. Le téléchargement pourra se faire depuis le répertoire Download du site www.mikrocontroller.com (mon domaine) ou, en cas de problèmes de téléchargement, depuis www.freenet.de/buss (les entrées dans le livre d'hôte sont intéressantes). Sur la page d'accueil Microcontrôleurs nous avons un forum assez animé et une platine intéressante qui pourrait intéresser un débutant. Cette page n'a pas de but commercial.
Holger Buss (par E-Mail)

Data-Spy pour NMEA

Le Data-Spy décrit dans le numéro d'octobre 2001 convient à merveille pour tester les données en provenance d'un récepteur GPS (GPS-NMEA). Il n'est pas même nécessaire de disposer du matériel BinTerm ! Si l'on recopie le programme BinTerm sur une disquette on dispose d'un programme de test « portable ». Une autre remarque :
Si l'on copie le programme dans le répertoire-racine (root) d'une disquette il est impossible à BinTerm, en fin de programme, d'actualiser le fichier BinTerm.ini. vu que le trajet adopté est A:\\BinTerm.ini (on notera les 2 \\ !). BinTerm s'attend à trouver un sous-répertoire qu'on ne manquera pas de lui donner. Les choses se passent alors parfaitement même lorsque l'on quitte le programme.
Ferdinand J. Schubert
(via E-Mail)

Update du Data-Spy

Le bogue mentionné par Mr Schubert a été éliminé et depuis j'ai mis la version réactualisée sur ma page d'accueil.
Au cours de cette opération BinTerm s'est vu doté de nouvelles possibilités :
– Une aide (avec Online-Hilfe)
– Prise en compte du tampon horodateur lors d'un changement de ligne (heure:minute:seconde[.milliseconde]
– Texte de changement de ligne également en début de ligne
– Changement de ligne lors d'un changement de direction des données
– Adjonction d'une 4ème page de code « ANSI » standard
– Modification de tous les textes en caractères ANSI dans le protocvole et la conversion des données.

L'adresse de ma page d'accueil (en allemand) :
http://www.mmvisual.de
M. Müller

Avec l'aimable autorisation de l'auteur la version la plus récente de BinTerm (V2.1.1B16) est également mise à votre disposition sur le site Elektor à l'adresse : *www.elektor.presse.fr*.
Le téléchargement du Data-Spy se compose de 2 fichiers :
1. BinTerm.exe – Le programme principal
2. BinTerm.chm – Le fichier d'aide (en allemand).

Les 2 sont intégrés dans un fichier .zip doté du numéro **010041-11a** que vous pourrez trouver dans la liste des téléchargements du numéro 280, octobre 2001.

Li-ion

Dans votre numéro de septembre 2001 vous dites, dans votre article consacré aux accus Lithium-Ion que ces derniers avaient une tension de cellule de, selon le cas, 3,6 ou 3,7 V. Il existe actuellement pour les téléphones portables dotés en série d'accus NiMH ou NiCd de 2,4 V des accus Li-ion de remplacement ayant une tension de 2,4 V. Commnent cela est-il possible ? Comportent-ils, par exemple, une résistance ohmique servant à limiter la tension ? Ou existe-t-il quand même des accu Li-ion ayant une tension de cellule différente.
Il est en outre dit dans l'article que les accus Li-ion sont très sensibles à une surcharge. Reste-t-il partant possible d'utiliser les anciens chargeurs ou faut-il, avec l'achat d'un accu Li-ion, acquérir un nouveau chargeur ?
Volker Gloßner
(via E-mail)

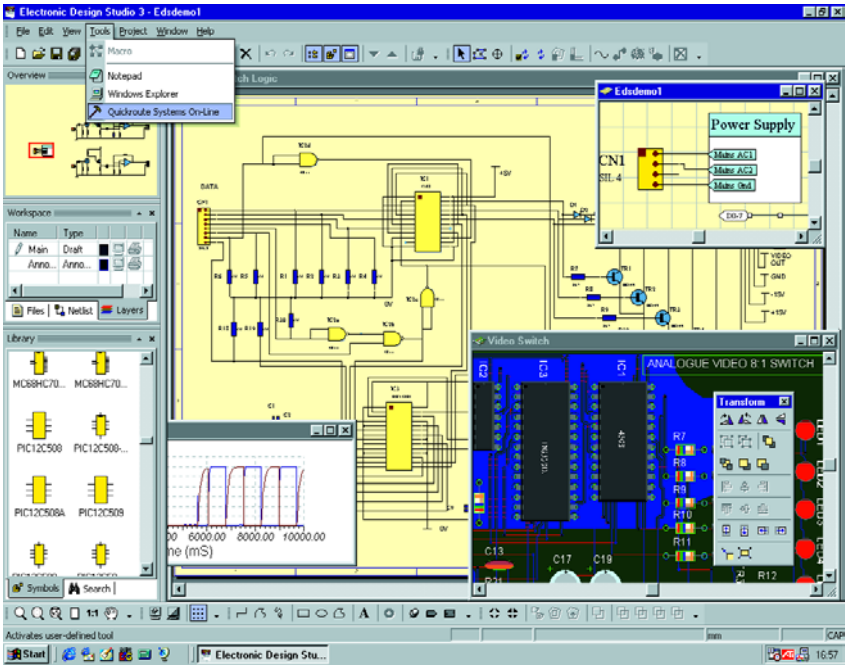
Nous n'avons pas connaissance d'accus Li-ion travaillant à 2,4 V. L'approche la plus probable est que ces packs d'accus sont dotés d'un régulateur de tension (linéaire ou à découpage) pour 2,4 V. Cette solution permet de continuer à utiliser les chargeurs d'origine. Nous avons, dans le numéro 262 d'avril 2000, publié la photo d'un portable à accu Li-ion ouvert où l'on constate la présence d'électronique additionnelle dans le pack d'accus.

Le EDS nouveau, EDS3, est arrivé

Il se pourrait bien que l'attente inévitable qui caractérise la réalisation de prototypes de circuits imprimés soit sensiblement plus courte avec le lancement, par Quickroute (GB) de la nouvelle version de son programme de conception de circuits imprimés, Electronic Design Studio, EDS3 pour les initiés.

Il semblerait que ce nouveau set de logiciels soit le premier en son genre et est en tous cas à la pointe de la technologie actuelle puisqu'il combine 3 programmes puissants en un ensemble parfaitement intégré.

EDS3 n'est pas une simple réactualisation mais une avance importante, offrant aux ingénieurs, concepteurs et enseignants des gains importants de coûts et de temps grâce à l'intégration sans faille des 3 composantes de ce logiciel, offrant la saisie de schéma, la simulation, l'autoroutage, le CADCAM et la fabrication réelle de platines, tout cela en un coffret.
Ian Frost de Quickroute a travaillé près de 12 mois au développement de EDS 3. Voici ce qu'il nous a dit : « Avec EDS 3 l'utilisateur passera moins de temps à passer d'un programme à l'autre et à s'inquiéter des listes d'équipotentielles (*netlist*) et pourra partant consacrer plus de temps à la conception. Ajoutons à cela que EDS 3 travaille de concert avec un périphérique de précision pour produire directement la platine prototype depuis son ordinateur de table.
EDS travaille à base de projet, et avec ses navigateurs de projets et de listes d'équipotentielles vous pouvez sauter vers n'importe quel document, module, symbole, porte ou broche par un simple clic de souris. Plus besoin de partir à la chasse d'une page de schémas ou d'un symbole.
La nouvelle fenêtre de vue d'ensemble permet de zoomer sur n'importe quelle partie du document par un simple clic et le dispositif de CADCheck assure une synchronisation en temps réel de la platine avec son schéma - plus de soucis à se faire avec les listes d'équipotentielles.
EDS 3 offre un support complet pour les schémas multi-feuilles modulaires et hiérarchiques. Des onglets (tags) visuels facilitent les liaisons entre les modules et les feuilles de dessin; il y a également un support complet pour le navigateur de projet.
Il apparaît que l'édition sous EDS 3 est devenue plus simple elle aussi. De nouveaux outils de câblage « Follow me » et des symboles intelligents fournissent une information immédiate quant au câblage réalisé. De nouveaux « habillages » de documents permettent de changer instantanément l'apparence de vos schémas.
La nouvelle mouture de EDS 3 inclut une simulation avancée basée SPICE/XSPICE. Tous les symboles et propriétés de simulation peuvent être paramétrés directement depuis EDS même, vous pouvez créer vos propres modèles, étendre les bibliothèques fournies voire utiliser des modèles SPICE externes. Toutes les sorties sont visualisées sur des fenêtres graphiques dans EDS prêtes à être saisies et transférées dans un rapport par exemple.



Au nombre des nouveautés qui caractérisent EDS 3 on trouve un certain nombre de nouveaux outils d'édition de PCB qui incluent le câblage « Follow me », des pastilles intelligentes qui changent de couleur lorsque le lien établi est correct et le nouveau module « *polyblend* » qui permet de souder des polygones, de percer des orifices dans des polygones et bien plus. Il existe également un « remplisseur » dynamique de zones de cuivre, un autorouteur *rip-up&retry* Viper, et des caractéristiques de DRC à base de boîtier en ligne. Sans parler d'un nombre de couche illimité.
Cependant, l'aspect le plus frappant de EDS 3 est peut-être le fait qu'il établisse un lien direct vers le matériel CADCAM (pour le moment la série CAMM de Roland) pour fabriquer votre platine sur votre propre PC de bureau en utilisant une plaquette cuivrée standard. Les pistes de cuivre sont dessinées en utilisant une technologie à tête flottante de précision. Le programme supporte, pour le moment, le forage et 2 couches.
De l'avis de Ian Frost, la fabrication (DTM pour *DeskTop Manufacture*) sous EDS 3 permet de gagner et du temps et (partant) de l'argent. Il n'est plus nécessaire de faire appel à une agence « extra muros » pour transformer vos dessins en platines. Finis également tous les produits chimiques dangereux pour l'environnement que requiert la gravure chimique des PCB.
En outre, le nouveau produit de Quickroute dispose de barre d'outils totalement paramétrables par l'utilisateur avec une possibilité d'intégrer ses outils et programmes favoris dans le nouveau menu Tools.
EDS 3 peut se targuer d'un « look » moderne avec un rien de XP tant quant à son aspect et sa mise en oeuvre.
La plupart des 18 000 utilisateurs possesseurs d'une licence d'un produit de Quickroute y compris les logiciels EDS devraient pouvoir réactualiser à un coût modeste. Les prix commencent à moins de 100 £ pour des systèmes avec autoroutage et simulation. La version complète de EDS 3 (avec DTM) coûte 3 995 £ + TVA. Pour de plus amples informations concernant EDS 3 on pourra faire un tour sur le site Internet de Quickroute sis à l'adresse *www.dotqr.com*

(017140-1)