

DAVID REY

INTERFACES GSM

2^e édition

**Montages pour
téléphones portables**

ETSF

EDITIONS TECHNIQUES ET SCIENTIFIQUES FRANÇAISES

Consultez nos parutions sur dunod.com

The screenshot shows the Dunod.com homepage. At the top, there's a navigation bar with links for 'Recherche' (Search), 'Collections', and 'Index thématique' (Thematic Index). Below the header, there's a search bar and a sidebar with links for 'Ediscience', 'ETSI', 'InterEditions', and 'Microsoft Press'. The main content area features several book covers: 'Bacchus 2008', 'Profession dirigeant', 'Python', '150 exercices de psychologie à sport', and 'LE DANGER TUE LE LIVRE'. There are also sections for 'Nouveautés' (New titles) and 'Interviews'. A sidebar on the right lists 'LES BIBLIOTHÈQUES DES MÉTIERS' and 'LES NEWSLETTERS'.

Le pictogramme qui figure ci-contre mérite une explication. Son objet est d'alerter le lecteur sur la menace que représente pour l'avenir de l'écrit, particulièrement dans le domaine de l'édition technique et universitaire, le développement massif du photocopillage. Le Code de la propriété intellectuelle du 1^{er} juillet 1992 interdit en effet expressément la photocopie à usage collectif sans autorisation des ayants droit. Or, cette pratique s'est généralisée dans les établissements

d'enseignement supérieur, provoquant une baisse brutale des achats de livres et de revues, au point que la possibilité même pour les auteurs de créer des œuvres nouvelles et de les faire éditer correctement est aujourd'hui menacée. Nous rappelons donc que toute reproduction, partielle ou totale, de la présente publication est interdite sans autorisation de l'auteur, de son éditeur ou du Centre français d'exploitation du droit de copie (CFC, 20, rue des Grands-Augustins, 75006 Paris).



© Dunod, Paris, 2004, 2010
ISBN 978-2-10-055334-1

Couverture : Rachid MARAÏ
Illustrations : Alain et Ursula BOUTEVEILLE

Le Code de la propriété intellectuelle n'autorisant, aux termes de l'article L. 122-5, 2^e et 3^e al., d'une part, que les « copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective » et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, « toute représentation ou reproduction intégrale ou partielle faite sans le consentement de l'auteur ou de ses ayants droit ou ayants cause est illicite » (art. L. 122-4).

Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles L. 335-2 et suivants du Code de la propriété intellectuelle.

TABLE DES MATIÈRES

CHAPITRE	PAGE
Téléchargez les fichiers du livre !	VII
Introduction	1
Généralités	3
1 Codage des SMS	5
1.1 Introduction	6
1.2 Généralités	6
1.3 Mode PDU	6
SMS-SUBMIT	7
SMS-DELIVER	16
1.4 Codage/décodage par logiciel	21
2 Commandes « AT »	25
2.1 Norme GSM07.07	28
Description détaillée des commandes	29
2.2 Norme GSM07.05	41
Description détaillée des commandes	41
3 Matériels utilisés	55
3.1 Téléphones portables	56
Adaptateur TTL/RS232	56
Adaptateur pour FBUS/MBUS (ou M2BUS)	59
Cordons DATA	61
3.2 Modules GSM intégrés	62
Le TM2 de TELTONIKA	62
Pour aller plus loin...	70
4 Interfacer un téléphone GSM	73
4.1 Avec un PC	74
Matériel	74
Hyper Terminal	75
Commandes générales	78
Commandes SMS	83

Commandes spécifiques au TM2 de Teltonika	90
Logiciel intégré pour la gestion des SMS	90
Logiciel « WinGSM »	91
4.2 Avec un PicBasic	92
L'instruction SEROUT	93
Envoi d'un SMS	94
L'instruction SERIN	95
Réception d'un SMS	95
5 Réalisations électroniques	101
5.1 Récepteur/émetteur SMS	102
Récepteur de SMS sur écran LCD	102
Émetteur de SMS	116
5.2 Télécommandes par GSM	123
1 sortie sur relais	123
4 sorties sur relais	131
4 sorties sur triacs	142
4 sorties analogiques	152
5.3 Télémesures par GSM	163
4 entrées logiques	163
4 entrées analogiques	173
Thermomètre	183
5.4 Carte Entrées/Sorties pilotée par GSM	194
PicBasic	194
Schéma électrique	195
Réalisation	204
Programme PicBasic : « ces.bas »	205
Programmation et configuration	221
Essais	222
Interface de puissance	224
5.5 Géolocalisation par GSM	228
Cell Monitor	228
Tracker GPS	233
Positionnement géographique	233
Annexes	251
Glossaire	261
Bibliographie	264

TÉLÉCHARGEZ LES FICHIERS DU LIVRE !

Tous les circuits imprimés, programmes et logiciels des montages décrits dans cet ouvrage sont téléchargeables à partir du site :

<http://www.dunod.com>

Il faut tout d'abord rechercher l'ouvrage (par titre ou par auteur) puis, une fois sur la page dédiée à l'ouvrage, cliquer sur *documents téléchargeables* dans la rubrique *Compléments en ligne*. Un mot de passe, issu de l'ouvrage, vous sera alors demandé.

L'impression directe sur transparent des circuits imprimés pourra ainsi être réalisée facilement. La meilleure qualité sera obtenue en utilisant une imprimante laser qui permet, de par sa précision, un rendu du tracé exceptionnel. Si l'on utilise une imprimante à jet d'encre, il conviendra de vérifier minutieusement le résultat car des micro-coupures apparaissent fréquemment sur les pistes.

La dernière solution est de sortir une impression sur papier et d'effectuer une photocopie de ce dessin sur transparent. Dans ce cas, il est nécessaire de réaliser deux transparents et de les superposer, le tracé n'étant pas assez opaque pour l'insolation aux ultraviolets.



INTRODUCTION

Actuellement le réseau GSM français compte pas moins de 58 millions d'usagés. Le « portable » est devenu en quelques années un produit de consommation courante. Ce petit trésor de technologie ouvre la porte à de nombreuses applications électroniques sans fil à celui qui sait l'interfacer avec un PC ou un micro-contrôleur. Il devient alors possible via l'envoi et la réception de SMS de piloter et de surveiller un processus quelconque. La distance n'est désormais plus un souci puisque le réseau GSM couvre 99 % du territoire français et ne cesse de progresser au niveau mondial.

GÉNÉRALITÉS

L'usage d'un téléphone portable est bien entendu destiné à l'être humain, aussi il dispose d'interfaces qualifiées d'homme-machine. Parmi ces interfaces citons le clavier qui permet par exemple la saisie du numéro de téléphone de la personne que l'on souhaite contacter. Une seconde interface l'écran permet de contrôler visuellement que le numéro saisi est correct. La dernière interface constituée par le micro et le haut-parleur permet de converser oralement avec son interlocuteur. Il existe un autre type d'interface machine-machine du fait peu connu de l'utilisateur. Physiquement cette interface prend la forme d'un connecteur multibroche. De nombreux accessoires prennent place sur ce connecteur, citons par exemple les kits mains libres ou piétons, le chargeur de batterie... Parmi ces accessoires un nous intéresse plus particulièrement, il s'agit d'un cordon d'adaptation RS232 qui permet d'accéder à toutes les fonctions du téléphone. Un simple PC muni lui aussi d'un port RS232 et équipé d'un logiciel terminal standard suffit alors à prendre le contrôle du processeur central du téléphone, encore faut-il « parler » le même langage que celui-ci...

1 CODAGE DES SMS

1.1	Introduction	6
1.2	Généralités	6
1.3	Mode PDU	6
1.4	Codage/décodage par logiciel	21

2	Commandes « AT »	25
3	Matériels utilisés	55
4	Interfacer un téléphone GSM	73
5	Réalisations électroniques	101
	Annexes	251
	Glossaire	261
	Bibliographie	264

1.1 INTRODUCTION

Un des services offert par le GSM est la gestion des mini-messages ou SMS (*Short Message Service*). Il ne s'agit plus de la transmission de sons mais d'un texte limité théoriquement à 160 caractères. Chaque message envoyé transite vers un centre de messagerie baptisé SMSC (*Short Message Service Centre*) où il est temporairement stocké. Dès que le destinataire est disponible, c'est-à-dire lorsque le mobile est sous tension et présent dans une zone couverte par le réseau GSM, le message est transmis. Le message peut exceptionnellement rester sur le SMSC durant plusieurs jours, ce qui est le cas notamment en fin d'année où le nombre de SMS envoyés atteint des records, vœux de bonne année oblige... En situation normale on peut considérer que l'envoi d'un SMS est instantané à condition que le mobile de destination soit opérationnel. Les SMSC sont identifiés par un numéro d'appel spécifique à chaque opérateur. Il est possible d'envoyer un SMS autrement qu'avec un téléphone portable. On trouve sur Internet des logiciels qui permettent à un PC muni d'un modem d'accéder à un SMSC. Le moyen le plus simple et le moins onéreux consiste à passer par le Web. Certains providers comme AOL proposent à leurs abonnés un service SMS en ligne. Il existe même des sites qui permettent l'envoi gratuit de SMS au prix tout de même d'un message publicitaire qui vient se greffer à votre message.

1.2 GÉNÉRALITÉS

Il y a deux façons de transmettre un message SMS, soit par le mode PDU qui est le mode de base ou le mode TEXT. Le mode PDU est une suite de caractères hexadécimaux qui codifient le SMS, le mode TEXT n'est rien d'autre qu'une représentation sous forme de texte des données qui composent le SMS. Il y a différents types d'alphabets utilisés pour passer du mode PDU au mode TEXT. Par exemple votre téléphone portable affiche en mode TEXT sur son écran les données d'un éventuel SMS reçu, c'est lui qui détermine automatiquement quel type d'alphabet à utiliser. Attention, il faut savoir que certains téléphones portables ne supportent pas le mode TEXT lorsqu'ils sont interfacés avec un PC. Il est donc utile de connaître en détail le mode PDU.

1.3 MODE PDU

Lorsqu'un mobile A envoie un SMS au mobile B, le message transite obligatoirement par un centre de messagerie baptisé SMSC. Dans son message l'utilisateur du mobile A doit définir deux adresses, celle du SMSC qu'il souhaite utiliser et celle du mobile B.

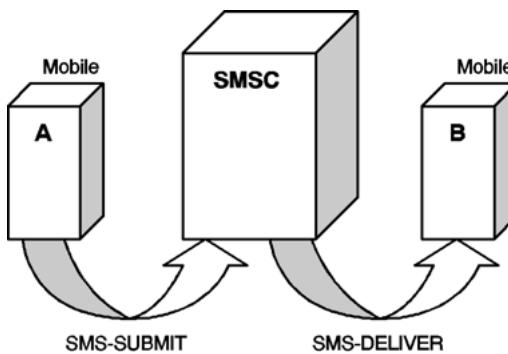


Figure 1.1.

L'acheminement du dit message du mobile A à destination du SMSC est baptisé **SMS-SUBMIT**. Une fois le message traité par le SMSC il est délivré au mobile B, on parle alors de **SMS-DELIVER** (**figure 1.1**).

Les protocoles SMS-SUBMIT et SMS-DELIVER sont des PDU, il en existe d'autres, d'importance moindre, qui permettent de signaler des éventuelles erreurs d'acheminement :

- SMS-DELIVER-REPORT : le cas échéant, il indique une défaillance lors du transfert du SMS par le SMSC au destinataire ;
- SMS-SUBMIT-REPORT : le cas échéant, il indique une défaillance lors du transfert du SMS par le mobile au SMSC ;
- SMS-STATUS-REPORT : le SMSC envoie un rapport d'état au mobile émetteur du SMS ;
- SMS-COMMAND : le mobile envoie une commande au SMSC.

Les trames codant le SMS sont différentes suivant le type de PDU mis en œuvre. Dans la partie qui va suivre nous nous contenterons de décrire les deux principaux protocoles : SMS-SUBMIT et SMS-DELIVER.

SMS-SUBMIT

Téléphone GSM vers SMSC.

La taille maximale de la trame d'un SMS-SUBMIT est de 173 octets. Le champ le plus important en terme de taille est le champ qui codifie le corps du message qui peut atteindre 140 octets. Les deux autres champs indispensables sont SCA qui codifie l'adresse du SMSC et DA qui codifie l'adresse de l'émetteur.

1-10 octets 1 octet 1 octet 2-12 octets 1 octet 0-7 octets 1 octet 1 octet 0-140 octets

SCA	PDU	MR	DA	PID	DCS	VP	UDL	UD
-----	-----	----	----	-----	-----	----	-----	----

SCA : Service Centre Adresse

Adresse du centre de messagerie.

1 octet	2 octets	0 - 8 octets
LEN	Type Number	Numéro SMSC

Le champ SCA ne possède pas une taille fixe, elle dépend de la longueur du numéro du SMSC utilisé, ce paramètre est stocké dans le champ LEN.

LEN : nombre d'octets nécessaires pour codifier le numéro du SMSC.

Type Number : indique le format du numéro de téléphone du SMSC.

7	6	5	4	3	2	1	0
1	Type number			Numbering Plan Identification			

Type number : spécifie le type de numéro de téléphone utilisé. La valeur la plus utilisée est 001_{bin} qui signale un numéro de type international (**tableau 1.1**).

Tableau 1.1.

Bit 6	Bit 5	Bit 4	Description
0	0	0	Format non spécifié
0	0	1	Numéro international
0	1	0	Numéro national
0	1	1	Numéro spécifique au réseau
1	0	0	Numéro d'abonné
1	0	1	Codification en accord avec la norme GSM TS 03.38 alphabet par défaut sur 7 bit
1	1	0	Numéro abrégé
1	1	1	X

Numbering Plan Identification : le numbering plan identification est pris en compte dans le cas où le type number est égal à 000_{bin} , 001_{bin} ou 010_{bin} . Si type number est égal à 101_{bin} alors les bits 3 à 0 sont réservés. Pour adresser n'importe quelle entité, le numbering plan identification doit être égal à 0001_{bin} (**tableau 1.2**).

Bit 3	Bit 2	Bit 1	Bit 0	Description
0	0	0	0	X
0	0	0	1	ISDN/téléphone numbering plan (E.164/E.163)
0	0	1	1	Data numbering plan (X.121)
0	1	0	0	Telex numbering plan
1	0	0	0	National numbering plan
1	0	0	1	Private numbering plan
1	0	1	0	ERMES numbering plan (ETSI DE/PS 3 01-3)
1	1	1	1	X

Tableau 1.2.

Compte tenu de ce que nous avons dit plus haut, le **Type Number** le plus utilisé est : 91_{hex} .

7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	1

Numéro du SMSC (service de centre de messagerie) : attention, le codage est effectué en décimal codé binaire (BCD). Un octet contient donc deux quartés codés en BCD et qui plus est de poids inversés. Le nombre de semi-octet devant être obligatoirement paire il est parfois nécessaire de compléter par F_{hex} .

Digit2	Digit1	Digit4	Digit3	...	Digit n	Digit n-1
--------	--------	--------	--------	-----	---------	-----------

Voici les principaux numéros utilisés en France pour contacter un SMSC, le signe « + » signale qu'il s'agit d'un numéro international :

- +33609001390 (SFR)
- +33689004000 (Orange)
- +33660003000 (Bouygues Télécom)

Par exemple codons le numéro du SMSC utilisé par les abonnés d'Orange de France Télécom : le nombre de chiffres étant impair il est nécessaire d'ajouter un F_{hex} à la fin. Ensuite il suffit de permuter chaque chiffre comme le montre le tableau ci-après.

INTERFACES GSM

Octet n° 1	Octet n° 2	Octet n° 3	Octet n° 4	Octet n° 5	Octet n° 6
3 3	6 0	9 0	0 1	3 9	0 F
3 3	0 6	0 9	1 0	9 3	F 0

Finalement le numéro +33660003000 une fois codé devient : 3306091093F0.

Remarque : le champ SCA est optionnel, de ce fait lorsqu'il est positionné à 00_{hex} cela signifie que le SMSC utilisé est celui stocké dans la mémoire du téléphone, correspondant en principe à l'opérateur auquel vous avez souscrit votre abonnement.

Type de PDU

Le champ PDU toujours codé sur 1 seul octet a pour fonction principale de définir s'il s'agit d'un SMS-DELIVER ou d'un SMS-SUBMIT (bits 0 et 1) – **tableau 1.3.**

7	6	5	4	3	2	1	0
RP	UDHI	SRR	VPF	RD	MTI		

Tableau 1.3.

Champ	Bit 7	Description	
RP	0	Il n'existe pas de chemin de repli	
	1	Il existe un chemin de repli	
Champ	Bit 6	Description	
UDHI	0	Le champ UD contient uniquement un message	
	1	Le champ UD contient un en-tête en plus du message	
Champ	Bit 5	Description	
SRR	0	Un rapport d'état ne sera pas retourné	
	1	Un rapport d'état sera retourné	
Champ	Bit 4	Bit 3	Description
VPF	0	0	Le champ VP n'est pas présent
	0	1	X
	1	0	Le champ VP existe, il est codifié en entier (relatif)
	1	1	Le champ VP existe, il est codifié en semi-octet (absolu)

Champ	Bit 2	Description	
RD	0	Indique au SMSC qu'il ne doit pas recevoir un nouvel SMS portant le même MR et la même adresse de destination	
	1	Indique au SMSC qu'il peut accepter la réception d'un nouvel SMS portant le même MR et la même adresse de destination	
Champ	Bit1	Bit 0	Description
MTI	0	1	SMS-SUBMIT : Achemine le SMS du mobile vers le SMSC

Tableau 1.3 (suite).

MR : Message référence

Chaque message envoyé par le mobile au SMSC est identifié par un numéro compris entre 0 et FF_{hex} baptisé MR (Référence du Message). Le fait de positionner MR à 0 indique au mobile que c'est lui qui doit définir automatiquement ce champ. Dans ce cas il s'incrémentera pour chaque nouveau message envoyé pour un même destinataire. Ce champ est lié au bit **RD** du champ **PDU**.

1 octet

MR

DA : Destination Adress

Le codage de l'adresse de l'émetteur est sur le principe semblable au codage de l'adresse du SMSC (voir champ SCA).

1 octet

2 octets

0 - 8 octets

LEN	Type Number	Numéro du destinataire
-----	-------------	------------------------

LEN : longueur du numéro du destinataire correspondant cette fois au nombre de chiffres et pas au nombre de semi-octets utilisés pour sa codification comme c'est le cas pour le SMSC.

PID : Protocol Identifier

1 octet

PID

Le champ **PID** codé sur un octet indique à quel type de service télématique est destiné le message. Dans notre cadre d'utilisation ce champ sera toujours positionné à 00_{hex}. Attention, s'il est

INTERFACES GSM

certain que le PID 00_{hex} est supporté par tous les SMSC, il en est autrement pour les autres (**tableau 1.4**).

Tableau 1.4.

PID	Description
00 _{hex}	La trame est traitée comme un message court
11 _{hex}	La trame est traitée comme un telex
02 _{hex}	La trame est traitée comme un telefax de groupe 3
03 _{hex}	La trame est traitée comme un telefax de groupe 4
12 _{hex}	La trame est traitée comme un e-mail

Pour plus d'informations vous pouvez consulter la norme GSM 03.40 chapitre 9.2.3.9.

DCS : Data Coding Scheme

Le champ DCS indique de quelle manière est codé le champ UD qui correspond au corps du message. Il peut aussi indiquer une classe du message (bits 4 à 7 positionnés à 1). Dans la pratique, tous les bits sont positionnés à zéro (**tableau 1.5**).

Tableau 1.5.

7	6	5	4	3	2	1	0
Coding Group							

Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Description
0	0	0	0	0	0	0	0	Indique que le champ UD est codé avec l'alphabet GSM, aucune classe n'est spécifiée
0	0	0	0	0	0	0	1	Réservé
0	0	0	0	
0	0	0	1	1	1	1	1	
1	1	1	1	0	0	x	x	Indique que le champ UD est codé avec l'alphabet par défaut, chaque caractère est codé sur 7 bits, une classe est spécifiée
1	1	1	1	0	1	x	x	Indique que le champ UD est codé en ASCII sur 8 bits
1	1	1	1	0	x	0	0	Classe 0 : le message s'affiche immédiatement à l'écran
1	1	1	1	0	x	0	1	Classe 1 : spécifique au mobile (ME)
1	1	1	1	0	x	1	0	Classe 2 : spécifique à la carte SIM
1	1	1	1	0	x	1	1	Classe 3 : spécifique à l'équipement terminal (TE)

Avec l'alphabet GSM chaque caractère est codé sur 7 bits (voir Annexes). Avec ce type d'alphabet il est donc possible de coder 8 caractères avec 7 octets. Le champ UD peut dans ce cas codifier un total de 160 caractères.

Dans la pratique, le plus simple consiste à positionner tous les bits du champ DCS à zéro, ce qui sélectionne l'alphabet GSM, aucune classe n'est mentionnée ainsi c'est le mobile de destination qui choisira le stockage adéquat du SMS.

VP : Validity Period

Permet d'indiquer au SMSC la durée de validité du SMS à condition que les bits 3 et 4 (champ VPF) de l'octet PDU soient correctement positionnés. Si VPF (voir champ PDU) est à 0, le champ VP sera ignoré par le SMSC, le SMS aura une durée de vie illimitée.

Cette durée peut être relative ($VPF = 10_{bin}$), si le SMSC n'a pas réussi à transmettre le SMS au destinataire dans la durée définie par VP, le SMS est détruit (**tableau 1.6**).



VP _{dec}	Durée de validité du SMS
0 ... 143	$(VP + 1) \times 5 \text{ minutes}$
144 ... 167	$12 \text{ heures} + ((VP - 143) \times 30 \text{ minutes})$
168 ... 196	$(VP - 166) \times 1 \text{ jour}$
197 ... 255	$(VP - 192) \times 1 \text{ semaine}$

Tableau 1.6.

La durée peut être absolue ($VPF = 11_{bin}$), le SMSC à jusqu'à la date définie par VP pour délivrer le SMS au destinataire, passé cette date le message est détruit.

| 1 octet |
|---------|---------|---------|---------|---------|---------|---------|
| Année | Mois | Jour | Heure | Minute | Seconde | Fuseau |

En codage absolu le champ VP se compose de 7 octets, contenant chacun deux champs codés en BCD et de poids inversés. Le champ Fuseau exprimé en quart d'heure indique la différence entre l'heure locale et l'heure GMT.

■ *UDL : User Data Length, UD : User Data*

1 octet	0 - 140 octets
UDL	UD

■ UDL contient la taille en octets utilisés pour codifier le message dans UD.

Exemple de codage

À titre d'exemple, essayons de constituer la trame qui permettrait d'envoyer un SMS ayant une validité de 4 jours, contenant le message « TEST » au numéro « 0612345678 » en utilisant le centre de messagerie « +33609001390 ».

Numéro du SMSC utilisé : **+33609001390**.

Le signe « + » indique qu'il s'agit d'un numéro international, on a donc **Type Number = 91_{hex}**.

7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	1
Numéro international				ISDN / Telephone numbering plan			

■ Pour chacun des octets qui composent le numéro, on inverse les deux groupes composés de 4 bits chacun. Le nombre de chiffres étant impair, il est nécessaire d'ajouter l'octet F_{hex} à la fin du numéro. On obtient le numéro suivant : **3306091093F0**.

3	3	6	0	9	0	0	1	3	9	0	F
3	3	0	6	0	9	1	0	9	3	F	0

■ Le nombre d'octets utilisés pour coder le numéro du SMSC est de 07_{dec} d'où **LEN = 07_{hex}** (Type Number inclus).

Finalement **SCA = 07913306091093F0**.

7	6	5	4	3	2	1	0
RP	UDHI	SRR	VPF		RD	MTI	
0	0	0	1	0	0	0	1

■ RP = 0. Il n'existe pas de chemin de repli.

UDHI = 0. Le champ UD contient uniquement un message.

SRR = 0. Aucun rapport d'état ne sera retourné au mobile.

VPF = 10. Le champ VP et codé en relatif.

MTI = 01. SMS-SUBMIT (Envoi).

On a donc PDU = 11_{hex}.

Référence du SMS, MR = 00_{hex}, indique que c'est le mobile qui doit définir cette valeur.

Le numéro de téléphone du destinataire est « 0612345678 », la codification est identique à celle du champ SCA. Dans le cas présent le nombre de chiffres étant pair il n'est pas utile d'ajouter F_{hex} à la fin.

0	6	1	2	3	4	5	6	7	8
6	0	2	1	4	3	6	5	8	7

On obtient le numéro : 6021436587.

Nous allons utiliser un format de numéro non spécifié donc **Type of Number = 81_{hex}**.

7	6	5	4	3	2	1	0
1	0	0	0	0	0	0	1

Le numéro comporte 10 chiffres donc **LEN = 0A_{hex}**.

D'où **DA = 0A816021436587**.

Le message doit être traité par le SMSC comme un SMS donc **PID = 00_{hex}**.

Le champ UD est codé avec l'alphabet par défaut, aucune classe n'est spécifiée donc **DCS = 00_{hex}**.

La durée de validité du message est limitée à 4 jours. On applique la formule nb jours = VP - 166, avec nb jours = 4 cela implique que **VP = AA_{hex}**.

Corps du message : « TEST », comme spécifié par le champ DCS le codage est réalisé avec l'alphabet GSM. Pour chacun des caractères composant le message on cherche la correspondance en binaire dans le tableau alphabet GSM (voir Annexes). Un caractère correspond à un bloc de 7 bits. Ensuite on regroupe les bits par paquet de 8 en commençant par la droite, remarquez qu'il est nécessaire d'ajouter 4 zéros pour compléter le dernier paquet.

Chaque octet est finalement converti en un nombre hexadécimal, pour ce faire vous pouvez utiliser la calculatrice scientifique de Windows (**tableau 1.7**).

Tableau 1.7.

T	S	E	T
0000 1010100	1010011	1000101	1010100
→ 00001010	→ 10010100	→ 11100010	→ 11010100
0A	94	E2	D4

Chaque lettre est codée sur 7 bits, pour former la trame composée d'octets on regroupe les bits par bloc de 8 on en déduit alors la valeur en hexadécimal.

On a donc dans le champ **UD = D4E294OA**.

Il faut 4 octets pour coder le message donc **UDL = 04**.

Finalement pour envoyer un SMS ayant une validité de 4 jours, contenant le message « TEST » au numéro « 0612345678 » en utilisant le centre de messagerie « +33609001390 » il faut constituer la trame :

07913306091093F011000A8160214365870000AA04D4E294OA

Pour utiliser le centre de messagerie associé au téléphone, il suffit de remplacer les octets concernant le SMSC par 00 :

0011000A8160214365870000AA04D4E294OA

SMS-DELIVER

SMSC vers téléphone GSM.

La taille maximale de la trame d'un SMS-DELIVER est de 173 octets. Le champ le plus important en terme de taille est le champ qui codifie le corps du message qui peut atteindre 140 octets. Les deux autres champs indispensables sont SCA qui codifie l'adresse du SMSC et OA qui codifie l'adresse du destinataire.

1-10 octets	1 octet	2-12 octets	1 octet	1 octet	7 octets	1 octet	0-140 octets
SCA	PDU	OA	PID	DCS	SCTS	UDL	UD

SCA : Service Centre Adresse

Adresse du centre de messagerie.

Le codage est identique à celui présenté dans la partie SMS-SUBMIT. Dans le cas présent il indique quel est le SMSC qui a traité le SMS.

PDU : Protocol Data Unit

	7	6	5	4	3	2	1	0
	RP	UDHI	SRI	X	X	MMS	MTI	

Champ	Bit 7	Description	
RP	0	Il n'existe pas de chemin de repli	
	1	Il existe un chemin de repli	
Champ	Bit 6	Description	
UDHI	0	Le champ UD contient uniquement un message	
	1	Le champ UD contient un en-tête en plus du message	
Champ	Bit 5	Description	
SRI	0	Aucun rapport d'état ne sera retourné au mobile	
	1	Un rapport d'état sera retourné au mobile	
Champ	Bit 2	Description	
MMS	0	Des messages supplémentaires pour le MS sont en attente dans le SMSC	
	1	Pas de message supplémentaire en attente pour le MS dans le SMSC	
Champ	Bit 1	Bit 0	Description
MTI	0	0	SMS-DELIVER : Achemine le SMS du mobile vers le SMSC

Tableau 1.8.

Le champ MTI est le plus important, dans le cas d'un SMS-DELIVER il est positionné à 00.

OA : Originator Adress

Le codage de l'adresse de l'émetteur est sur le principe semblable au codage de l'expéditeur dans la partie SMS-SUBMIT.

1 octet 2 octets 0 - 8 octets

LEN	Type Number	Numéro de l'émetteur du SMS
-----	-------------	-----------------------------

PID : Protocol Identifier

Le codage est identique à celui présenté dans la partie SMS-SUBMIT.

1 octet

PID

SCTS : Service Centre Time Stamp

1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet
Année	Mois	Jour	Heure	Minute	Seconde	Fuseau

Le champ SCTS se compose de 7 octets, contenant chacun deux champs codés en BCD et de poids inversés. Il indique au destinataire la date et l'heure à laquelle le SMS est arrivé au SMSC. Le champ Fuseau exprimé en quart d'heure indique la différence entre l'heure locale et l'heure GMT.

UDL : User Data Length, UD : User Data

1 octet

0 - 140 octets

UDL	UD
------------	-----------

Exemple de décodage

Imaginons que nous devions décoder la trame suivante :

07913306091093F0000A81609121436500009920215075032104D4E
2940A

Le premier octet « 07 » nous indique que l'adresse du SMSC utilisé pour acheminer le SMS est codée sur 7 octets.

On extrait donc les 7 octets suivants afin de déterminer le numéro du SMSC, soit **913306091093F0**.

Le premier octet « 91 » nous indique qu'il s'agit d'un numéro international.

7	6	5	4	3	2	1	0
1	0	0	1	0	0	0	1
Numéro international							

Les 6 octets suivants contiennent le numéro du SMSC :

Pour chacun des octets qui composent le numéro on inverse les deux groupes composés de 4 bits chacun.

3	3	0	6	0	9	1	0	9	3	F	0
3	3	6	0	9	0	0	1	3	9	0	F

L'avant dernier octet F ne correspond à aucun chiffre, il n'est là que pour avoir un nombre de chiffres pair.

Le numéro du SMSC utilisé pour convoyer le message est donc : **+33609001390**.

L'octet suivant « 00 » indique le **PDU** utilisé :

7	6	5	4	3	2	1	0
RP	UDHI	SRI	X	X	MMS	MTI	
0	0	0	0	0	0	0	0

RP = 0. Il n'existe pas de chemin de repli.

UDHI = 0. Le champ UD contient uniquement un message.

SRI = 0. Aucun rapport d'état ne sera retourné au mobile.

MMS = 0. Pas de message supplémentaire en attente pour le MS dans le SMSC.

MTI = 0. SMS-DELIVER.

L'octet suivant $0A_{hex} = 10_{dec}$ indique le nombre de chiffres composant le numéro de l'émetteur du message. Attention on ne comptabilise pas l'octet utilisé pour le champ Type Number. D'où OA = **0A816091214365**.

6	0	9	1	2	1	4	3	6	5
0	6	1	9	1	2	3	4	5	6

Après permutation des chiffres on obtient le numéro : 0619123456.

PID = **00_{hex}** donc il s'agit d'un message SMS.

DCS = **00_{hex}** donc le champ UD est codé avec l'alphabet par défaut, aucune classe n'est spécifiée.

Les 7 octets suivants **99202150750321** codifient le champ SCTS (**tableau 1.9**).

INTERFACES GSM

Tableau 1.9.

1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet	1 octet
Année	Mois	Jour	Heure	Minute	Seconde	Fuseau	
9 9	2 0	2 1	5 0	7 5	0 3	2 1	
9 9	0 2	1 2	0 5	5 7	3 0	1 2	
12 février 1999			05 : 57 : 03			GMT+3h	

Le SMS a donc été expédié par le SMSC le 12 février 1999 à 05 : 57 : 03 (GMT+3h).

Il reste à décoder le dernier morceau de la trame : **04D4E294OA**.

L'octet $04_{\text{hex}} = 04_{\text{dec}}$ indique la longueur du champ **UD** contenant le corps du message.

Tableau 1.10.

0A	94	E2	D4
00001010	10010100	11100010	11010100
0000 1010100	1010011	1000101	1010100
T	S	E	T

Chaque octet exprimé en hexadécimal est converti en un nombre binaire composé de 8 bits. D'après le champ DCS on sait que UD est codé avec l'alphabet GSM. On regroupe les bits par paquets de 7 en commençant par la gauche. En s'aidant du tableau de conversion de l'alphabet GSM (voir Annexes) on en déduit le caractère correspondant à chaque paquet. Finalement en effectuant une lecture de la droite vers la gauche on obtient le corps du message soit « TEST ».

En conclusion la trame :

**07913306091093F0000A81609121436500009920215075032104D4E
2940A**

signifie qu'il s'agit d'un SMS contenant le message « TEST » envoyé par « 0619123456 » traité par le SMSC « +33609001390 » le 12 février 1999 à 05 : 57 : 03 (GMT+3h).

1.4 CODAGE/DÉCODAGE PAR LOGICIEL

Comme vous pouvez le constater le codage/décodage manuel d'une trame PDU est assez fastidieux. Dans la pratique ceci est heureusement totalement transparent pour l'utilisateur du téléphone portable. D'une part le numéro du SMSC utilisé est celui figurant dans la mémoire du mobile, définit par l'opérateur, il est donc inutile de le mentionner lors de la rédaction du message. Le numéro du correspondant peut être sélectionné dans le répertoire ou alors saisie manuellement en mode TEXT. Le corps du message est également saisi en mode TEXT à l'aide du clavier. Rien de plus simple en somme pour rédiger un SMS. Tous les autres champs que nous avons vus précédemment sont gérés par le processeur du mobile. Le mobile se charge ensuite de convertir chacun des champs en valeurs hexadécimales pour constituer la trame qui sera finalement envoyée sur le réseau. Le mobile destinataire du SMS fera le cheminement inverse pour restituer à l'utilisateur seulement les informations pertinentes sur son écran. Malheureusement pour certains téléphones lorsque le port série est relié par exemple à un PC, le mode TEXT n'est plus supporté. Les trames SMS affichées/constituées à l'écran du PC sont obligatoirement en mode PDU ce qui complique fortement les manipulations. Heureusement nous allons faire en sorte grâce au logiciel « ConvertSMS.exe » (**figures 1.2 et 1.3**) que ce soit le PC qui prenne en charge le codage/décodage des données SMS. Le logiciel développé avec Delphi 4 est relativement simple, le code fait largement appel aux fonctions de manipulation de chaînes de caractères.



Figure 1.2.



Figure 1.3.

Algorithme de codage

Tous les paramètres correspondant au codage d'un SMS sont présents dans l'onglet nommé : « TEXT -> PDU ».

Pour faciliter le codage d'un SMS nous allons figer certains champs (grisés). Le champ SCA est positionné à 00, ainsi le numéro du SMSC utilisé est celui présent dans la mémoire du portable, inutile de s'en soucier. Le champ PDU est à 11_{hex} pour indiquer qu'il s'agit d'un SMS-SUBMIT et que le champ VP est codé en relatif (bit 3 = 0 et bit 4 = 1). Le champ MR est figé à zéro, ainsi l'identification du message est confiée au mobile. Comme il s'agit d'un SMS, le champ PID est à zéro. Le champ DCS est aussi à zéro, ainsi le champ UD doit être codé avec l'alphabet GSM et aucune classe n'est spécifiée. Les autres champs devront être renseignés par l'utilisateur, sauf le champ UDL qui est calculé par le logiciel. Le champ DA correspond au numéro du destinataire du SMS. Le champ VP indique la durée de validité du SMS, sa valeur est comprise entre 0 et 255_{dec}, la durée correspondante s'affiche à côté, par exemple avec VP = 170_{dec} le message aura une durée de validité de 4 jours. Le champ UD correspond au texte du message. Le bouton « convertir » permet d'afficher la trame au format PDU correspondante aux informations saisies. Dans la copie d'écran nous avons repris l'exemple vu précédemment.

Algorithme de décodage

Le deuxième onglet nommé « PDU -> TEXT » permet le décodage d'un SMS réceptionné.

On considère que la trame commence toujours par les caractères « 0791 », en effet le numéro du SMSC est toujours international donc Type Number = 91_{hex} et il se compose de 11 chiffres, donc codé sur 7 octets. Le bouton « convertir » permet de décoder les différents champs correspondant à la trame saisie ; attention lors de la saisie de la trame à ne pas insérer de saut de ligne qui viendrait gêner le décodage. Dans la copie d'écran nous avons repris l'exemple vu précédemment.

ConvertSMS.dll

Ceux qui pratiquent un langage autre que Delphi ne sont pas oubliés, les fonctions de codage/décodage ont été compilées dans le fichier « ConvertSMS.dll ». Les deux fonctions à déclarer et à appeler dans votre programme sont :

```
Function PduToText(pdu: string):TTEXT;
```

L'argument pdu contient la trame à convertir.

La fonction retourne la variable typée TTEXTE constituée ainsi :

```
type
  TTexte=record
    SMSC:string[12];
    PDU:string[2];
    OA:string[12];
    PID:string[2];
    DCS:string[2];
    UDL:string[160];
    SCTS:string[50];
  end;
  Function TextToPdu(SCA_PDU_MR:string;targetms:string;PID_DCS:
  string;VP:Integer;text:string):TPDU;
```

L'argument SCA_PDU_MR contient les 3 champs concaténés SCA, PDU et MR, par exemple : « 001100 ».

targetms contient le numéro du destinataire, par exemple : « 0601020304 ».

PID_DCS contient les champs concaténés PID et DCS.

VP est un nombre entier compris entre 0 et 255.

text contient le texte du message.

■ La fonction retourne la variable typée TPDU constituée ainsi :

```
type  
TPDU=record  
Len:string[12];  
Trame:string[255];  
end;
```

■ Le formalisme de déclaration et d'appel des fonctions contenues dans la DLL « ConvertSMS.dll » s'effectuera bien évidemment suivant le type de langage utilisé.

2 COMMANDES « AT »

2.1 Norme GSM07.07	28
2.2 Norme GSM07.05	41

3	Matériels utilisés	55
4	Interfacer un téléphone GSM	73
5	Réalisations électroniques	101
	Annexes	251
	Glossaire	261
	Bibliographie	264

Il existe un standard de télécommunication européen (ETS) qui spécifie une liste de commandes AT qui permettent l'accès aux fonctions d'un téléphone portable par l'intermédiaire d'un terminal. Ces commandes s'inspirent fortement du standard Hayes, du nom de la société américaine qui dans les années 1970 a défini une liste de commandes universelles permettant de piloter un modem. Chaque instruction débute par les caractères ASCII « AT » tirés de l'abréviation « ATtention » et se termine par un retour chariot, CR : *Carriage Return*, d'où le nom souvent donné à cette série de commandes : instructions « AT ». On peut effectivement comparer un téléphone portable à un modem sans fil, il est donc logique qu'il utilise des instructions semblables au modem fixe qui équipe nos PC. Les constructeurs se doivent de fabriquer des téléphones portables qui respectent ces normes. La première baptisée **GSM07.07** permet l'accès aux fonctions générales du téléphone, la deuxième **GSM07.05** concerne la gestion des SMS.

Dans les textes officiels qui traitent du GSM on retrouve les termes **ME** pour *Mobile Equipment* qui correspond par exemple à un téléphone portable, **TE** pour *Terminal Equipment* qui physiquement peut être un ordinateur ou un microcontrôleur et **TA** pour *Terminal Adaptator* qui assure la liaison entre le ME et le TE, à ne pas confondre avec le câble série.

Dans la pratique il y a trois possibilités concernant la disposition des différents éléments (**figure 2.1**) :

- TA, ME et TE sont trois entités distinctes ;
- *TA et ME forment une seule entité*, ce qui est le cas le plus fréquent. Par exemple un téléphone portable standard ou un terminal GSM contient dans son boîtier à la fois le TA et le ME. Le TE forme une entité à part, par exemple il peut s'agir d'un ordinateur de type PC qui dispose d'un port série ou d'un circuit électronique basé sur un µC qui implémente un port série ;
- TA, ME et TE forment une seule entité.

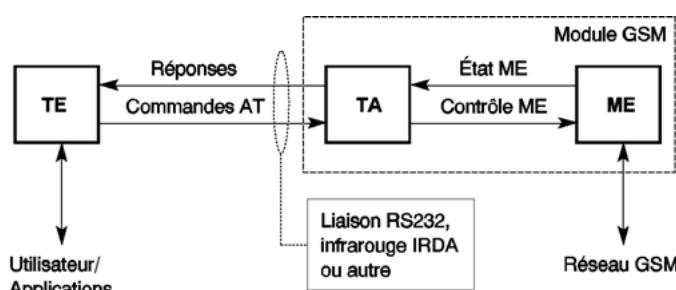


Figure 2.1.

Paramètres

Liste des différents paramètres qui sont utilisés avec les commandes AT :

- <xxx> Indique que xxx est un paramètre de la commande AT associée.
- [<xxx>] Indique que le paramètre <xxx> est facultatif.
- <CR> *Carriage Return* (retour chariot)
 $\langle CR \rangle = 13_{dec} = 0D_{hex}$
- <LF> Line Feed
 $\langle LF \rangle = 10_{dec} = 0A_{hex}$
- <ctrl-Z/ESC> Touche CTRL plus touche Z équivalent au code ASCII EOF pour *End Of File*; en informatique c'est un caractère qui signale la fin d'un fichier, ici il signale la fin d'une instruction.
 $\langle CTRL-Z \rangle = 26_{dec} = 1A_{hex}$
 OU
 Touche ESC ou Escape permet de sortir de la commande en cours de frappe sans qu'elle ne soit exécutée.
 $\langle ESC \rangle = 27_{dec} = 1B_{hex}$

Comme le montre le **tableau 2.1**, il existe trois manières d'envoyer une même commande AT.

Tableau 2.1.

Commande de test	AT+CXXX=?	Retourne la liste des paramètres utilisables avec la commande CXXX.
Commande de lecture	AT+CXXX?	Retourne le ou les paramètres en cours associés à la commande CXXX.
Commande d'écriture	AT+CXXX=<xxx>	Applique le ou les paramètres <xxx> à la commande CXXX.

Dans tous les cas le téléphone doit répondre, favorablement ou non, à la commande envoyée. Si la commande est acceptée, la réponse renvoyée est de la forme : <CR><LF>OK<CR><LF>. Si la commande n'est pas reconnue, ou que le ME rencontre un problème lors de son exécution, un message d'erreur est renvoyé : <CR><LF>ERROR<CR><LF>, accompagné éventuellement d'un message décrivant la nature de l'erreur (voir commande AT+CMEE).

2.1 NORME GSM07.07

La norme GSM07.07 regroupe environ 80 commandes permettant d'accéder à toutes les fonctions du ME. Nous n'allons pas détailler la totalité de ces commandes mais seulement celles qui seront susceptibles de nous intéresser dans les chapitres suivants (**tableau 2.2**).

Tableau 2.2.

Commandes	Fonction	Page
AT+CGMI	Identification fabricant	29
AT+CGMM	Identification modèle	29
AT+CGMR	Identification version	29
AT+CGSN	Identification numéro de série (IMEI)	30
AT+CIMI	Information d'identité internationale du mobile (IMSI)	30
AT+CLIP	Présentation du numéro	31
AT+CSCS	Alphabet utilisé par le TE	32
AT+CPAS	État d'activité du téléphone	32
AT+CPIN	Entre le code PIN	33
AT+CBC	État de charge batterie	33
AT+CREG	Enregistrement sur le réseau	34
AT+CSQ	Qualité du signal	35
AT+CIND	Indicateurs de contrôle	35
AT+CPBS	Sélectionne un répertoire téléphonique	36
AT+CPBR	Lecture du répertoire téléphonique	37
AT+CPBF	Recherche une entité dans le répertoire téléphonique	37
AT+CPBW	Écriture dans le répertoire téléphonique	38
AT+CCLK	Horloge	38
AT+CALA	Alarme	39
AT+CMEE	Signalisation d'une erreur	39

Description détaillée des commandes

AT+CGMI : Identification fabricant	
Commande de test AT+CGMI=?	Réponse OK si erreur +CME ERROR: <err>
Commande de lecture AT+CGMI <i>(exceptionnellement pas de point d'interrogation)</i>	Réponse +CGMI: <manufactur> Renvoie des informations (2 048 caractères au maximum) concernant le fabricant du ME. si erreur +CME ERROR: <err>

AT+CGMM : Identification modèle	
Commande de test AT+CGMM=?	Réponse OK si erreur +CME ERROR: <err>
Commande de lecture AT+CGMM <i>(exceptionnellement pas de point d'interrogation)</i>	Réponse +CGMM: <model> Retourne le modèle du ME. si erreur +CGMM ERROR: <err>

AT+CGMR : Identification version	
Commande de test AT+CGMR=?	Réponse OK si erreur +CME ERROR: <err>
Commande de lecture AT+CGMR <i>(exceptionnellement pas de point d'interrogation)</i>	Réponse +CGMR: <revision> Retourne la version du ME. si erreur +CGMR ERROR: <err>

INTERFACES GSM

AT+CGSN : Identification numéro de série (IMEI)	
Commande de test AT+CGSN=?	Réponse OK si erreur +CME ERROR: <err>
Commande de lecture AT+CGSN <i>(exceptionnellement pas de point d'interrogation)</i>	Réponse +CGSN: <sn> Retourne le numéro de série du ME nommé IMEI (<i>International Mobile station Equipment Identity</i> ; voir norme GSM 03.03). si erreur +CGMI ERROR: <err>

AT+CIMI : Information d'identité internationale du mobile (IMSI)	
Commande de test AT+CGMI=?	Réponse OK si erreur +CME ERROR: <err>
Commande de lecture AT+CIMI <i>(exceptionnellement pas de point d'interrogation)</i>	Réponse <IMSI> Retourne le numéro IMSI : <i>International Mobile Subscriber Identity</i> . Permet au TE d'identifier la carte SIM liée au ME. si erreur +CGMI ERROR: <err>

AT+CLIP : Présentation du numéro	
Commande de test AT+CLIP=?	Réponse +CLIP: (liste des <n>s supportés) si erreur +CGMI ERROR: <err>
Commande de lecture AT+CLIP?	Réponse +CLIP= <n>,<m> Paramètres <n> : paramètre qui active/désactive la présentation du numéro au TE 0 inactif (valeur par défaut) 1 actif <m> : paramètre qui indique l'état de la fonctionnalité « présentation du numéro » 0 L'opérateur ne fournit pas ce service (non prévu dans l'abonnement) 1 L'opérateur fournit ce service 2 Inconnu (ex : connexion au réseau impossible...) si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CLIP=[<n>]	Réponse OK Paramètre <i>Voir commande de lecture</i> si erreur +CGMI ERROR: <err> Remarque : Si la présentation du numéro est active (AT+CLIP=1) et à condition que l'appelant ne soit pas en mode secret, le numéro est envoyé au TE : +CLIP: <number>,<type>[,<alpha>] <number> : numéro de téléphone <type> : type de numéro (national/international) <alpha> : nom provenant du répertoire, correspondant au numéro de téléphone

INTERFACES GSM

AT+CSCS : Alphabet utilisé par le TE	
Commande de test AT+CSCS?	Réponse OK si erreur +CME ERROR: <err>
Commande de lecture AT+CSCS=?	Réponse +CSCS: <chset> Indique au TA quel est l'alphabet utilisé par le TE. Ainsi le TA peut correctement convertir les chaînes de caractères entre le TE et le ME. si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CSCS= <chset>	Réponse OK Paramètre <chset> : "GSM" GSM alphabet par défaut "HEX" chaîne de nombre hexadécimal 00 à FF "IRA" alphabet de référence international (ITU-T T.50 [13]) si erreur +CGMI ERROR: <err>

AT+CPAS : État d'activité du téléphone	
Commande de test AT+CPAS=?	Réponse +CPAS: (liste des <pas>s supportés)
Commande de lecture AT+CPAS?	Réponse +CPAS : <pas> Paramètre Retourne l'état d'activité <pas> du ME 0 prêt 1 indisponible 2 indéfini 3 sonnerie (le ME est prêt pour le transfert de commandes entre TA/TE, mais la sonnerie est active) 4 appel en cours (le ME est prêt pour le transfert de commandes entre TA/TE, mais un appel est en cours) 5 en veille si erreur +CGMI ERROR: <err>

AT+CPIN : Entre le code PIN	
Commande de test AT+CPIN=?	Réponse OK si erreur +CGMI ERROR: <err>
Commande de lecture AT+CPIN?	Réponse +CPIN= <code> Paramètre <code> : READY ME aucun mot de passe à donner SIM PIN ME attente du SIM PIN SIM PUK ME attente SIM PUK si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CPIN=<pin>	Réponse OK Permet de rentrer le code PIN si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CPIN= <pin>,<newpin>	Réponse OK Permet de modifier le code PIN, <newpin> est le nouveau code. +CGMI ERROR: <err>

AT+CBC : Charge de la batterie	
Commande de test AT+CBC=?	Réponse +CBC: (liste des <bcs>s), (liste des <bcl>s) si erreur +CGMI ERROR: <err>
Commande de lecture AT+CBC?	Réponse +CBC: <bcs>,<bcl> Paramètres <bcs> : état de connexion de la batterie 0 le ME est alimenté par la batterie 1 le ME est connecté à une batterie, mais il n'est pas alimenté par celle-ci 2 le ME n'est pas connecté à une batterie 3 défaut d'alimentation <bcl> : niveau de charge de la batterie 0 la batterie est déchargée, ou le ME ne dispose pas de batterie 1..100 capacité de la batterie entre 1 et 100 % si erreur +CGMI ERROR: <err>

INTERFACES GSM

AT+CREG : Enregistrement sur le réseau	
Commande de test AT+CREG=?	Réponse +CREG: (liste des <n>s supportés) si erreur +CGMI ERROR: <err>
Commande de lecture AT+CREG?	Réponse +CREG: <n>,<stat>[,<lac>,<ci>] Paramètres <n> : indique dans quel mode la commande AT+CREG est utilisée 0 commande inactive (valeur par défaut) retourne un code indiquant si le téléphone est enregistré sur le réseau +CREG: <stat> retourne un code <stat> indiquant si le téléphone est enregistré sur le réseau et les informations <lac> et <ci> <stat> : 0 téléphone non enregistré, pas de recherche d'opérateur en cours 1 téléphone enregistré sur le réseau 2 téléphone non enregistré, recherche d'opérateur en cours 3 enregistrement interdit 4 inconnu 5 enregistré, <i>roaming</i> <lac> : <i>Location Area Code</i> , deux octets codés en hexadécimal <ci> : Cell-ID, deux octets codés en hexadécimal si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CREG=[<n>]	Réponse OK Paramètre <i>Voir commande de lecture</i> si erreur +CGMI ERROR: <err>

AT+CSQ : Qualité du signal	
Commande de test AT+CSQ=?	Réponse +CSQ: (<list des <rssi>s supportés),(<list des <ber>s supportés) si erreur +CGMI ERROR: <err>
Commande de lecture AT+CSQ?	Réponse +CSQ: <rssi>,<ber> Paramètres <rssi>: 0 – 113 dBm ou moins 1 – 111 dBm 2..30 - 109... – 53 dBm 31 – 51 dBm ou plus 99 inconnu ou non détectable <ber>: 0..7 RXQUAL 99 inconnu ou non détectable si erreur +CGMI ERROR: <err>

AT+CIND : Indicateurs de contrôle	
Commande de test AT+CIND=?	Réponse +CIND: (<descr>,(<list des <ind>s supportés)) [,(<descr>,<list des <ind>s supportés))[,...]] si erreur +CGMI ERROR: <err>
Commande de lecture AT+CIND?	Réponse +CIND: <ind>[,<ind>[,...]] Paramètres <ind> : niveau correspondant au <descr> <descr> : "battchg" niveau de charge de la batterie (0-5) "signal" qualité du signal (0-5) "service" disponibilité du service (0-1) "sounder" activité sonore (0-1) "message" message reçu (0-1) "call" appel en cours (0-1) "vox" transmission activée par activité vocale (0-1) "roam" indicateur de roaming (0-1) "smsfull" la mémoire de stockage des sms est pleine (1), ou disponible (0) si erreur +CGMI ERROR: <err>

INTERFACES GSM

AT+CPBS : Sélectionne un répertoire téléphonique	
Commande de test AT+CPBS=?	Réponse +CPBS: (liste des <storage>s supportés) OK si erreur +CGMI ERROR: <err>
Commande de lecture AT+CPBS?	Réponse +CPBS: <storage>[,<used>,<total>] Paramètres <i>Voir commande d'écriture</i> si erreur +CPBS ERROR: <err>
Commande d'écriture AT+CPBS=<storage>	Réponse OK Paramètres <storage>: "DC" : liste des numéros appelés "EN" : numéro d'urgence stocké dans mémoire SIM ou ME "FD" : répertoire fixe "LD" : dernier numéro appelé "MC" : liste des numéros appelés, mais sans réponse "ME" : répertoire du ME "MT" : répertoire combiné, ME et SIM "ON" : numéros propres à la carte SIM / ME (MSISDNs) "RC" : liste des appels reçus "SM" : répertoire de la carte SIM "TA" : répertoire du TA <used> : indique l'espace utilisé dans la mémoire <total> : taille de la mémoire si erreur +CGMI ERROR: <err>

AT+CPBR : Lecture du répertoire téléphonique	
Commande de test AT+CPBR=?	<p>Réponse</p> <p>+CPBR: (<index>s supportés),[<nlength>],[<tlength>]</p> <p>Paramètres</p> <p><nlength> : taille maximum du champ <number></p> <p><tlength> : taille maximum du champ <text></p> <p>si erreur</p> <p>+CGMI ERROR: <err></p>
Commande d'écriture AT+CPBR=<index1>[,<index2>]	<p>Réponse</p> <p>+CPBR: <index1>,<number>,<type>,<text>[[...]]<CR><LF>+CPBR: <index2>,<number>,<type>,<text>]]</p> <p>Cette commande affiche le contenu du répertoire situé entre les emplacements <index1> et <index2></p> <p>Paramètres</p> <p><number> : numéro de téléphone</p> <p><type> : type de numéro</p> <p><text> : nom de la personne</p> <p>si erreur</p> <p>+CGMI ERROR: <err></p>

AT+CPBF : Recherche une entité dans le répertoire téléphonique	
Commande de test AT+CPBF=?	<p>Réponse</p> <p>+CPBF: [<nlength>],[<tlength>]</p> <p>OK</p> <p>Paramètres</p> <p><nlength> : taille maximum du champ <number></p> <p><tlength> : taille maximum du champ <text></p> <p>si erreur</p> <p>+CGMI ERROR: <err></p>
Commande de lecture AT+CPBF=<findtext>	<p>Réponse</p> <p>+CPBF: <index1>,<number>,<type>,<text>[[...]]<CR><LF>+CPBF: <index2>,<number>,<type>,<text>]]</p> <p>Recherche un élément dans le répertoire courant (celui sélectionné par la commande +CPBS) qui commence par les caractères spécifiés par <findtext></p> <p>Paramètres</p> <p><number> : numéro de téléphone</p> <p><type> : type de numéro</p> <p><text> : nom de la personne</p> <p>si erreur</p> <p>+CGMI ERROR: <err></p>

INTERFACES GSM

AT+CPBW : Écriture dans le répertoire téléphonique

Commande de test AT+CPBW=?	Réponse +CPBW: (<index>s supportés),[<nlength>],[<tlength>] Paramètres <nlength> : taille maximum du champ <number> <tlength> : taille maximum du champ <text> si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CPBW= [<index>] [,<number> [,<type>[,<text>]]]	Réponse OK Paramètres <number> : numéro de téléphone <type> : type de numéro <text> : nom de la personne si erreur +CGMI ERROR: <err>

AT+CCLK : Date / Heure

Commande de test AT+CCLK?	Réponse +CCLK: <time> Retourne la date et l'heure au format "aa/mm/jj","hh:mm:ss" si erreur +CGMI ERROR: <err>
Commande d'écriture AT+CCLK=<time>	Réponse OK mise à jour de la date et de l'heure si erreur +CGMI ERROR: <err>

AT+CALA : Alarme	
Commande de test AT+CALA=?	<p>Réponse</p> <p>+CALA: (liste des <n>s supportés), (liste des <type>s supportés), <tlength></p> <p>Paramètres</p> <p>Voir commande de lecture si erreur</p> <p>+CGMI ERROR: <err></p>
Commande de lecture AT+CALA?	<p>Réponse</p> <p>+CALA: <time>,<n1>,<type>,[<text>] [<CR><LF>+CALA: <time>,<n2>,<type>,[<text>][...]]</p> <p>Paramètre l'alarme horaire du ME. Il est possible de programmer plusieurs alarmes, chaque alarme affiche un message différent sur l'écran du ME.</p> <p>Paramètres</p> <p><time> : date et heure au format "aa/mm/jj" "hh:mm:ss"</p> <p><n>, <n1>, <n2> : nombre entier spécifiant l'index de l'alarme (spécifique au fabricant)</p> <p><type> : nombre entier qui indique le type d'alarme, son, volume, Led...</p> <p><text> : texte qui doit s'afficher sur l'écran du ME lorsque l'alarme est active</p> <p><Tlength> : taille maximum du champ <text></p> <p>si erreur</p> <p>+CGMI ERROR: <err></p>
Commande d'écriture AT+CALA=<time> [, <n>[, <type> [, <text>]]]	<p>Réponse</p> <p>OK</p> <p>Programmation d'une alarme</p> <p>si erreur</p> <p>+CGMI ERROR: <err></p>

AT+CMEE : Signalisation d'une erreur	
Commande de test AT+CMEE=?	<p>Réponse</p> <p>+CMEE: (liste des <n>s supportés)</p>
Commande de lecture AT+CMEE?	<p>Réponse</p> <p>+CMEE: <n></p>
Commande d'écriture AT+CMEE=[<n>]	<p>Paramètres</p> <p><n>:</p> <p>0 seul le code "ERROR" est retourné</p> <p>1 retourne le code "ERROR" plus une valeur numérique</p> <p>2 retourne le code "ERROR" plus un commentaire</p>

Codes d'erreur

Lorsqu'une commande échoue, un code d'erreur <err> peut être renvoyé au TE (voir commande AT+CMEE) :

- 0..... échec du téléphone
- 1..... pas de connexion au téléphone
- 2..... "phone-adaptor link reserved"
- 3..... opération interdite
- 4..... opération non supportée
- 5..... PH-SIM PIN requis
- 6..... PH-FSIM PIN requis
- 7..... PH-FSIM PUK requis
- 10..... SIM absente
- 11..... SIM PIN requis
- 12..... SIM PUK requis
- 13..... échec de SIM
- 14..... SIM occupée
- 15..... SIM fausse
- 16..... mot de passe incorrect
- 17..... SIM PIN2 requis
- 18..... SIM PUK2 requis
- 20..... mémoire pleine
- 21..... index invalide
- 22..... non trouvé
- 23..... échec de mémoire
- 24..... chaîne de texte trop longue
- 25..... caractère invalide dans la chaîne
- 26..... numéro de téléphone trop long
- 27..... caractère invalide dans le numéro
- 30..... pas de réseau
- 31..... timeout réseau
- 32..... pas de réseau, appel d'urgence seulement
- 40..... code PIN d'identification sur le réseau requis
- 41..... code PUK d'identification sur le réseau requis
- 42..... code PIN second d'identification sur le réseau requis
- 43..... code PUK second d'identification sur le réseau requis
- 44..... code PIN d'identification sur l'opérateur requis
- 45..... code PUK d'identification sur l'opérateur requis
- 46..... code PIN requis pour une identification
- 47..... code PUK requis pour une identification
- 100.... inconnu

Attention, certains éléments présentés ci-avant sont extraits d'un document officiel de l'ETSI et par conséquent ils sont soumis à un copyright :

« © ETSI 1999. Further use, modification, redistribution is strictly prohibited. ETSI standards are available from:

<http://pda.etsi.org/pda/> and <http://www.etsi.org/eds/> »

2.2 NORME GSM07.05

La norme GSM07.05 spécifie les commandes AT permettant la gestion des SMS (**tableau 2.3**).

Commande	Fonction	Page
AT+CSMS	Sélection du service de messagerie	45
AT+CPMS	Sélection de la zone mémoire pour le stockage des SMS	46
AT+CMGF	Sélection du format du SMS (PDU ou TEXT)	46
AT+CSCA	Définition de l'adresse du centre de messagerie	47
AT+CSDH	Affiche en mode TEXT le paramétrage des SMS	47
AT+CSAS	Sauvegarde du paramétrage	48
AT+CRES	Restauration du paramétrage par défaut	48
AT+CNMI	Indication concernant un nouveau SMS	49
AT+CMGL	Liste les SMS stockés en mémoire	50
AT+CMGR	Lecture d'un SMS	50
AT+CMGS	Envoie un SMS	51
AT+CMSS	Envoie d'un SMS stocké en mémoire	51
AT+CMGW	Écriture d'un SMS	52
AT+CMGD	Efface un SMS	52

Tableau 2.3.

Description détaillée des commandes

Paramètres concernant le stockage

<index> Nombre entier indiquant l'emplacement du SMS dans la mémoire associée.

<mem1> Mémoire dans laquelle les messages sont lus ou effacés. Les commandes utilisant ce paramètre

sont +CMGL qui liste les messages, +CMGR qui effectue la lecture d'un message et +CMGD qui efface un message.

<mem2> Mémoire utilisée pour rédiger ou envoyer un message. Les commandes utilisant ce paramètre sont +CMSS qui envoie le message situé dans cette mémoire et +CMGW qui écrit un message dans cette mémoire.

<mem3> Mémoire utilisée pour stocker les messages reçus. Les messages reçus peuvent éventuellement être directement transmis au TE, voir commande +CNMI.

Les paramètres <mem1>, <mem2> et <mem3> peuvent théoriquement prendre les valeurs suivantes :

<memx>	Description
ME	Stockage dans la mémoire du mobile
MT	Tous les stockages associés au mobile
SM	Stockage dans la carte SIM
TA	Stockage dans le TA (<i>Terminal Adaptator</i>)

<stat> Si le mobile est utilisé en mode PDU ce paramètre est un nombre entier compris entre 0 et 4. En mode TEXT il s'agit alors d'une chaîne de caractères. Dans les deux cas il indique l'état du message situé en mémoire :

Mode PDU	Mode TEXT	Signification
0	REC UNREAD	Message reçu non lu
1	REC READ	Message reçu lu
2	STO UNSENT	Message stocké non envoyé
3	STO SENT	Message stocké déjà envoyé
4	ALL	Tous les messages (voir commande +CMGL)

<total1> nombre entier indiquant le nombre de messages qu'il est possible de stocker dans la mémoire <mem1>.

- <total2> nombre entier indiquant le nombre de messages qu'il est possible de stocker dans la mémoire <mem2>.
- <total3> nombre entier indiquant le nombre de messages qu'il est possible de stocker dans la mémoire <mem3>.
- <used1> nombre entier indiquant le nombre de messages stockés dans la mémoire <mem1>.
- <used2> nombre entier indiquant le nombre de messages stockés dans la mémoire <mem2>.
- <used3> nombre entier indiquant le nombre de messages stockés dans la mémoire <mem3>.

Paramètres concernant les données

On retrouve bien entendu des paramètres communs à ceux que nous avons détaillés dans le chapitre « Codage des SMS ».

- <alpha> Il s'agit d'une chaîne de caractères représentant le nom du destinataire <da> ou de l'expéditeur du message <oa> correspondant à une entrée trouvée dans le répertoire du mobile (voir commande associée +CSCS).
- <da> Adresse du destinataire du message. Le type d'adresse utilisé est donné par <toda>.
- <data> Correspond au champ User Data qui stocke le corps du message.
- <dcs> *Data Coding Scheme*, indique l'alphabet utilisé pour composer le champ <data> et la classe du message.
- <dt> *Discharge Time* respectant le format : "yy/MM/dd, hh:mm:ss±zz", où les caractères indiquent l'année (deux derniers chiffres), mois, jour, heure, minutes, secondes et le décalage horaire.
- <fo> Indique le type de message :

<fo>	Description
17 _{dec}	SMS-DELIVER ou SMS-SUBMIT
2 _{dec}	SMS-STATUS-REPORT ou SMS-COMMAND
- <length> Indique la longueur du champ <data>. En mode TEXT il indique le nombre de caractères, en mode PDU il indique le nombre d'octets.

<code><mr></code>	Référence du message (nombre entier compris entre 0 et 255_{dec}).
<code><oa></code>	Adresse de l'émetteur du message.
<code><pdu></code>	Contient les informations relatives au type de PDU.
<code><pid></code>	Protocole Identifier, indique à quel type de service télématique est destiné le message, par défaut, il est à 0, le message est donc traité comme un SMS.
<code><sca></code>	Adresse du centre de messagerie.
<code><scts></code>	Service Centre Time Stamp contient la date et l'heure à laquelle le SMS est arrivé au SMSC (voir <code><dt></code>).
<code><toda></code>	Type de l'adresse de destination, quand le premier caractère de <code><da></code> est + on a <code><toda>=145_{dec}</code> (numéro international) sinon <code><toda>=129_{dec}</code> (numéro national).
<code><tooa></code>	Type de l'adresse de l'émetteur du message, quand le premier caractère de <code><da></code> est + on a <code><tooa>=145_{dec}</code> (numéro international) sinon <code><tooa>=129_{dec}</code> (numéro national).
<code><tosca></code>	Format de codage de l'adresse du centre de messagerie, ce paramètre est facultatif, s'il est omis les numéros téléphoniques nationaux/internationaux sont reconnus par le caractère « + » qui précède le numéro.
<code><vp></code>	Nombre compris entre 0 et 255_{dec} indiquant la durée de validité du message.

Commandes AT pour la gestion des SMS

AT+CSMS : Sélection du service de messagerie	
Commande de test AT+CSMS=?	Réponse +CSMS: <liste des <service>s supportés> Paramètres <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande de lecture AT+CSMS?	Réponse +CSMS: <service>,<mt>,<mo> Retourne les types de messages supportés par le ME : <mt> pour le mobile qui reçoit le message, <mo> pour le mobile qui émet le message. Paramètres <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande d'écriture AT+CSMS= <service>	Réponse +CSMS: <mt>,<mo> Paramètres <service>: 0 GSM 03.40 et 03.41 1...127 réservé 128... spécifique au constructeur <mt>,<mo>: 0 type non supporté 1 type supporté si erreur +CMS ERROR <err>

INTERFACES GSM

AT+CPMS : Sélection de la zone mémoire pour le stockage des SMS	
Commande de test AT+CPMS=?	Réponse +CPMS: (<mem1>s supportées), (<mem2>s supportées), (<mem3>s supportées) Liste pour chaque type de mémoire si erreur +CMS ERROR <err>
Commande de lecture AT+CPMS?	Réponse +CPMS: <mem1>,<used1>,<total1>,<mem2>,<used2>,<total2>, <mem3>,<used3>,<total3> Affiche pour chaque type de mémoire <memx> l'espace utilisé <usedx> et la capacité de stockage <totalx> si erreur +CMS ERROR: <err>
Commande d'écriture +CPMS= <mem1> [,<mem2>[,<mem3>]]	Réponse +CPMS: <used1>,<total1>,<used2>,<total2>,<used3>,<total3> Affecte une mémoire aux paramètres <mem1>, <mem2> et <mem3> si erreur +CMS ERROR <err>

AT+CMGF : Sélectionne le format des SMS (PDU ou TEXT)	
Commande de test AT+CMGF=?	Réponse +CMGF: liste des <mode>s supportés OK Paramètre <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande de lecture AT+CMGF?	Réponse +CMGF: <mode> OK Paramètre <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande d'écriture AT+CMGF= [<mode>]	Réponse OK sélectionne le format de dialogue utilisé Paramètre <mode> : 0 mode PDU (mode disponible sur tous les types de mobile) 1 mode TEXT si erreur +CMS ERROR <err>

AT+CSCA : Adresse du centre de messagerie (SMSC)	
Commande de test AT+CSCA=?	Réponse OK si erreur +CMS ERROR <err>
Commande de lecture AT+CSCA?	Réponse +CSCA: <sca>,<tosca> OK si erreur +CMS ERROR <err>
Commande d'écriture Si mode TEXT (+CMGF=1) : AT+CSCA= <sca>[,<tosca>]	Réponse OK Définit l'adresse du centre de messagerie à utiliser pour l'envoi des SMS si erreur +CMS ERROR <err>

AT+CSDH : Affiche en mode TEXT le paramétrage des SMS	
Commande de test AT+CSDH=?	Réponse +CSDH: (liste des <show>s supportés) Paramètre <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande de lecture AT+CSDH?	Réponse +CSDH: <show> Paramètre <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande d'écriture AT+CSDH= [<show>]	Réponse OK Paramètre <show> : 0 n'affiche pas tout le paramétrage du SMS, concerne les commandes +CSCA et +CSMP (<sca>, <tosca>, <fo>, <vp>, <pid> et <dcs>) sinon <length>, <toda> ou <tooa> pour +CMT, +CMGL, +CMGR 1 affiche tout le paramétrage du SMS si erreur +CMS ERROR <err>

INTERFACES GSM

AT+CSAS : Sauvegarde du paramétrage en cours	
Commande de test AT+CSAS=?	Réponse +CRES: liste des <profile>s supportés si erreur +CMS ERROR <err>
Commande d'écriture AT+CSAS= [<profile>]	Sauvegarde en mémoire du paramétrage du service de messagerie en cours (concerne la commande +CSCA). Un mobile peut contenir dans sa mémoire volatile jusqu'à 255 profiles différents. si erreur +CMS ERROR <err>

AT+CRES : Restauration du paramétrage par défaut	
Commande de test AT+CRES=?	Réponse +CRES: liste des <profile>s supportés si erreur +CMS ERROR <err>
Commande d'écriture AT+CRES= [<profile>]	Restauration d'un des paramétrages du service de messagerie (concerne la commande +CSCA) stockés en mémoire. Un mobile peut contenir dans sa mémoire volatile jusqu'à 255 profiles différents définis à l'aide de la commande +CSAS. si erreur +CMS ERROR <err>

AT+CNMI : Indication concernant un nouveau SMS	
Commande de test AT+CNMI=?	Réponse +CNMI: (liste des <mode>s supportés), (liste des <mtn>s supportés) Paramètres <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande de lecture AT+CNMI?	Réponse +CNMI: <mode>,<mtn> Paramètres <i>Voir commande d'écriture</i> si erreur +CMS ERROR <err>
Commande d'écriture +CNMI= [<mode> [,<mtn>]]	Réponse OK Détermine comment le mobile doit informer le TE lorsqu'un nouveau SMS arrive du réseau. <mode> : 0 Les indications concernant la réception d'un nouveau message sont stockées dans le TA. 1 Rejette les indications concernant la réception d'un nouveau message lorsque la liaison entre le TA et le TE est réservée. Sinon les indications sont directement transférées vers le TE. 2 Sauvegarde dans le TA les indications concernant la réception d'un nouveau message lorsque la liaison entre le TA et le TE est réservée. Lorsque la liaison est libre, les indications sont transférées vers le TE. 3 Les indications concernant la réception d'un nouveau message sont directement transférées vers le TE. <mtn> : 0 Aucune indication concernant le SMS-DELIVER n'est envoyée au TE. 1 Si le SMS-DELIVER est stocké dans le mobile, les indications concernant l'emplacement en mémoire du message sont envoyées au TE en utilisant le code +CMTI: <mem>,<index> 2 Les SMS-DELIVERs sont directement acheminés au TE en utilisant le code +CMT : [<alpha>],<length><CR><LF><pdu> 3 Les SMS-DELIVERs de classe 3 sont directement acheminés au TE en utilisant le code : +CMT: [<alpha>],<length><CR><LF><pdu> si erreur +CMS ERROR <err>

INTERFACES GSM

AT+CMGL : Liste les SMS stockés en mémoire

Commande de test AT+CMGL=?	Réponse +CMGL: liste des <stat>s supportés OK si erreur +CMS ERROR <err>
Commande d'écriture AT+CMGL= [<stat>]	Réponse Si mode PDU (AT+CMGF=0) et commande réussie +CMGL: <index>,<stat>,[<alpha>],<length><CR><LF><pdu> [<CR><LF>]+CMGL:<index>,<stat>,[<alpha>],<length><CR><LF> <pdu>[...] Si mode TEXT (AT+CMGF=1) et commande réussie +CMGL: <index>,<stat>,<a/da>,[<alpha>],[<scts>] [,<tooa/toda>,<length>]<CR><LF><data>[<CR><LF> +CMGL: <index>,<stat>,<da/oa>,[<alpha>],[<scts>] [,<tooa/toda>,<length>]<CR><LF><data>[...] Retourne tous les messages stockés avec leur état <stat> depuis la mémoire <mem1> Si l'état d'un message est « received unread » il devient « received read » si erreur +CMS ERROR <err>

AT+CMGR : Lecture d'un SMS

Commande de test +CMGR=?	Réponse OK si erreur +CMS ERROR <err>
Commande d'écriture +CMGR= <index>	Réponse Si mode PDU (AT+CMGF=0) et commande réussie +CMGR: <stat>,[<alpha>],<length><CR><LF><pdu> Si mode TEXT (AT+CMGF=1), commande réussie et SMS-DELIVER +CMGR: <stat>,<a>,[<alpha>],<scts> [,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]<CR> <LF><data> Si mode TEXT (AT+CMGF=1), commande réussie et SMS-SUBMIT +CMGR: <stat>,<da>,[<alpha>] [,<toda>,<fo>,<pid>,<dcs>,[<vp>],<sca>,<tosca>,<length>]<CR> <LF><data> Retourne le message ayant pour emplacement <index> dans la mémoire <mem1> Si l'état d'un message est « received unread » il devient « received read » si erreur +CMS ERROR <err>

AT+CMGS : Envoi d'un SMS	
Commande de test AT+CMGS=?	Réponse OK si erreur +CMS ERROR <err>
Commande d'écriture Si mode PDU (+CMGF=0) : +CMGS= <length><CR> trame PDU <ctrl-Z/ESC> Si mode TEXT (+CMGF=1) : +CMGS= <da>[,<toda>] <CR> texte <ctrl-Z/ESC>	Réponse +CMGS: <mr>[,<scts>] OK Envoi du SMS sur le réseau (SMS-SUBMIT). La référence du message <mr> est retournée au terminal émetteur si le message est correctement envoyé. Si erreur : +CMS ERROR: <err> Note : La fin du message est signalée par CTRL Z ESC annule l'envoi du message en cours, bien que le terminal retourne OK, le SMS n'est pas envoyé. Envoi d'e-mail par SMS : certains providers ne reconnaissent pas le symbole @, il est possible dans certains cas de le remplacer par !

AT+CMSS : Envoi d'un SMS stocké en mémoire	
Commande de test AT+CMSS=?	Réponse OK si erreur +CMS ERROR <err>
Commande d'écriture +CMSS=<index> [,<da>[,<toda>]]	Si l'envoi est réussi : +CMSS: <mr> Cette commande envoie le SMS (SMS-SUBMIT), situé à l'emplacement <index> de la mémoire <mem2> La référence du message <mr> est retournée au terminal émetteur si le message est correctement envoyé. si erreur +CMS ERROR <err>

INTERFACES GSM

AT+CMGW : Écriture d'un SMS en mémoire	
Commande de test AT+CMGW=?	Réponse OK si erreur +CMS ERROR <err>
Commande d'écriture Si mode PDU (+CMGF=0) : +CMGW= <length>[,stat] <CR> trame PDU <ctrl-Z/ESC> Si mode TEXT (+CMGF=1) : +CMGW=<oa/da> [,<tooa/toda> [,<stat>]]<CR> texte <ctrl-Z/ESC>	Réponse +CMGW: <index> L'exécution de cette commande effectue le stockage dans la mémoire <mem2> d'un message (SMS-DELIVER ou SMS-SUBMIT). L'emplacement du message dans la mémoire <index> est retourné. Par défaut l'état du message sera « stored unsent », mais le paramètre <stat> qui est facultatif autorise les autres valeurs possibles. si erreur +CMS ERROR <err>

AT+CMGD : Efface un SMS	
Commande de test AT+CMGD=?	Réponse OK si erreur +CMS ERROR <err>
Commande d'écriture AT+CMGD= <index>	Réponse OK Efface le message situé dans la mémoire <mem1> à l'emplacement défini par <index> si erreur +CMS ERROR <err>

Codes d'erreur

Lorsqu'une commande échoue, un code d'erreur <err> peut être renvoyé au TE (voir commande AT+CMEE) :

- 300échec ME
- 301service SMS du ME réservé
- 302opération non autorisée
- 303opération non supportée
- 304paramètre invalide (mode PDU)
- 305paramètre invalide (mode TEXT)
- 310SIM non insérée
- 311SIM PIN nécessaire
- 312PH-SIM PIN nécessaire
- 313échec SIM
- 314SIM occupée
- 315SIM faux
- 320échec mémoire
- 321index mémoire non valide
- 322mémoire pleine
- 330adresse SMSC inconnue
- 331pas de réseau
- 332timeout réseau
- 500erreur inconnue
- 501 à 511réservé
- 512spécifique au constructeur

Attention, certains éléments présentés ci avant sont extraits d'un document officiel de l'ETSI et par conséquent ils sont soumis à un copyright :

« © ETSI 1999. Further use, modification, redistribution is strictly prohibited. ETSI standards are available from:

<http://pda.etsi.org/pda/> and <http://www.etsi.org/eds/> »

3 MATÉRIELS UTILISÉS

3.1 Téléphones portables	56
3.2 Modules GSM intégrés	62

4	Interfacer un téléphone GSM	73
5	Réalisations électroniques	101
	Annexes	251
	Glossaire	261
	Bibliographie	264

3.1 TÉLÉPHONES PORTABLES

La majorité des téléphones portables disposent en interne d'un TA, dans ce cas il est possible de récupérer les lignes TxD et RxD disponibles sur un connecteur multibroche, propre à chaque modèle de téléphone. Un simple circuit adaptateur de niveau de tension TTL/RS232 permet alors de relier le téléphone au port série d'un PC. Bien entendu tout portable normalement constitué se doit de reconnaître les normes GSM07.07 et GSM07.05, ainsi il est possible via l'ordinateur d'accéder à toutes les fonctions du téléphone à l'aide des commandes AT que nous avons vues précédemment. Il est important de noter que *certaines modèles de portables ne supportent pas le mode TEXT*, la gestion des SMS entre l'ordinateur et le mobile se fait uniquement en mode PDU. Rappelons que la commande AT+CMGF=? permet de lister les modes supportés par le mobile, si la réponse est +CMGF : (0,1) cela signifie que les modes TEXT (0) et PDU (1) sont supportés. Si la réponse est +CMGF : (0), seul le mode PDU peut être utilisé.

Adaptateur TTL/RS232

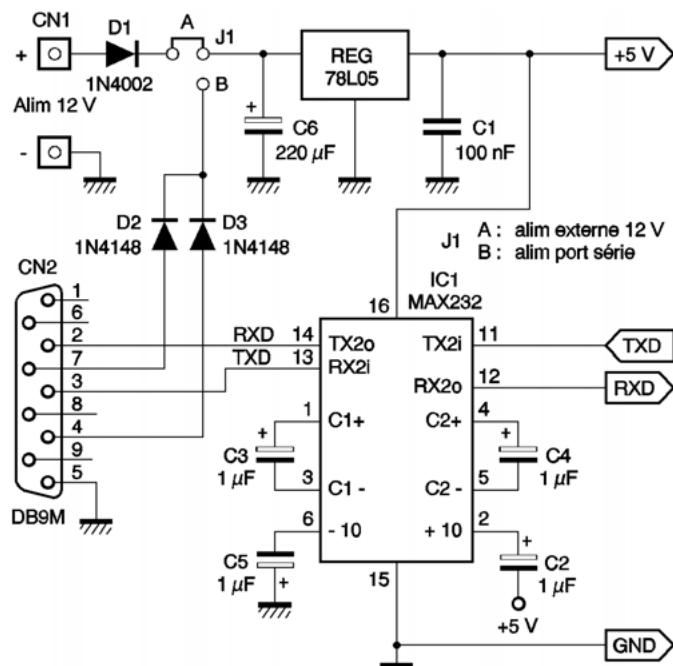


Figure 3.1.
Schéma
de l'adaptateur
TTL/RS232.

Nous allons faire appel au célèbre circuit intégré MAX232 qui, câblé avec ces 4 condensateurs au tantalum de $1 \mu\text{F}$, permet d'adapter

les niveaux de tension entre le PC et le téléphone. Paradoxalement le plus compliqué consiste à se procurer le connecteur pour relier le téléphone au montage. Il est possible de modifier un kit piéton encore faut-il que les broches qui nous intéressent soient présentes. Quelques exemples de brochages récupérés sur Internet vont vous permettre d'identifier les lignes à relier au montage. Attention les brochages présentés ici n'ont pas été vérifiés, leur utilisation est sous votre entière responsabilité. L'alimentation du montage peut se faire à partir des sorties DTR et RTS de l'ordinateur, dans ce cas le cavalier J1 est en position B. Comme certains PC ne disposent pas d'une puissance suffisante sur leur

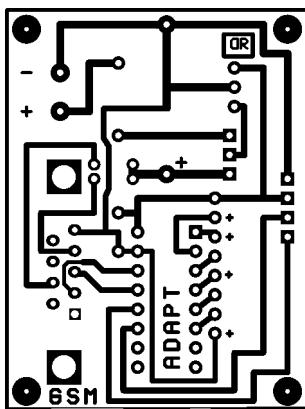


Figure 3.2.
Circuit imprimé.

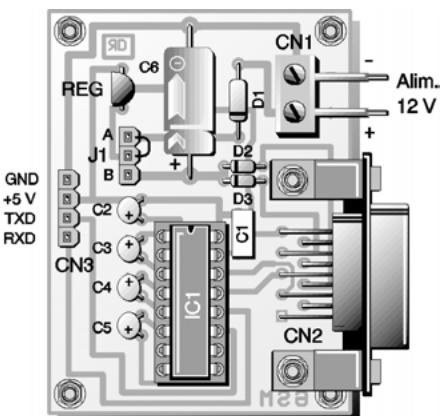


Figure 3.3.
Implantation
des composants.

Liste des composants

C1 : 100 nF / LCC jaune

C2 à C5 : 1 µF / tantalé / 15 V

C6 : 220 µF / électrolytique / 15 V

D1 : 1N4002

D2, D3 : 1N4148

REG : régulateur 78L05

J1 : barrette HE10 3 contacts
+ cavalier

CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle
pour CI/coudé 90°

CN3 : connecteur spécifique
au modèle de téléphone utilisé

IC1 : MAX232

+ support DIL 16 broches

INTERFACES GSM

	ALCATEL 1, 4, 9 : GND 2 : +5 V 3 : TXD 4 : RXD 7 : GND 8 : TXD
	ERICSSON 2XX - 3XX 8 : GND 11 : RXD 12 : TXD
	ERICSSON 6XX - 7XX 9 : TXD 10 : GND 11 : RXD 12 : +5 V
	ERICSSON 8XX - T18s 8 : +5 V 9 : TXD 10 : GND 11 : RXD
	ERICSSON T28/R320 6 : RXD 7 : TXD 10 : GND 11 : +3 V
	PANASONIC G400-600-450-500-520 10 : TXD 11 : RXD 16 : GND
	PANASONIC GD30-GD90 6 : RX 7 : TX 18 : GND
	PHILIPS SAWY 8 : RXD 9 : TXD 12 : GND
	SAGEM 612-615-715-725-730 14 : GND 10 : RXD 3 : TXD
	SAGEM 9XX - MYX-5 6 : GND 10 : RXD 11 : TXD
	SIEMENS S6/8 9 : GND 7 : RXD 15 : TXD
	SIEMENS C/S25/S35 1 : GND 5 : TXD 6 : RXD
	SIEMENS S10/S11 3 : RXD 8 : GND 14 : TXD

Figure 3.4.
Différents brochages.

port série, c'est le cas notamment des portables, il est possible de connecter sur le bornier CN1 une alimentation externe délivrant une tension de 12 V, dans ce cas le cavalier J1 est en position A. Dans les deux cas le régulateur 78L05 se charge de réguler à + 5 V la tension destinée au MAX232 mais aussi au téléphone, certains modèles ont besoin de cette tension pour activer leur port série. Il est possible de tester le montage avant même de l'avoir connecté au téléphone. Pour cela, reliez momentanément les lignes TXD et RXD, ainsi les données transmises sur TXD sont recopiées sur RXD. Un simple logiciel comme Hyper Terminal suffira à vérifier que le texte saisi à l'écran est renvoyé comme un écho par le montage.

Adaptateur pour FBUS/MBUS (ou M2BUS)

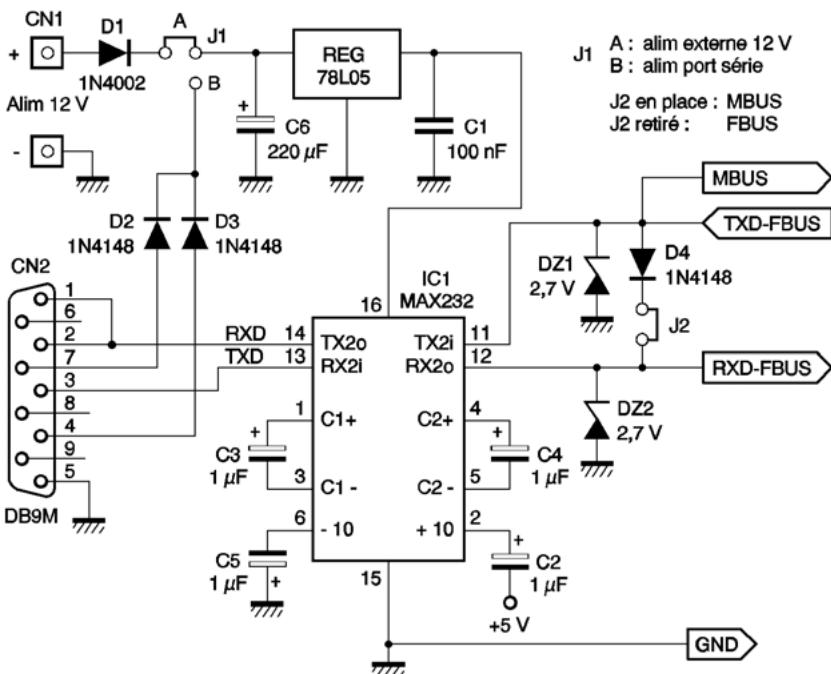


Figure 3.5.
 Schéma
 de l'adaptateur
 pour FBUS/MBUS.

Certains téléphones, notamment ceux de la marque Nokia, nécessitent une électronique légèrement différente pour communiquer avec un PC. En effet ils utilisent des protocoles de transmission FBUS et/ou MBUS propres à ce constructeur. Le FBUS est le mode de communication privilégié de l'utilisateur, il permet le transfert de données à une vitesse maximale de 115 kbauds, on retrouve les broches TXD et RXD mais les niveaux de tension ne doivent pas dépasser 3 V, d'où la présence des diodes zener DZ1

et DZ2 qui possèdent une tension de seuil de 2,7 V. Le MBUS est plutôt réservé au personnel technique de Nokia pour effectuer le paramétrage système du téléphone, bien qu'il puisse être aussi utilisé pour le transfert de données mais avec une vitesse limitée à 9 600 bauds. La transmission et la réception se font sur une seule broche nommée MBUS.

Le montage présenté ici peut communiquer selon les deux protocoles FBUS (cavalier J2 retiré) ou MBUS (cavalier J2 en place). Il est possible de tester le montage avant même de l'avoir relié au téléphone. En effet, lorsque rien n'est connecté au bus, les lignes RXD et TXD se trouvent reliées par la diode D4 de sorte que toute information envoyée sur TXD est retransmise sur RXD (si le cavalier J2 en place).

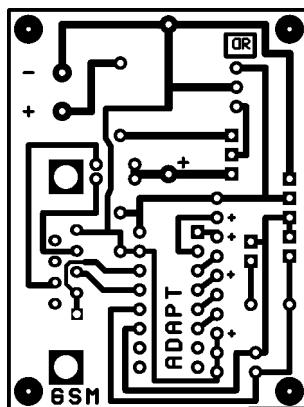


Figure 3.6.
Circuit imprimé.

Figure 3.7.
Implantation
des composants.

Liste des composants

C1 : 100 nF / LCC jaune

C2 à C5 : 1 µF / tantale / 15 V

C6 : 220 µF / électrolytique / 15 V

D1 : 1N4002

D2 à D4 : 1N4148

DZ1, DZ2 : diode zener 2,7 V

REG : régulateur 78L05

J1 : barrette HE10 3 contacts + cavalier

J2 : barrette HE10 2 contacts + cavalier

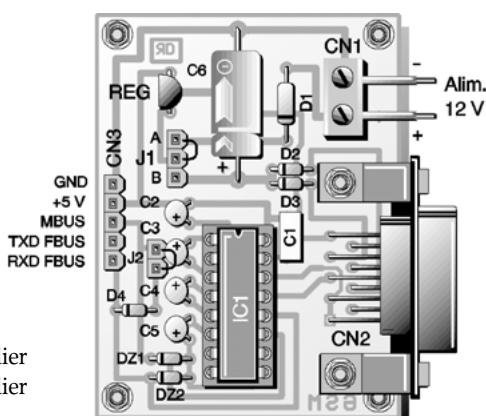
CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle pour CI coudé 90°

CN3 : connecteur spécifique

au modèle de téléphone utilisé

IC1 : MAX232 + support DIL 16 broches



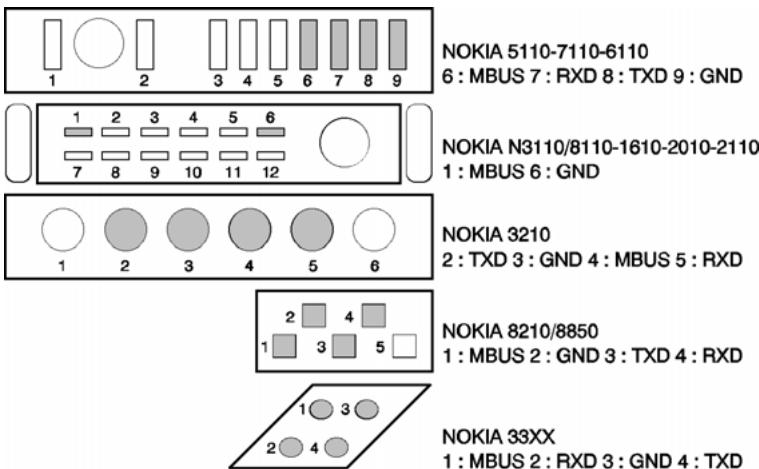


Figure 3.8.
Brochages
concernant
les téléphones
de la marque
NOKIA.

Remarque : pour relier l'adaptateur TTL/RS232 ou l'adaptateur pour FBUS/MBUS (ou M2BUS) au port série d'un ordinateur il faut utiliser un câble RS232 « droit », la ligne TXD de l'adaptateur doit être connectée sur la ligne TXD du PC et la ligne RXD de l'adaptateur doit être connectée sur la ligne RXD du PC (voir figure 3.9).

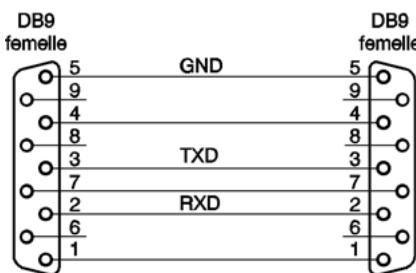


Figure 3.9.
Câble adaptateur.

Cordons DATA

Si vous n'êtes pas convaincu du brochage trouvé sur Internet et que vous avez peur d'abîmer votre téléphone, il est possible d'acheter un câble tout prêt. Ces câbles appellés DATA ou DATA LINK sont malheureusement difficiles à se procurer dans le commerce traditionnel. Encore une fois Internet vient à notre secours, sur le site www.maisondugsm.com vous trouverez certainement le câble correspondant à votre modèle de téléphone, avec la possibilité de commander par correspondance si le paiement en ligne vous rebute. Ces câbles disposent en interne de toute la circuiterie permettant une liaison directe entre votre téléphone et

le port série du PC. L'énergie nécessaire à la mise à niveaux des signaux est prélevée, en principe, de la ligne DTR, ce qui rend le câble totalement autonome. Notez qu'il existe des câbles permettant de connecter simultanément une alimentation externe afin de recharger la batterie du téléphone.

3.2 MODULES GSM INTÉGRÉS

On trouve désormais sur le marché des téléphones GSM intégrés débarrassés de leurs interfaces homme-machine, ne subsiste que la partie interface machine-machine (M2M), qui physiquement correspond à un connecteur multibroche quelconque ou encore à un connecteur DB9 facilitant la connexion à un PC. Ces modules sont universels puisqu'ils supportent les normes GSM07.07 et GSM07.05 décrites précédemment et permettent de ce fait l'échange de données, de SMS, d'emails et même de télécopies (FAX) via le réseau de téléphonie mobile. Leur simplicité de mise en œuvre ouvre des perspectives très intéressantes concernant la réalisation de montages électroniques sans fil.

La société Lextronic propose à la vente via son site Internet plusieurs modèles de modules GSM intégrés l'adresse <http://www.lextronic.fr/R203-modules-gsm--gprs.html>

Le TM2 de TELTONIKA

Parmi les différents modèles proposés par Lextronic notre choix s'est porté sur le TM2 fabriqué par la société Teltonika. Il s'agit d'un modèle quadri bandes qui utilise les fréquences 850, 900, 1 800 et 1 900 MHz. Il est capable de fonctionner dans les modes voix, données, FAX et surtout, le plus intéressant pour nous, dans le mode SMS. Le module dispose d'un support destiné à recevoir l'indispensable carte SIM et un connecteur MMCX permettant de relier une petite antenne RF également fournie par Lextronic. Toutes les entrées et sorties utiles au pilotage du module sont disponibles sur un connecteur comportant 60 points en CMS. Inutile d'espérer souder vous-même un tel composant. Heureusement Lextronic propose un adaptateur qui répartit l'ensemble des connexions sur 4 rangées de 15 points au pas classique de 2,54mm.

Nous avons ajouté au module TM2 une interface au format RS232 afin de pouvoir facilement le connecter au port série d'un PC et surtout à toutes les réalisations présentées dans le chapitre 5. Une fois de plus nous faisons donc appel au traditionnel MAX232 pour l'adaptation des niveaux de tensions des lignes TXD0 et RXD0. Comme le TM2 utilise des niveaux de tension de

Transmission	Voix, données et SMS
Alimentation	3.5 V à 4.2 V, typiquement 3.8 V
Bandes de fréquences	GSM 850 MHz, EGSM 900 MHz, DCS 1800 MHz, PCS 1900 MHz
Courant absorbé	GSM900 : 147 mA (900 mA max) GSM1800 : 127 mA (700 mA max) GSM1900 : 113 mA (650 mA max)
Puissance d'émission	Class 4 (2 W) pour bandes GSM/EGSM Class 1 (1 W) pour bandes DCS/PCS
Lecteur de carte SIM	Intégré au module, supporte les cartes SIM 3,3v et 1,8v
Antenne	Externe par connecteur MMCX
Interfaces	Connecteur 60 points (CIVILUX CBRB060PC2000R0) : Audio, (2x analog, 1x digital), I2C bus, SPI bus, 2x ADC, 2x analog out (PWM), 12 GPIOs et 2 port série de type UART
Normes respectées	GSM07.07 et GSM07.05
Modes SMS	PDU et TEXT
GPRS Data Services	GPRS multi-slot class (MSC) 10 (4+1, 3+2), GPRS PBCCH/PCCCH support, GPRS Class B and CC
FAX	G3, Classe 2.0
Température d'utilisation	- 20 °C à + 55 °C
Taille	33,5 mm x 38,8 mm x 5,6 mm
Masse	< 10 g

Tableau 3.1.
Caractéristiques principales

+3,3v il faut prendre soin d'abaisser la tension fournie par la sortie TXD du MAX232 à l'aide d'un simple pont diviseur de tension constitué par les résistances R1 et R2.

L'entrée KIN2 est reliée au GND ainsi le TM2 devient actif dès sa mise sous tension.

Le module TM2 doit être alimenté par une tension de +3,8v via ses entrées VBAT. Nous utilisons un régulateur de tension LM317 (REG2) qui délivre une tension fonction de la résistance du potentiomètre P1. Avant même d'insérer le module TM2 il est impératif de régler le potentiomètre P1 jusqu'à l'obtention d'une tension de +3,8v entre la broche VBAT et la broche GND.

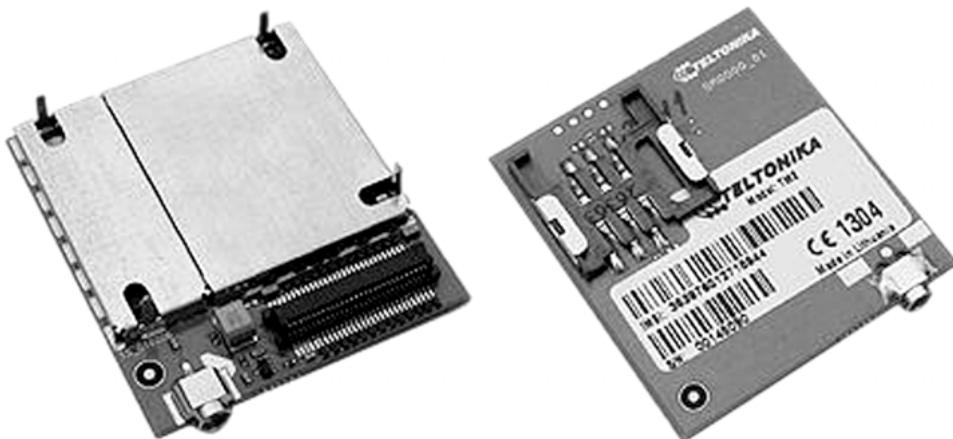


Figure 3.10.
Le module TM2 vu
de dessous (à gauche)
et de dessus
(à droite).

Une petite chute de plaque d'aluminium fera office de dissipateur thermique pour REG2.

Un deuxième régulateur 78L05 (REG1) est nécessaire pour alimenter le circuit MAX232 avec une tension plus conventionnelle de +5v.

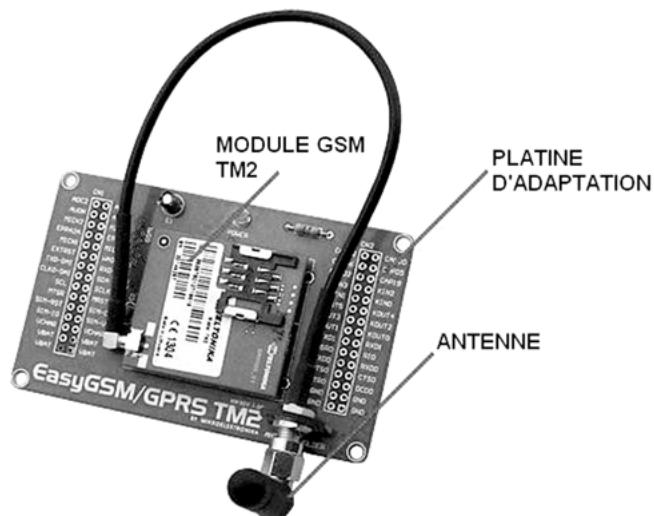
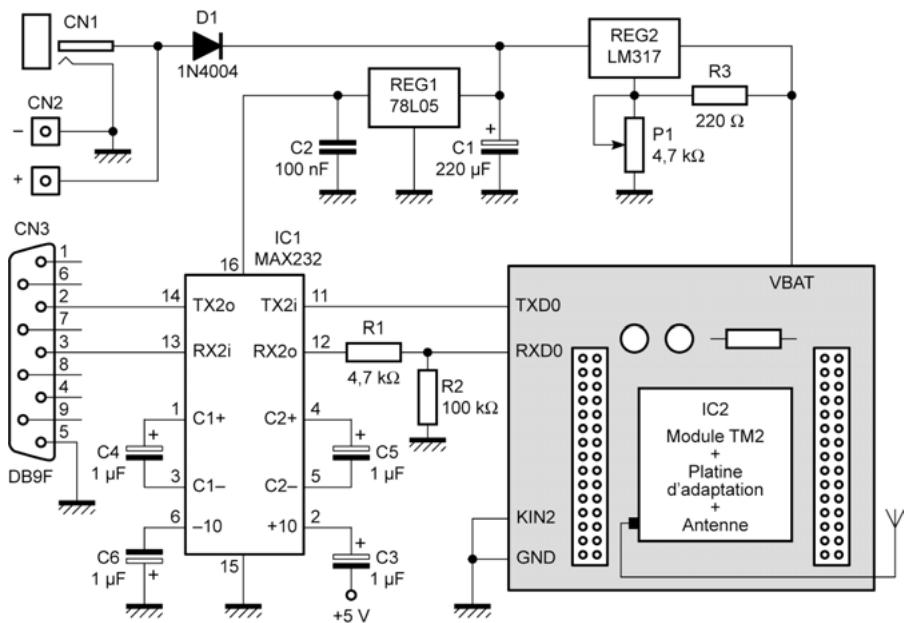


Figure 3.11.
Le module TM2 avec
son antenne sur sa
platine d'adaptation.

Compte tenu de l'intensité absorbée par le module GSM notamment lors des phases de recherche de réseau, il conviendra d'utiliser un bloc alimentation secteur délivrant au moins une intensité de 1A pour une tension continue comprise entre 9 et 12v. Une prise jack (CN1) femelle permet de relier facilement n'importe quel bloc du commerce. L'alimentation est reprise sur le connecteur CN2 afin d'alimenter le montage hôte, un de ceux présentés dans le chapitre 5.


 Figure 3.12.
 Schéma électrique.

Seules les broches utiles au montage sont reliées à la carte d'adaptation via 4 connecteurs CN4 à CN7 de type HE10. Comme les broches VBAT et GND sont déjà interconnectées sur la carte d'adaptation il n'est pas utile de toutes les reliées à notre montage.

INTERFACES GSM

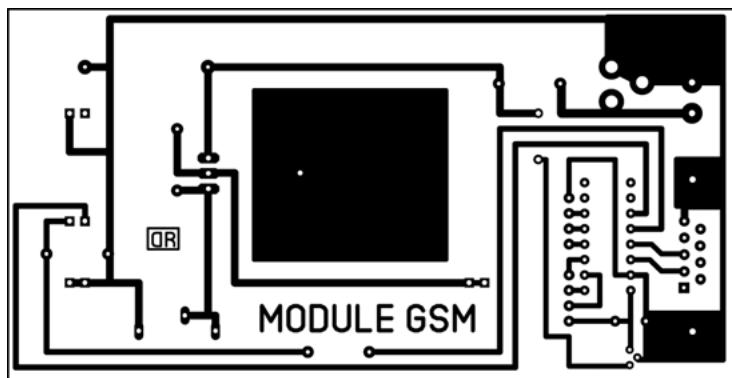


Figure 3.13.
Circuit imprimé.

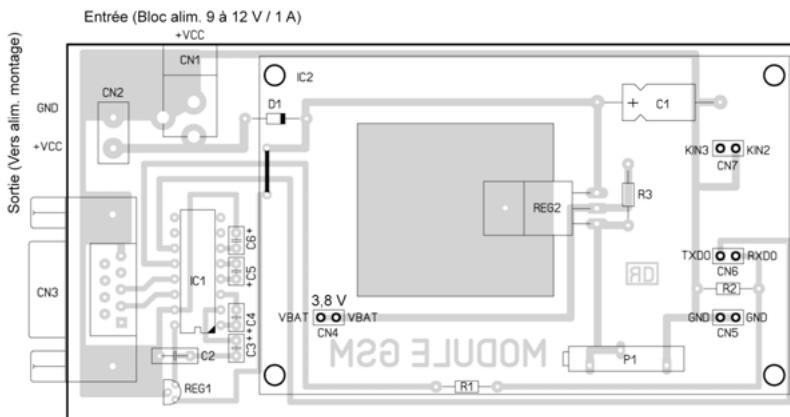


Figure 3.14.
Implantation
des
composants.

Liste des composants

- R1 : 4,7 kΩ
- R2 : 100 kΩ
- R3 : 220 Ω (précision 1 %)
- P1 : potentiomètre multi-tours horizontal/4,7 kΩ
- C1 : 220 µF/électrolytique/25v horizontal
- C2 : 100 nF/LCC jaune
- C3 à C6 : 1 µF/tantale/15v
- D1 : diode 1N4004
- IC1 : MAX232 + support DIL 16 broches
- IC2 : module GSM TM2 + platine d'adaptation + antenne (www.Lextronic.fr)
- REG1 : 78L05
- REG2 : LM317
- CN1 : prise jack femelle pour CI
- CN2 : bornier 2 plots
- CN3 : connecteur DB9 femelle
- CN4 à CN7 : connecteur HE10

Mise en œuvre

Le montage peut être connecté au port série d'un ordinateur de type PC. Dans cette configuration le chapitre 4 vous montrera comment utiliser le logiciel Hyper terminal pour envoyer des commandes AT au module TM2.

La connexion du module TM2 peut s'effectuer directement au port série du PC ou par l'intermédiaire d'un câble « droit » constitué de deux connecteurs DB9 mâle et femelle.

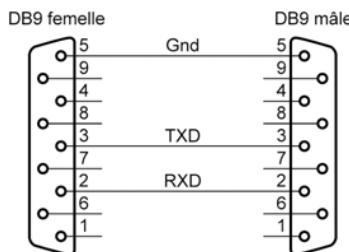


Figure 3.15.
Câble « droit »
constitué de deux
connecteurs DB9 mâle
et femelle.

Le montage peut également être directement connecté au port série d'une des 5 réalisations présentées au chapitre 5.

Commandes AT spécifiques au module TM2

En supplément des commandes AT présentées au chapitre 2, il est nécessaire de connaître les commandes détaillées ci-après. Elles nous seront utiles à la fin du chapitre suivant afin de paramétriser la vitesse de transmission entre le module TM2 (TA) et le montage hôte (TE).

AT+IPR : Définit la vitesse de transmission série

Tableau 3.2.

AT+IPR : Définit la vitesse de transmission série	
Commande de test AT+IPR=?	Réponse +IPR : liste des <vitesses> supportées pour la transmissions des données entre la TA et le TE 0, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps. 115200 est le réglage sortie d'usine. 0 signifie que le TA se synchronise automatiquement sur la vitesse du TE
Commande de lecture AT+IPR?	Réponse +IPR : <vitesse> OK Retourne la vitesse de transmission en cours si erreur +CME ERROR : <error>
Commande d'écriture AT+IPR=<vitesse>	Réponse OK Définit la vitesse de transmission si erreur +CME ERROR : <error>

INTERFACES GSM

Tableau 3.3.

AT&W : sauvegarde la configuration en cours	
Commande d'écriture AT&W [<value>]	<p>Réponse</p> <p>OK</p> <p>Cette commande enregistre en mémoire la configuration active du TA dans un des deux profils utilisateurs.</p> <p><value>=0 premier profil (valeur par défaut)</p> <p><value>=1 deuxième profil</p> <p>si erreur</p> <p>+CME ERROR : <error></p> <p><i>Nota : cette commande doit être utilisée conjointement avec la commande AT+CPWROFF pour que la mémorisation du profil soit effective</i></p> <p>Liste des paramètres mémorisés dans le profil utilisateur :</p> <ul style="list-style-type: none">– AT&C : Circuit 109 behavior ;– AT&D : Circuit 108 behavior ;– AT&K : Flow control ;– ATE : Echo mode ;– ATQ : Response Suppression Mode ;– ATV : Response Formatting Mode ;– ATX : Call Progress Monitoring Control ;– ATS0 : Automatic answer ;– ATS2 : Escape character ;– ATS3 : Command line termination character ;– ATS4 : Response formatting character ;– ATS5 : Command line editing character ;– ATS7 : Connection completion timeout ;– AT+CBST : Data Rate, Bearer Service, Connection Element ;– AT+CRLP : RlpIws (IWF to MS window size), RlpMws (MS to IWF window size), Rlp96T1 (acknowledgement timer T1), RlpN2 (retransmission attempts N2) ;– AT+CR : Service Report Control Mode ;

AT&W : sauvegarde la configuration en cours

Tableau 3.3. (suite)

- AT+CRC : Cellular Result Mode ;
- +BR (+IPR saved value) : Baud Rate ;
- AT+COPS : Cops mode, Cops PLMN to Register ;
- AT+NMGC : Microphone Gain Control ;
- AT+NSGC : Speaker Gain Control ;
- AT+NSTN : Sidetone ;
- AT+NUBF : Uplink Biquad Filters ;
- AT+NDBF : Downlink Biquad Filters ;
- AT+NHFP : Hand Free Parameters ;
- AT+ICF : DTE-DCE character framing ;

AT&V : affiche la configuration en cours

Tableau 3.4.

AT&V[<value>]	Réponse ACTIVE PROFILE : &C1, &D1, &K3, E1, Q0, V1, X4, S00 : 000, S02 : 043, S03 : 013, S04 : 010, S05 : 008, S07 : 060, +CBST : 007, 000, 001, +CRLP : 061, 061, 048, 006, +CR : 000, +CRC : 000, +BR : 57600, +COPS : 2, FFFF STORED PROFILE 0 : &C1, &D1, &K3, E1, Q0, V1, X4, S00 : 000, S02 : 043, S03 : 013, S04 : 010, S05 : 008, S07 : 060, +CBST : 007, 000, 001, +CRLP : 061, 061, 048, 006, +CR : 000, +CRC : 000, +BR : 115200, +COPS : 2, FFFF STORED PROFILE 1 : &C1, &D1, &K3, E1, Q0, V1, X4, S00 : 000, S02 : 043, S03 : 013, S04 : 010, S05 : 008, S07 : 060, +CBST : 007, 000, 001, +CRLP : 061, 061, 048, 006, +CR : 000, +CRC : 000, +BR : 115200, +COPS : 2, FFFF OK Cette commande retourne la configuration en cours ainsi que les configurations stockées dans les profiles utilisateurs. si erreur +CME ERROR : <error>
---------------	--

Tableau 3.5.

AT&Y : Définit le profile chargé à la suite d'un RESET	
AT&Y[<value>]	Réponse OK <value>=0 premier profile (valeur par défaut) <value>=1 deuxième profile si erreur +CME ERROR : <error>

Pour aller plus loin...

Le module TM2 est aussi capable de mettre en œuvre des connexions de type GPRS, de fait il est possible d'ouvrir des sockets permettant le transfert de données suivant le protocole TCP/IP.

La configuration en mode client ou serveur se fait à l'aide de commandes AT propres au module TM2.

Sans trop rentrer dans les détails nous vous présentons deux exemples de configuration tirés de la datasheet du constructeur Teltonika AT commands.

Configuration du profile GPRS

Tout d'abord il faut configurer un profile GPRS à partir des informations de votre opérateur téléphonique (SFR dans cet exemple).

Tableau 3.6.

at+npsd=0,1,"websfr" OK	Création d'un profile GPRS n° 0 Code APN : websfr
at+npsd=0,2," " OK	Nom d'utilisateur (vide)
at+npsd=0,3," " OK	Mot de passe (vide)
at+npsd=0,4,"172.20.2.10" OK	DNS1
at+npsd=0,5,"194.6.128.4" OK	DNS2
at+npsd=0,7,"0.0.0.0" OK	Adresse IP (0.0.0.0 ⇔ dynamique)

Création d'un socket serveur

at+npsda=0,3 OK at+npsnd=0,0 +NPSND : 0,0,"217.201.129.34" OK at+nsocr=6 +NSOCR : 0 OK at+nsoli=0,80 OK +NUSOLI : 1,"151.9.34.66",9882	Active le profile GPRS n° 0 Obtention d'une adresse IP dynamique Adresse IP serveur=217.201.129.34 Création d'un soket 0 = n° du soket TCP Association du socket au port 80 Affichage de l'adresse IP du client qui tente de se connecter
+NUSORD : 1,28 at+nsord=1,28	Indication qu'il existe 28 octets en provenance du client Lecture de ces 28 octets
+NSORD : 1,28,"data sent from telnet client" OK at+nsowr=1,29,"sending data to telnet client" +NSOWR : 1,29 OK at+nsocl=1 OK at+nsocl=0 OK at+npsda=0,4 OK	Réponse Ecriture de 29 octets à destination du client Fermeture du socket 1 Fermeture du socket 0 Fermeture du profile GPRS n° 0

Tableau 3.7.

■ *Création d'un socket client*

Tableau 3.8.

at+npsda=0,3 OK at+nsocr=6 +NSOCR : 0 OK at+nsoco=0,"151.9.34.66",80 OK at+nsowr=0,18,"data to be written" +NSOWR : 0,18 OK +NUSORD : 0,18 at+nsord=0,8 +NSORD : 0,8,"data to " OK +NUSORD : 0,10 at+nsord=0,10 +NSORD : 0,10,"be written" OK at+nsocl=0 OK at+npsda=0,4	Active le profile GPRS n° 0 Création d'un soket TCP 0 = n° du soket Adresse IP serveur=151.9.34.66 et port=80 associés au socket n°0 Ecriture de 18 octets à destination du serveur Confirmation d'écriture Notification de réception de 18 octets émis par le serveur Lecture des 8 premiers octets Notification qu'il existe 10 octets restant Lecture des 10 octets Fermeture du socket 0 Fermeture du profile GPRS n° 0
--	---

■ Pour plus d'informations sur la mise en œuvre de montages autonomes utilisant la technologie TCP/IP pour communiquer, nous vous renvoyons à l'ouvrage *Contrôle, commande et mesure via Internet* disponible aux éditions DUNOD.

4

INTERFACER UN TÉLÉPHONE GSM

4.1 Avec un PC	74
4.2 Avec un PicBasic	92

5	Réalisations électroniques	101
	Annexes	251
	Glossaire	261
	Bibliographie	264

Ce chapitre va vous montrer comment interfaçer facilement un téléphone (ou terminal) GSM. Dans un premier temps nous utiliserons un ordinateur de type PC en guise de TE (*Terminal Equipment*), les commandes AT vues précédemment saisies à partir du logiciel Hyper Terminal seront envoyées via le port série. Dans un deuxième temps c'est un simple microcontrôleur PicBasic beaucoup plus compacte qui nous permettra, en matière d'envoi et de réception de SMS, d'atteindre les mêmes résultats que le PC.

4.1 AVEC UN PC

Le logiciel Hyper Terminal livré en standard avec Windows est utilisé pour envoyer les commandes AT tirées des normes GSM07.05 et GSM07.07. Un logiciel maison développé sous DELPHI vous permettra d'envoyer, de recevoir, et de consulter facilement vos SMS.

Matériel

Téléphone GSM standard

- un câble spécifique, du commerce, ou que vous fabriquerez vous-même (adaptateur RS232/TTL), comme nous l'avons vu dans le chapitre 3 ;
- une carte SIM pour vous connecter au réseau GSM (vous pouvez utiliser une carte prépayée telle que mobicarte) ;
- un PC disposant d'un port série libre (exemple : COM2 ou COM1). Nota : si votre ordinateur n'a pas de port série vous pouvez utiliser un convertisseur USB-Série du commerce ;
- un logiciel Windows de type Hyper Terminal (livré en standard avec toutes les versions de Windows).

Terminal GSM : TM2 de Teltonika

- une carte SIM pour vous connecter au réseau GSM (vous pouvez utiliser une carte prépayée telle que mobicarte) ;
- une antenne GSM (fournie par le fabricant) ;
- un bloc secteur pour l'alimentation (9 à 12 v / 1 A) ;
- un PC disposant d'un port série libre (exemple : COM2 ou COM1). Nota : si votre ordinateur n'a pas de port série vous pouvez utiliser un convertisseur USB-Série du commerce ;
- un logiciel Windows de type Hyper Terminal (livré en standard avec toutes les versions de Windows).

Attention, veillez à mettre le téléphone ou le terminal hors tension avant d'insérer ou de retirer la carte SIM de son lecteur.

Hyper Terminal

Ouvrez une session du logiciel Hyper Terminal généralement situé sous le répertoire « C:\Program Files\Windows ». Vous pouvez créer un raccourci sur le bureau pour un accès ultérieur plus rapide.



Figure 4.1.
Description de la connexion.

Dans la fenêtre « Description de la connexion » (**figure 4.1**) spécifiez un nom pour la connexion que vous allez créer. Choisissez également une icône qui sera associée à la connexion.

Dans la fenêtre « Numéro de téléphone » (**figure 4.2**) sélectionnez dans la liste déroulante « Se connecter en utilisant » le port COM1 ou COM2. Les autres listes de la fenêtre qui sont utilisées uniquement avec un modem doivent se griser.

La fenêtre Propriétés de COMx (**figure 4.3**) permet de configurer le protocole de transfert. La vitesse de transmission définie en bits par seconde (ou bauds) est fixée à 9 600. Cette valeur n'est, en principe, pas primordiale pour un téléphone GSM car celui-ci est prévu pour se synchroniser sur la vitesse de transmission du TE. Nous avons volontairement choisi une vitesse ni trop rapide ni trop lente, susceptible d'être acceptée par tous les modèles de téléphone. Les autres paramètres 8 bits de données, pas de parité et 1 bit de stop correspondent à la configuration par défaut. Aucun contrôle de flux n'est spécifié, il suffira d'attendre la confirmation de traitement de la commande saisie avant d'envoyer la suivante. Lorsque vous cliquez sur la touche OK la connexion est établie.

INTERFACES GSM



Figure 4.2.
Numéro de téléphone.

Attention : concernant le module GSM TM2 vous devez utiliser une vitesse de 115 200 bds. Nous verrons dans le paragraphe « Commandes spécifiques au TM2 de Teltonika » comment ramener cette vitesse à 9 600 bds.

Remarque : il n'est pas utile de paramétriser le logiciel pour que les commandes saisies au clavier s'affichent à l'écran car chaque caractère saisi est automatiquement renvoyé par le ME en écho.



Figure 4.3.
Propriétés de COM2.

Ceci permet de contrôler que la commande est correctement réceptionnée par le ME. On constate d'ailleurs un très léger retard entre la saisie d'une commande et son affichage à l'écran. On constate aussi que si une nouvelle commande ne commence pas par les caractères "AT+" les caractères saisis ne sont pas affichés à l'écran, donc refusés par le ME.

Pour tester la liaison vous pouvez utiliser la commande la plus simple qui soit :

AT [ENTREE]

Rappelons que la touche [ENTREE] ou Return du clavier correspond au caractère <CR>, à l'écran il se traduit par un retour à la ligne.

Si la liaison est établie le mobile doit simplement répondre par :

OK

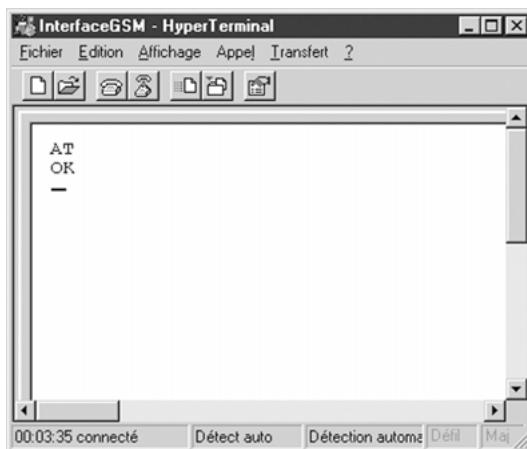


Figure 4.4.
Hyper Terminal.

Lorsque l'on utilise un logiciel informatique comme Hyper Terminal pour envoyer ou recevoir des données via le port série du PC, les caractères saisis à l'écran sont codés suivant la table des caractères ASCII. Par exemple si vous tapez la lettre majuscule A c'est le code 1000001_{bin} qui est envoyé au système connecté sur le port série. Il est donc impératif que ce système utilise la même table pour convertir la donnée réceptionnée. Dans notre cas, le téléphone GSM peut utiliser différents alphabets citons par exemple : IRA, GSM, HEX... malheureusement l'alphabet ASCII n'est pas supporté. Cependant, si l'on se limite à l'utilisation des caractères « usuels » (A...Z, a...z, 0...9,...) la compatibilité est assurée. Il faut rester prudent dans l'utilisation des autres caractères, notamment les caractères accentués. Pour les utilisateurs du Terminal TC35

(ou MC35) il existe une solution pour envoyer les caractères non compatibles, il faut saisir un anti-slash suivi du caractère ASCII ; voir quelques exemples au **tableau 4.1.**

Tableau 4.1.

Caractère GSM	Valeur hexa. du caractère GSM	Caractère ASCII	Caractères ASCII de remplacement	Codes hexa. des caractères de remplacement
@	00	(null)	\00	5C 30 30
\$	02		\02	5C 30
ù	06		\	5C
à	7F		\	5C

Remarque : pour connaître tous les caractères incompatibles entre l'alphabet GSM et l'alphabet ASCII il suffit de comparer les deux tableaux situés en Annexes.

La commande AT+CSCS permet de sélectionner un alphabet. Il est recommandé d'utiliser l'alphabet GSM (en principe c'est l'alphabet configuré par défaut par les fabricants) qui est théoriquement supporté par tous les téléphones.

Alphabets supportés par le téléphone :

```
AT+CSCS=?  
+CSCS: ("GSM", "UCS2")
```

Commande pour sélectionner l'alphabet « GSM » :

```
AT+CSCS="GSM"  
OK
```

Commandes générales

Nous allons tester dans cette partie uniquement les commandes de la norme GSM07.05.

Pour être tranquille dans nos expérimentations, nous allons de suite entrer le code PIN pour déverrouiller la carte SIM, grâce à la commande AT+CPIN :

```
AT+CPIN="xxxx"  
OK
```

Les caractères xxxx doivent bien entendu être remplacés par le code PIN de votre téléphone. Si le code saisi est valide le téléphone doit répondre par OK.

Caractéristiques du module GSM

En principe lorsque le TE commence à établir une communication avec le TA et le ME, celui-ci demande leurs caractéristiques grâce aux commandes +GMI, +GMM, +GMR et +GSN, c'est ce que nous allons faire :

Retourne le nom du fabricant :

```
AT+CGMI  
SAGEM  
OK
```

■ Retourne le modèle :

```
AT+CGMM  
MY X-5 GPRS  
OK
```

■ Retourne la version :

```
AT+CGMR  
1.00  
OK
```

■ Retourne le numéro de série :

```
AT+CGSN  
987612345-123  
OK
```

■ Tout le monde connaît le fameux code *#06# qui une fois composé sur le clavier du téléphone permet d'afficher son identifiant international (IMEI). On obtient le même résultat avec la commande AT+CGSN :

```
AT+CGSN  
351030358226964  
OK
```

■ Pour information déchiffrons ce code qui sert de base pour le calcul du code de déverrouillage...

Les deux premiers chiffres (33) indiquent le pays d'origine du mobile, selon le code international de la numérotation téléphonique, dans cet exemple il s'agit de la France.

Les quatre chiffres suivants (1030) représentent le TAC (*Type Approval Code*) qui identifie le modèle de poste au regard de la procédure d'agrément.

Les deux chiffres suivants (35) sont le FAC (*Final Assembly Code*) qui précise le lieu d'assemblage final de l'appareil, donc en Bretagne dans cet exemple.

Les six chiffres suivants (822696) sont le numéro de série du mobile (SNR).

Le dernier chiffre est une clef dont le codage est similaire au dernier chiffre des numéros de cartes bancaires.

Indicateurs et contrôles

État d'activité :

Cette commande peut être utilisée pour interroger le ME avant de faire effectuer une action au téléphone :

```
AT+CPAS  
+CPAS: 0  
OK
```

Le zéro indique que le ME est prêt à recevoir des commandes de la part du TE.

Charge de la batterie indique l'état de connexion de la batterie et son niveau de charge :

```
AT+CBC  
+CBC: 0,80  
OK
```

Le ME est alimenté par la batterie et la charge est de 80 %.

Qualité du signal :

```
AT+CSQ  
+CSQ: 7,99  
OK
```

Le premier chiffre correspond au champ <rssi> qui indique la puissance du signal reçu. On sait que pour $rssi = 2$ la puissance correspondante est de -109 dBm, pour $rssi = 30$ on a une puissance de -53 dBm. De ces 4 valeurs on définit l'équation suivante : $P = 2rssi - 113$. Donc en injectant dans l'équation $rssi = 7$ on obtient une puissance égale à -99 dBm pour cet exemple. Rappelons que le dBm est une unité de mesure exprimant un niveau référencé par rapport à une puissance de 1 mW. En réalité la valeur obtenue par l'équation précédente est le gain (G). Si l'on considère que la station d'émission GSM envoie un signal d'une puissance égale à 1 mW et que le téléphone mobile reçoit une puissance P_s , le gain est donné par l'équation suivante : $G = 10 \log(P_s/1 \times 10^{-3})$. On obtiendra toujours un gain négatif

car bien évidemment la puissance reçue par le téléphone est toujours plus faible que le signal émis par la station, au mieux on peut avoir puissance reçue = puissance émise auquel cas le gain est égal à zéro. Pour calculer la puissance reçue en mW, on utilise l'équation : $Ps = 10^{G/10}$. Par exemple avec $G = -99$ dBm on obtient $Ps = 125 \times 10^{-12}$ mW.

La deuxième valeur 99 correspondant au champ <ber> indique que le taux d'erreur de bit est inconnu ou non détectable.

La commande +CIND regroupe les 3 commandes vues précédemment.

Demande la liste des indicateurs supportés par le ME et valeurs possibles :

```
AT+CIND=?  
+CIND:  
("battchg", (0..5)), ("signal", (0..5)), ("service", (0,1)), ("call", (0,  
1))  
OK
```

■ Commande de lecture :

```
AT+CIND?  
+CIND: 4,3,1,0  
OK
```

La lecture nous indique ici que la batterie est chargée à 80 %, que la qualité du signal est de 60 %, que le téléphone est en service et qu'il n'y a pas d'appel en cours.

Lecture de la date et de l'heure du ME :

```
AT+CCLK?  
+CCLK: "03/04/15,10:43:49"
```

Attention, le jour et l'année sont permutés, il faut lire : 15/04/03.

Programmation d'une alarme. L'alarme peut produire différents effets, écrire un message à l'écran du ME, émettre un son... Les effets sont spécifiques au fabricant du téléphone, pour connaître ceux disponibles sur votre téléphone tapez la commande suivante :

```
AT+CALA=?  
+CALA: (1),(sound)
```

Le chiffre « 1 » indique le nombre d'alarme qu'il est possible de programmer. Le deuxième paramètre « sound » signale l'effet de l'alarme, en l'occurrence d'émettre un son.

Exemple de programmation d'une alarme. Le ME émettra un son le 15 mars 2003 à 11:50.

AT+CALA="03/03/15,11:50:00"

OK

Gestion des répertoires téléphonique

La saisie des numéros mais surtout des noms dans un répertoire téléphonique n'est pas des plus aisées d'autant que la taille des touches du clavier ne cesse de diminuer à chaque nouveau modèle. Désormais plus de soucis grâce aux commandes +CPBR et +CPBW qui permettent de lire et de rajouter des entrées dans votre répertoire.

Un téléphone peut contenir, au maximum, 15 répertoires. La commande +CPBS permet de savoir quels sont ceux disponibles sur votre téléphone :

AT+CPBS=?

+CPBS: ("DC", "ME", "SM")

Ensuite il faut sélectionner un répertoire parmi ceux proposés, prenons le cas le plus courant, celui placé sur la carte SIM. Souvent le répertoire utilisateur est mémorisé sur la carte SIM, ce qui évite de ressaisir son contenu lorsque l'on change de téléphone.

AT+CPBS="SM"

OK

Chaque répertoire possède un espace mémoire fixe. La commande +CPBS utilisée en mode lecture permet d'obtenir cette information :

AT+CPBS?

+CPBS: "SM",2,50

Dans cet exemple, la mémoire possède une capacité de 50 enregistrements, dont 2 sont utilisés.

Chaque enregistrement est accessible via son index, le premier enregistrement ne possède pas forcément un index égal à 1. De plus les champs numéro et nom sont limités en terme de nombre de caractères. Consultons ces paramètres avec la commande +CPBR :

AT+CPBR=?

+CPBR: (1-150),10,12

La carte SIM dispose de 150 enregistrements encadrés par les index 1 à 150, il n'y a donc pas d'offset dans cet exemple. Le champ numéro peut contenir 10 caractères et le champ nom 12 caractères.

■ Il est possible de lire tous les enregistrements en une seule fois :

AT+CPBR=1,150

■ Seuls les enregistrements non vides sont affichés à l'écran. Une ligne correspondant à un enregistrement, voici le résultat obtenu :

```
+CPBR: 1,"0600000001",145,"Julien"  
+CPBR: 2,"0600000002",129,"stéphanie"
```

On retrouve le champ index puis le champ numéro, le chiffre 129_{dec} (soit 81_{hex}) indique qu'il s'agit d'un numéro national, le chiffre 145_{dec} (soit 91_{hex}) indique un numéro international, pour finir on trouve le champ nom.

Il existe même une fonction qui permet de rechercher dans le répertoire en cours le ou les enregistrements qui ont un champ texte qui commence par la chaîne de caractères spécifiée :

AT+CPBF="st"

■ La commande retourne l'enregistrement correspondant :

```
+CPBR: 2,"0600000002",129,"stéphanie"
```

Terminons par la commande +CPBW qui permet d'ajouter un enregistrement au répertoire. Si le champ index n'est pas spécifié, le nouvel enregistrement sera positionné au premier emplacement de libre trouvé. Si l'on utilise un index contenant déjà un enregistrement, celui-ci sera écrasé.

AT+CPBW=,"0600000003",129,"jacques"

■ On aurait obtenu le même effet avec la commande :

AT+CPBW=3,"0600000003",129,"jacques"

■ Si la commande est utilisée seulement avec le paramètre index, l'enregistrement correspondant est effacé. Par exemple pour effacer l'enregistrement numéro 3 :

AT+CPBW=3

Commandes SMS

Sélection de la zone mémoire pour lecture/écriture des SMS

Avant d'utiliser les commandes relatives aux SMS, il faut savoir quels sont les types de mémoires disponibles sur votre téléphone :

AT+CPMS=?

```
+CPMS: ("ME","SM"),("ME","SM"),("ME","SM")
```

D'après cet exemple les mémoires ME et SIM peuvent être utilisées aussi bien pour la lecture (<mem1>) que pour l'écriture (<mem2> et <mem3>).

Regardons la configuration actuelle :

AT+CPMS?

+CPMS: "ME",7,100,"ME",7,100,"ME",7,100

Avec cette configuration toutes les commandes de lecture et d'écriture se font sur la mémoire « ME » propre au téléphone qui dispose ici de 100 emplacements. La mémoire « ME » contient actuellement 7 messages.

Pour travailler sur la mémoire de la carte SIM il suffit d'envoyer la commande suivante :

AT+CPMS="SM","SM","SM"

+CPMS: "SM",1,12,"SM",1,12,"SM",1,12

Désormais, l'écriture et la lecture des SMS se fera exclusivement dans la mémoire de la carte SIM qui dispose de 11 emplacements libres.

Certains téléphones supportent l'option « MT », ainsi les commandes de lecture et d'écriture des SMS peuvent être utilisées sur toutes les mémoires disponibles sur le téléphone.

Voici à titre d'exemple ce que renvoie un MYX-5 de SAGEM :

AT+CPMS=?

+CPMS: ("ME","SM","MT")

On remarque que seule la mémoire <mem1> utilisée par les commandes de lecture est disponible. Il est possible d'affecter à <mem1> la mémoire de la carte SIM :

AT+CPMS="SM"

+CPMS : "SM",1,12

Affectons maintenant à <mem1> la mémoire du téléphone :

AT+CPMS="ME"

+CPMS : "ME",7,100

Dernière possibilité, affectons à <mem1> les deux mémoires :

AT+CPMS="MT"

+CPMS : "MT",8,112

On remarque que l'espace mémoire disponible est bien la somme de l'espace mémoire ME et du SM ($100 + 12 = 112$). Il en va de même pour le nombre de messages contenus ($1 + 7 = 8$).

Envoi d'un SMS

En Mode PDU

Première chose il faut s'assurer que le mode PDU est supporté par le module GSM, c'est normalement le cas quel que soit le modèle utilisé :

AT+CMGF=?

Le module retourne la liste des modes qu'il supporte :

+CMGF: (0,1)

Dans le cas présent le mode PDU est supporté (0) de même que le mode TEXT (1).

Activons donc le mode PDU :

AT+CMGF=0

Il faut entrer la longueur, en octets, de la trame qui compose le SMS :

AT+CMGS=17

Un curseur vous invite à saisir la trame. L'action des touches [CONTROL] et [Z] valide la saisie et envoie le SMS directement sur le réseau, il ne sera pas stocké sur le téléphone :

>0011000A8160102030400000AA04D4E2940A

Si tout s'est bien déroulé, le module GSM doit retourner la réponse suivante :

+CMGS: 0

OK

Le chiffre « 0 » correspond au champ MR qui est la référence du message comprise entre 0 et 255, cette référence est générée automatiquement par le mobile et signale également que le message est correctement envoyé. Par exemple, si on envoie un nouveau message au même destinataire, la référence sera incrémentée d'une unité.

Pour constituer la trame il est bien entendu fortement recommandé d'utiliser le logiciel « ConvertSMS.exe » détaillé dans le chapitre 1, sélectionnez la trame constituée par le logiciel puis faites un copier/coller pour l'importer dans Hyper Terminal.

En Mode TEXT

Première chose, il faut s'assurer que le mode TEXT est supporté par le module GSM :

AT+CMGF=?

■ Le module retourne la liste des modes qu'il supporte :

+CMGF: (0,1)

■ Dans le cas présent le mode PDU est supporté (0) de même que le mode TEXT (1).

Activons donc le mode TEXT :

AT+CMGF=1

OK

■ Comme nous l'avons vu pour envoyer un SMS il faut indiquer le numéro du SMSC que l'on souhaite utiliser. Ce paramètre est normalement déjà dans la mémoire du mobile, il correspond à celui de l'opérateur auquel vous avez souscrit votre abonnement. Pour s'en assurer demandons les paramètres associés à la commande +CSCA :

AT+CSCA?

+CSCA: "+33689004000"

■ Vous pouvez bien entendu modifier ce paramètre :

AT+CSCA="+61418706700"

OK

■ Il faut entrer le numéro de téléphone du destinataire du message :

AT+CMGS="0601020304"

■ Entrez le texte du message par exemple « TEST » et validez la saisie par l'action simultanée des touches [CTRL] et [Z] qui provoque l'envoi du SMS sur le réseau GSM.

> TEST

■ Si tout s'est bien déroulé, le module GSM doit retourner la réponse suivante :

+CMGS: 0

OK

■ Le chiffre « 0 » correspond au champ MR qui est la référence du message comprise entre 0 et 255_{dec}, cette référence est générée automatiquement par le mobile et signale également que le message est correctement envoyé. Par exemple si on envoie un nouveau message au même destinataire, la référence sera incrémentée d'une unité.

Autre méthode d'envoi d'un SMS

Dans le cas précédent le message constitué n'est pas stocké en mémoire mais directement expédié sur le réseau. Il est également possible de le sauvegarder temporairement en mémoire <mem2> grâce à la commande +CMGW afin de l'expédier au moment opportun.

Par exemple si l'on souhaite que <mem2>="ME", il faut utiliser la commande suivante :

```
AT+CPMS="ME", "ME"
```

Le premier paramètre qui correspond à <mem1> est obligatoire. Dans cette configuration la lecture des messages reçus et stockés non envoyés est sauvegardée dans la même mémoire.

Mode PDU (AT+CMGF=0)

```
AT+CMGW=17  
0011000A8160102030400000AA04D4E2940A
```

Mode TEXT (AT+CMGF=1)

```
AT+CMGW="0601020304"  
TEST
```

Que ce soit en mode PDU ou TEXT, le module GSM retourne l'emplacement mémoire <index> où est stocké le message :

```
+CMGW: 900
```

La commande +CMSS permet d'envoyer un message stocké en mémoire <mem2> par exemple à l'emplacement 900, ce qui provoquera l'envoi du message saisi précédemment :

```
AT+CMSS=900
```

Si l'envoi est réussi :

```
+CMSS: 0
```

Le chiffre « 0 » correspond au champ MR qui est la référence du message comprise entre 0 et 255_{dec}, cette référence est générée automatiquement par le mobile et signale également que le message est correctement envoyé. Par exemple, si on envoie un nouveau message au même destinataire, la référence sera incrémentée d'une unité.

Réception/lecture/suppression d'un SMS

En Mode PDU (AT+CMGF=0)

■ La manière dont le ME indique au TE qu'il vient de recevoir un nouveau SMS dépend du paramétrage de la commande +CNMI :

AT+CNMI=1,1

■ Avec cette configuration le ME signalera au TE la réception d'un nouveau SMS en envoyant le code :

+CMTI: <mem>,<index>

■ Par exemple voici le code affiché sur l'écran du PC signalant qu'un nouveau message est reçu et qu'il est sauvegardé à l'emplacement 1 de la mémoire de la carte SIM

+CMTI: "SM",1

■ Pour lire le message en question il faut utiliser la commande +CMGR suivie du paramètre index, on considère que la mémoire utilisée pour la lecture est celle de la carte SIM (<mem1>="SM", voir commande AT+CPMS) :

AT+CMGR=1

■ Le module GSM doit retourner le contenu du message sous forme d'une trame PDU :

+CMGR: 1,23

07913306091093F0040B913316502193F100003050616124430004D4E2940A

OK

Où 1 est le code état qui indique que le message a déjà été lu, 23_{hex} indique la longueur du message.

En Mode TEXT (AT+CMGF=1)

Les instructions utilisées pour configurer la réception et effectuer la lecture d'un SMS sont identiques à celles du mode PDU. La différence réside dans l'affichage du contenu du message. Le module GSM retourne l'état du message, le numéro de l'émetteur et la date à laquelle le message a été traité par le SMSC :

+CMGR: "REC READ", "+33610512391", "03/05/16,16:42:34+00"

■ ainsi que le corps du message

TEST

OK

Liste tous les SMS en mémoire

■ La commande AT+CMGL (ou AT+CMGL=4) permet de lister tous les messages stockés dans la mémoire <mem1> (voir commande AT+CPMS) :

AT+CMGL

Il est possible d'utiliser le paramètre facultatif <stat> pour afficher certains types de messages, par exemple si <stat>=1, seuls les messages reçus non lus seront affichés. La commande de test AT+CMGL=? renvoie la liste des <stat>s supportés par le ME. L'affichage à l'écran du message dépend du mode sélectionné PDU ou TEXT.

En Mode PDU (AT+CMGF=0)

Les 3 paramètres qui suivent le texte +CMGL correspondent respectivement aux champs <index>, <stat> au format numérique et <length> qui est la taille de la trame PDU affichée sur la deuxième ligne.

```
+CMGL: 1,2,25  
00001FF0281603200A712EDF27C1E3E97416537284CA797DDF432  
+CMGL: 2,1,23  
07913306091093F0040B913316502193F100003050616124430004D4E2940A  
OK
```

En Mode TEXT (AT+CMGF=1)

Les paramètres qui suivent le texte +CMGL correspondent respectivement aux champs <index>, <stat> au format texte, <OA> qui est le numéro de l'expéditeur du message (si SMS-DELIVER) ou <DA> qui est le numéro du destinataire (si SMS-SUBMIT) et la date d'envoi du SMS (si SMS-DELIVER). Sur la deuxième ligne on trouve le corps du message.

```
+CMGL: 1,"STO UNSENT","06"  
message en attente  
+CMGL: 2,"REC READ","+33610512391","03/05/16,16:42:34+00"  
TEST  
OK
```

Effacer un message SMS

Mode PDU/TEXT

Il convient d'effacer périodiquement les messages reçus afin de ne pas saturer la mémoire <mem1>. La commande +CMGD efface le SMS situé, par exemple, à l'emplacement <index> = 1

```
AT+CMGD=1
```

■ Le module doit confirmer l'effacement par la réponse :

```
OK
```

■ Notez qu'il n'existe pas de commande AT qui permet d'effacer en une seule fois tous les messages stockés en mémoire.

Commandes spécifiques au TM2 de Teltonika

Par défaut la vitesse de transmission des données série est de 115 200 bauds. Comme les montages présentés dans le chapitre suivant intitulé « Réalisations électroniques » utilisent tous une vitesse de 9 600 bauds, il est nécessaire de modifier le paramétrage d'usine du TM2.

La commande AT+IPR permet de modifier cette vitesse de transmission :

AT+IPR=9600

Dès lors il faut modifier le paramétrage d'Hyper Terminal. Cliquez sur Déconnexion  puis dans le menu Fichier sélectionnez Propriétés enfin cliquez sur le bouton Configurer et sélectionnez une vitesse de 9 600 bds. Cliquer sur Connexion .

La vitesse de transmission s'effectue bien à 9 600 bds mais au prochain RESET elle reviendra par défaut à 115 200 bds. Il faut donc mémoriser cette configuration dans un profil utilisateur.

Tout d'abord il faut paramétrter le TM2 pour qu'il charge le profil utilisateur n° 0 à chaque RESET :

AT&YO

OK

Il faut maintenant mémoriser la configuration en cours dans le profil utilisateur n° 0 :

AT&WO

OK

Une dernière commande est obligatoire pour que la mémorisation soit effective :

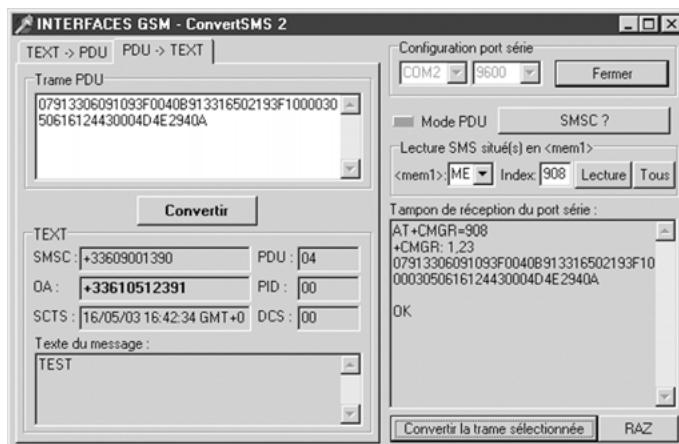
AT+CPWROFF

À l'issue de cette dernière commande le TM2 est automatiquement placé en mode veille.

Désormais à chaque RESET ou mise sous tension le profil utilisateur n° 0 sera automatiquement chargé. De fait le module TM2 communique à une vitesse de 9 600 bds compatible avec l'ensemble des réalisations électroniques présentées dans le chapitre suivant.

Logiciel intégré pour la gestion des SMS

Pour la gestion des SMS en mode PDU nous vous recommandons le logiciel ConvertSMS2.exe (**figure 4.5**) qui, comme son

Figure 4.5.
ConvertSMS2.

nom le laisse supposer, est une évolution de celui présenté dans le chapitre « Codage SMS ». Désormais il intègre une partie qui gère la liaison série. Le paramétrage par défaut est COM2, avec un débit de 9 600 bauds, 8 bits de données, pas de parité. Le port est actif une fois que vous avez cliqué sur le bouton Ouvrir. Au même instant une deuxième fenêtre plus petite s'affiche, vous demandant de saisir votre code PIN.

Après avoir constitué la trame PDU dans l'onglet « TEXT->PDU », il suffit de cliquer sur le bouton « envoi » pour que le message soit directement expédié sur le réseau via le ME, notez que le bouton « SMSC ? » indique le centre de messagerie utilisé.

Il est aussi possible de lire un SMS spécifique, dans la mémoire spécifiée par <mem1>, identifié par son index en cliquant sur « Lecture », ou lire tous les messages situés en mémoire grâce au bouton « Tous », le résultat s'affiche dans la zone de texte nommée « Tampon de réception du port série », sélectionnez alors la trame que vous souhaitez décoder puis cliquez sur le bouton « Convertir trame sélectionnée », le SMS s'affiche alors en mode text sur la fenêtre de gauche dans l'onglet « PDU->TEXT ».

Logiciel « WinGSM »

Si l'on fait abstraction des logiciels qui permettent de « déverrouiller » son portable, bien loin des préoccupations de cet ouvrage, il existe très peu de softs dédiés à la gestion des SMS. L'auteur a décidé de mettre fin à cette pénurie en développant son propre logiciel baptisé WinGSM qui permet de gérer et d'archiver sur le disque dur de son PC les SMS mémorisés sur son téléphone portable, et ce n'est pas tout :

Muni d'un cordon ad ok permettant de relier votre téléphone portable au port série de votre PC et du logiciel WinGSM vous avez la possibilité de :

- **Gérer/Archiver vos SMS**

Lorsque la mémoire de votre téléphone (ME) ou de votre carte SIM (SM) est pleine, vous êtes obligé de supprimer définitivement certains SMS. Grâce à WinGSM vous pouvez les transférer sur le disque dur de votre ordinateur (fichier SMS.gsm) et les consulter ultérieurement à votre convenance, même lorsque le téléphone n'est plus connecté au PC. Vous pouvez également composer facilement un nouveau SMS et l'expédier directement sur le réseau GSM ou le mémoriser pour l'envoyer au moment opportun.

- **Gérer/Sauvegarder votre répertoire téléphonique**

En cas de perte ou de vol de votre téléphone, vous êtes contraint de ressaisir les noms et les numéros de téléphones présents dans la mémoire de votre téléphone et/ou de votre carte SIM. Si vous avez pris soin, grâce à WinGSM, de faire une copie de sauvegarde sur le disque dur de votre PC (fichier répertoire .gsm), vous pouvez transférer, d'un click, toutes les entités de votre répertoire téléphonique vers votre nouveau mobile GSM. Vous pouvez bien entendu ajouter/modifier/supprimer facilement des numéros/noms dans le répertoire déjà en mémoire sur votre téléphone.

- **Visualiser à l'écran les paramètres de fonctionnement de votre téléphone**

WinGSM affiche de nombreuses données de paramétrage, souvent inaccessibles à l'utilisateur conventionnel.

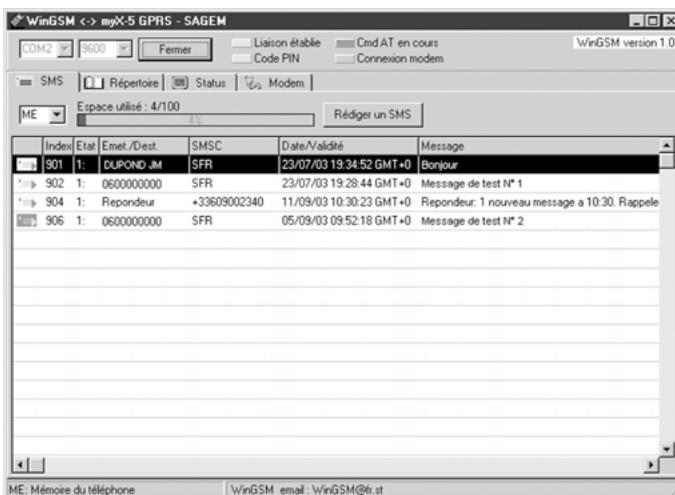
- **Utiliser votre téléphone comme un modem**

Avec WinGSM vous pouvez réaliser le transfert, à travers le réseau GSM, de données (DATA) à destination d'un modem fixe (RTC) ou à destination d'un autre téléphone GSM (**figure 4.6**).

WinGSM est compatible avec tous les téléphones GSM du marché et même avec les modules GSM intégrés MC35 et TC35 de Siemens ! (supporte les normes GSM07.05 et GSM05.05). Le programme fonctionne sous toutes les versions de Windows, 95 à XP.

4.2 AVEC UN PICBASIC

Le microcontrôleur choisi, pour jouer le rôle du TE, est un PicBasic du constructeur Coréen COMFILE TECHNOLOGY. Il existe 3 familles de PicBasic, celui que nous avons choisi, le PICBASIC-3B, appartient à la deuxième famille, il est un bon compromis

Figure 4.6.
WinGSM.

entre le coût et les possibilités offertes. Disponible en boîtier au format DIP 18 broches, il intègre un PIC 16C74A-04, un quartz de 4,19 MHz et une mémoire eeprom série d'une capacité de 4 Ko.

Que tous ceux qui sont allergiques au langage assembleur se rassurent, comme son nom le laisse présager, le PicBasic se programme en basic. Le logiciel PICBASIC-LAB fourni par le fabricant permet, à l'aide d'un PC, une conception vraiment très aisée du programme. Le puissant compilateur intégré permet de traduire les lignes basic en instructions spécifiques compréhensibles par le microcontrôleur. Le programme compilé peut ensuite être implanté dans la mémoire eeprom du PicBasic par le biais d'un cordon relié au port imprimante d'un PC.

Notez qu'en phase de conception la fonction debug vous permettra de suivre pas à pas l'exécution du programme par le PicBasic. Il est notamment possible de consulter l'état de toutes les variables utilisées par le programme. Une fois le programme au point il suffit de déconnecter le cordon pour rendre le PicBasic autonome.

Le langage Basic reconnu par le compilateur se compose une cinquantaine d'instructions. Celles qui nous intéressent fortement sont les instructions SERIN et SEROUT qui assurent la gestion d'un port série, elles nous permettront de communiquer facilement avec un module GSM.

L'instruction SEROUT

SEROUT Port, Param1, Mode, Param2, [Var1]

Cette instruction permet de transmettre des données sous forme série selon le protocole RS232. Une fois exécutée, la broche **Port**

du PICBASIC transmettra la ou les données **Var1** à une vitesse définie par **Param1**, selon la correspondance du tableau précédent. Le paramètre **Mode** permet d'instaurer une temporisation entre chaque caractère émis dont la durée en millisecondes est fonction de **Param2**. Les données envoyées doivent être de type « BYTE » c'est-à-dire comprise entre 0 et 255 (**tableau 4.2**).

Tableau 4.2.

Vitesse (bits/s)	PICBASIC 2H/3B/3H
2 400	
4 800	207
9 600	103
19 200	51

Envoi d'un SMS

Dans cet exemple nous allons montrer comment envoyer un SMS contenant le texte « TEST » au numéro « 0600000000 » à l'aide de la commande SEROUT. On considère que la sortie I/O17 (broche n° 22) du PICBASIC est utilisée et que la vitesse de transmission s'effectue à 9 600 bauds. Dans ce cas Port = 17, Param1 = 103, Mode = 0 ; la temporisation entre chaque caractère est fixée à 1 ms donc Param2 = 1. Comme nous l'avons déjà vu, la commande AT à utiliser est « AT+CMGS ».

Mode TEXT (AT+CMGF=1)

Premièrement il faut définir le numéro du destinataire :

```
SEROUT 17,103,0,1,[ "AT+CMGS=",34,"0600000000",34,13]
```

34_{dec} est le code ASCII du symbole guillemet. Les « vrais » guillemets encadrant le numéro sont là uniquement pour indiquer au compilateur qu'il doit traiter le numéro 0600000000 comme une chaîne de caractères, même chose pour la commande AT+CMGS. Alors que le code ASCII 34_{dec} ne sera pas interprété par le compilateur donc envoyé tel quel sur la sortie série. 13_{dec} est le code ASCII du retour chariot <CR>, qui déclenche l'exécution de la commande.

Ensuite il faut une temporisation d'au moins 1/2 seconde (500 ms) avant d'entrer le texte du message :

```
DELAY 500
```

■ Envoi du texte :

```
SEROUT 17,103,0,1,[ "TEST",26]
```

26_{dec} est le code ASCII du caractère EOF (*End Of File*) équivalent à la combinaison des touches CTRL+Z, qui provoque l'envoi du SMS sur le réseau GSM.

Mode PDU (AT+CMGF=0)

En mode PDU la première instruction indique la taille de la trame, 17 dans cet exemple :

```
SEROUT 17,103,0,1,["AT+CMGS=17",13]  
DELAY 500
```

■ Ensuite la trame elle-même est envoyée :

```
SEROUT 17,103,0,1,["0011000A8160000000000AA04D4E2940A",26]
```

26_{dec} est le code ASCII du caractère EOF (*End Of File*) équivalent à la combinaison des touches CTRL+Z, qui provoque l'envoi du SMS sur le réseau GSM.

L'instruction SERIN

```
SERIN Port, Param1, Mode, Param2, Adress, [Var1]
```

Cette instruction permet d'attendre la réception de données sous forme série selon le protocole RS232. Une fois exécutée, la broche **Port** du PICBASIC attendra la ou les données **Var1** à une vitesse définie par **Param1**, selon la correspondance du tableau vu précédemment. Durant cette phase le PICBASIC ne pourra pas effectuer d'autres tâches et attendra en permanence les données pendant une durée définie en millisecondes par **Param2**. Si la durée d'attente est dépassée, sans qu'aucune donnée ne soit reçue, le programme passera à l'adresse indiquée par **Adress**. Le paramètre **Mode** n'est pas utilisé et doit être positionné à 0.

Réception d'un SMS

La méthode présentée ici consiste à mettre en œuvre la commande AT+CNMI afin de configurer le ME (téléphone) pour qu'il signale au TE (PicBasic) l'arrivée d'un SMS. Rappelons que cette commande utilise les paramètres <mode> et <mt> voir norme GSM07.05. Lorsque <mode>=1 les indications concernant la réception d'un nouveau message sont directement transférées au TE. Les dites indications, si le message est du type SMS-DELIVER, sont de la forme +CMTI:<mem>,<index> à condition que <mt>=1, <mem> contient la référence de la mémoire utilisée pour stocker le message et <index> son emplacement au sein de cette mémoire.

En langage PicBasic nous devons utiliser dans un premier temps l'instruction SEROUT pour configurer le ME à l'aide de la com-

mande AT+CNMI suivie des paramètres <mode> et <mt> et du code ASCII 13_{dec} qui déclenche l'exécution de la commande :

```
SEROUT 17,103,0,1,[ "AT+CNMI=1,1",13 ]
```

Désormais le ME signalera systématiquement au TE l'arrivée d'un nouveau SMS en envoyant sur la ligne série RxD l'instruction +CMTI: <mem>,<index>. Le PicBasic doit donc en permanence scruter la ligne RxD dans l'espoir de recevoir cette instruction. L'instruction SERIN associée à WAIT place le µC dans une phase de scrutation de l'entrée série, correspondant ici à l'entrée I/O16 (broche 21), qui dure 1 000 ms (1 s), dès que la chaîne « TI » est reconnue, les caractères qui suivent sont placés dans la variable tableau Tampon qui peut contenir jusqu'à 8 valeurs de type BYTE. Si la chaîne attendue n'est pas reçue durant cette seconde le programme boucle sur l'étiquette ATT (pour ATTente). Le simple fait de placer l'étiquette ATT sur la même ligne que l'instruction SERIN permet de reconduire indéfiniment la phase d'attente.

```
ATT: serin 16,103,0,1000,ATT,[WAIT("TI"),Tampon(0)~8]
```

Imaginons maintenant que le ME vient de recevoir un SMS et qu'il a stocké dans la mémoire de la carte SIM à l'emplacement numéro 1. Aussitôt l'instruction +CMTI: "SM",1 est envoyée au PicBasic qui va placer les 8 caractères suivants « TI » dans la variable Tampon, voir **tableau 4.3**.

Tableau 4.3.

Tampon(0)	Tampon(1)	Tampon(2)	Tampon(3)	Tampon(4)	Tampon(5)	Tampon(6)	Tampon(7)
:		"	S	M	"	,	1
58 _{dec}	00 _{dec}	34 _{dec}	83 _{dec}	77 _{dec}	34 _{dec}	44 _{dec}	49 _{dec}

Il serait plus judicieux d'utiliser par exemple l'instruction WAIT("+CMTI: ") mais cette combinaison n'est malheureusement pas acceptée par le compilateur, l'instruction WAIT ne peut contenir que deux caractères.

Parmi les 8 octets contenus par la variable Tampon, les données Tampon(2) à Tampon(5) contiennent le nom de la mémoire <mem1> où est stocké le message et Tampon(7) contient la fameuse donnée <index>, ici égale à 1 qui correspond à l'emplacement du message dans la mémoire. On considère dans cet exemple que <index> est compris entre 0 et 9, en pratique ce n'est pas toujours le cas, mais il sera temps de voir cela dans le chapitre 5. Par contre le nom de la mémoire est toujours codé sur quatre caractères.

Premièrement nous allons paramétrier le téléphone pour que la commande de lecture des SMS se fasse sur la mémoire précisée par les variables Tampon(2) à Tampon(5), grâce à la commande AT+CPMS :

```
SEROUT 14,51,0,1,[ "AT+CPMS=", Tampon(2), Tampon(3), Tampon(4),  
Tampon(5),13]
```

Cette ligne de code est équivalente pour cet exemple à la commande :

```
AT+CPMS="SM"<CR>
```

Remarque : comme les données Tampon(2) et Tampon(5) contiennent toujours le code ASCII 34_{dec} qui correspond au guillemet, on aurait pu utiliser le code suivant :

```
SEROUT 14,51,0,1,[ "AT+CPMS=", 34, Tampon(3), Tampon(4), 34, 13]
```

La commande AT+CMGR suivie du paramètre <index> contenu dans la variable Tampon(7) permet d'effectuer la lecture du SMS en question, lecture qui va se faire, pour cet exemple, à l'emplacement numéro 1 dans la mémoire de la carte SIM grâce à la commande précédente.

```
SEROUT 14,51,0,1,[ "AT+CMGR=", Tampon(7),13]
```

Cette ligne de code est équivalente à la commande : AT+CMGR=1<CR>.

Temporisation de 500 ms (0,5 s) pour laisser le temps au ME d'exécuter la commande.

```
DELAY 500
```

Pour contenir le message nous imaginons qu'il existe une deuxième donnée baptisée SMS de type tableau limitée à 70 valeurs de type BYTE. La RAM du PicBasic 3B peut contenir 80 octets, 7 sont utilisés par la variable Tampon, on se garde donc une marge de 3 octets. L'instruction SERIN permet de capturer les 70 caractères envoyés par le ME sur la ligne RxD du PicBasic. Si le SMS comporte plus de 70 caractères il sera tronqué. À l'inverse, si le SMS contient moins de 70 caractères, le programme basculera sur l'étiquette SUITE au bout de 1 000 ms (1 s).

```
serin 16,103,0,1000,SUITE,[SMS(0)~70]
```

Le contenu de la variable SMS dépend du mode utilisé PDU ou TEXT.

Voici ce que pourrait contenir la variable SMS en mode PDU :

```
+CMGR: 0..24  
07911614786007F0040B911604994743F400009930139100406B05E8329BFD06  
OK
```

Le décodage de la trame PDU n'est pas évidente pour le PicBasic, c'est pour cela que *nous allons privilégier l'utilisation du mode TEXT dans le chapitre 5.*

En mode TEXT on aurait :

```
+CMGR: "REC READ", "+61407809050", "98/12/01,20:16:11+44"  
TEST  
OK
```

La première ligne contient l'en-tête du message, le premier paramètre « REC READ » indique qu'il s'agit d'un message non lu, le deuxième paramètre donne le numéro de l'expéditeur du message, le dernier paramètre indique la date à laquelle le message a été envoyé. Le texte du message qui nous intéresse est situé sur la deuxième ligne soit « TEST », on constate qu'il n'est pas évident à première vue d'extraire cette donnée. Il serait judicieux de placer dans la variable SMS uniquement le mot « TEST ». Malheureusement l'instruction WAIT ne peut pas être utilisée pour détecter le caractère <CR> afin de mémoriser uniquement la donnée située sur la deuxième ligne. La solution consiste à faire précéder le corps du message d'une paire de caractères spécifiques qui déclencherait la mémorisation dans la variable SMS. Il faut choisir des caractères qui ne devront jamais apparaître dans l'en-tête du message sous peine de récupérer des données inutiles. Deux points d'exclamations « !! » semble être une bonne solution. Voici donc l'instruction basic à utiliser :

```
ATT1: serin 16,103,0,1000,ATT1,[WAIT("!!"),SMS(0)~70]
```

En admettant cette fois que le SMS réceptionné est de la forme suivante :

```
+CMGR: "REC READ", "+61407809050", "98/12/01,20:16:11+44"  
!!TEST  
OK
```

La variable SMS est constituée comme le montre le **tableau 4.4**.

Les données SMS(0) à SMS(3) contiennent les données souhaitées. Il suffirait de faire tester ces 4 variables au PicBasic pour que celui-ci effectue une action en fonction de leur contenu, c'est ce que nous allons maintenant mettre en pratique dans le chapitre suivant.

Tableau 4.4.

SMS(0)	SMS(1)	SMS(2)	SMS(3)	SMS(4)	SMS(...)	SMS(...)	SMS(70)
T	E	S	T				
48 _{dec}	69 _{dec}	83 _{dec}	48 _{dec}				

5 RÉALISATIONS ÉLECTRONIQUES

5.1 Récepteur/émetteur SMS	102
5.2 Télécommandes par GSM	123
5.3 Téléméasures par GSM	163
5.4 Carte Entrées/Sorties pilotée par GSM	194
5.5 Géolocalisation par GSM	228

	Annexes	251
	Glossaire	261
	Bibliographie	264

Dans la réalisation d'une application sans fil, la portée est le paramètre le plus important. Avec les modules HF intégrés Aurel ou Mipot du commerce il est difficile d'assurer une transmission correcte sur une distance supérieure à 100 m et ceci même dans les conditions les plus favorables, temps clair, terrain dégagé... Même si la portée est liée à la puissance d'émission, il existe une limite fixée par la réglementation. Voilà pourquoi les modules vendus dans le commerce délivrent une puissance d'émission ne dépassant pas quelques dizaines de milliwatts.

Avec la technologie GSM, la portée n'est plus un problème. Un téléphone portable possède certes une puissance de quelques watts mais celle-ci est largement suffisante pour accéder au réseau téléphonique via les antennes relais quadrillant notre territoire. Il devient alors possible par l'envoi et la réception de commandes sous forme de SMS de piloter et de surveiller un processus quelconque se déroulant sur un site distant, voire même mobile (automobile). La mise en œuvre de ce procédé nécessite le développement d'une électronique autonome capable de s'interfacer avec un téléphone et de communiquer avec celui-ci via les commandes AT que nous avons décrites précédemment, et disposant d'actionneurs et de capteurs pour interagir avec son environnement. C'est le but que nous allons atteindre dans ce dernier chapitre.

Tous les montages présentés ci-après peuvent s'utiliser avec un téléphone portable GSM supportant le mode TEXT connecté à l'aide d'un cordon DATA ; ou avec n'importe quel terminal GSM supportant les standards GSM07.07 et GSM07.05, comme le TM2 de Teltonika pour ne citer que lui.

5.1 RÉCEPTEUR/ÉMETTEUR SMS

Les deux premiers montages ont surtout une vocation pédagogique. Il est certainement plus simple de lire directement le SMS reçu sur l'écran LCD de votre téléphone ; ou de composer sur le clavier le message à envoyer. Cependant ceci est moins évident sur les terminaux GSM (par exemple TM2) qui ne disposent d'aucune interface homme-machine...

Récepteur de SMS sur écran LCD

Dès que le téléphone portable ou le terminal GSM connecté à ce montage reçoit un nouveau SMS, un buzzer interpelle l'utilisateur pour qu'il consulte le contenu du message affiché sur l'écran LCD.

Schéma électrique

Voir **figure 5.1.**

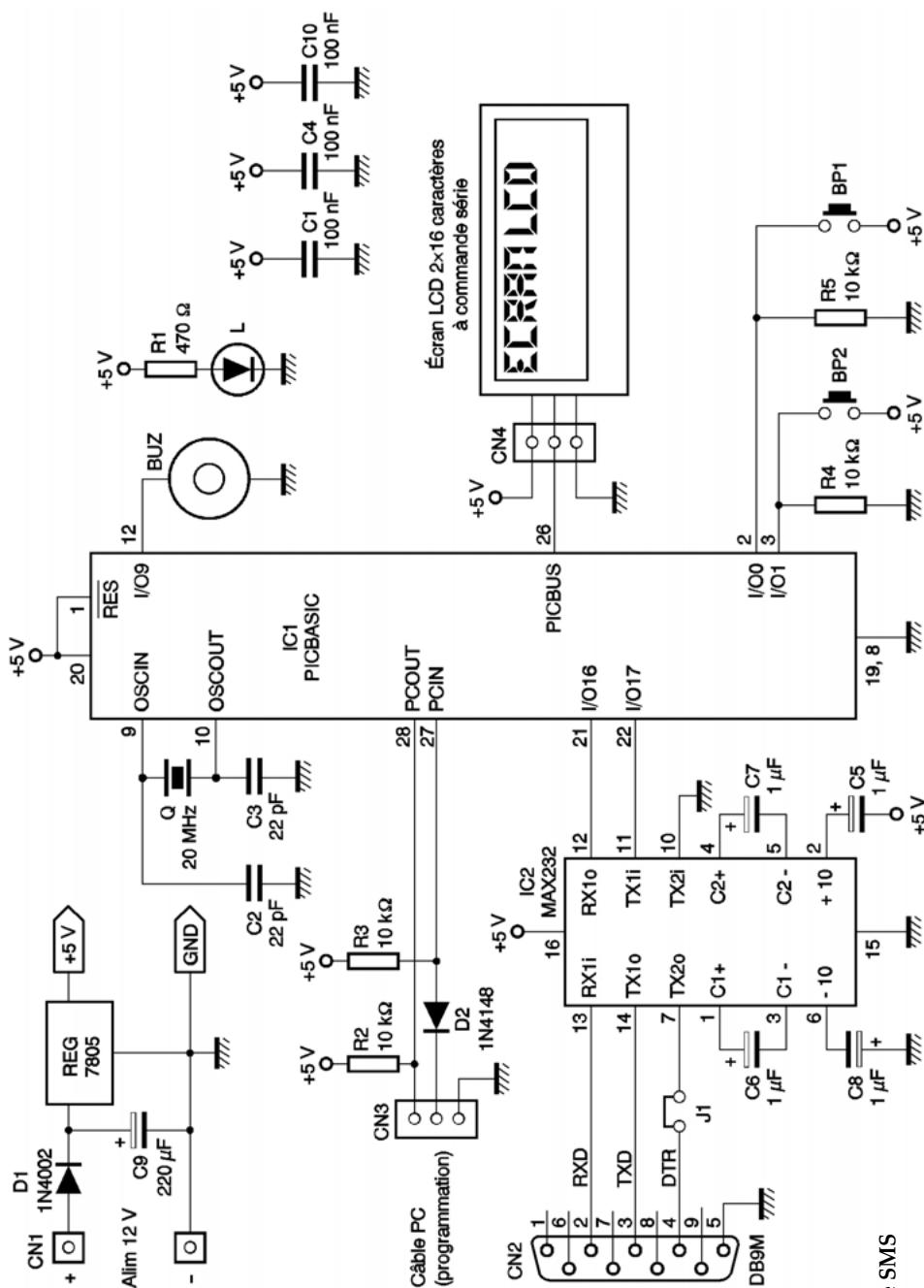


Figure 5.1.
 Schéma
 du récepteur de SMS
 sur écran LCD.

Le cœur du montage est comme convenu un PicBasic PB-3B. Les lignes I/O16 (broche 21) et I/O17 (broche 22) sont mises à contribution pour l'acquisition des données séries. Le circuit IC2, un Max232 d'adapte le niveau de tension des signaux en provenance et à destination du téléphone GSM relié au circuit par l'intermédiaire d'un câble adaptateur que nous avons décrit dans la première partie du chapitre 3. Si vous utilisez un terminal GSM tel que le TC35 de siemens qui dispose d'une sortie RS232 normalisée, la liaison se fera avec un câble RS232 standard. Le Max232 est câblé avec 4 condensateurs au tantale nécessaires à l'activation de sa pompe de charge interne qui permet de passer la tension d'alimentation de 5 à 12 V. L'entrée RXD (broche 2) qui récupère les données en provenance du téléphone est reliée à l'entrée RX1i, si sur cette entrée une tension de + 12 V est appliquée on aura sur la sortie correspondante RX1o une tension nulle qui sera appliquée sur la ligne I/O16 configurée pour l'occasion en entrée. Si l'entrée RX1i est soumise à une tension de - 12 V, on aura sur l'entrée I/O16 une tension de + 5 V. La ligne I/O17 (broche 22) est configurée en sortie pour l'envoi des données à destination du téléphone, elle est reliée à l'entrée TX1i. Une tension nulle appliquée sur TX1i donne une tension de + 12 V sur la sortie TX1o et sur TxD (broche 3). Une tension de + 5 V donne une tension de - 12 V sur TxD. La sortie DTR est aussi mise à contribution, mais uniquement pour l'alimentation du câble utilisé conjointement avec un téléphone portable. En effet les câbles DATA LINK du commerce utilisent cette sortie pour alimenter leur électronique interne. Comme l'entrée TX2i est reliée à la masse, on obtient sur TX2o donc sur DTR une tension de + 12 V. Cette sortie est inutile si vous utilisez un terminal GSM, il suffit dans ce cas de retirer le cavalier J1. Les lignes RXD, TxD et DTR sont disponibles sur un connecteur au format DB 9 broches mâle pour une connexion directe.

Comme tout microcontrôleur qui se respecte, le PicBasic utilise un quartz, de 20 MHz, associé aux condensateurs de découplage C2 et C3 de 22 pF, pour cadencer l'exécution du programme. L'entrée RES qui est en logique inversée, est reliée directement au + 5 V, ainsi à chaque mise sous tension du montage le PicBasic est remis à zéro. L'alimentation se fait par la broche 20, deux broches 8 et 19 sont utilisées pour la mise à la masse.

Le PicBasic dispose d'instructions spécialement dédiées pour le pilotage d'un afficheur LCD à commande série. Une seule ligne nommée PICBUS (broche 26) suffit à piloter l'afficheur 2 × 16 caractères que nous avons choisi. Un buzzer constitué d'un simple disque piezzo est connecté sur la sortie I/O9 (broche 12), là encore une instruction Basic délivre un signal carré d'une

fréquence de 4 kHz pour faire émettre un « bip ». Deux boutons poussoirs BP1 et BP2 respectivement reliés aux lignes I/O0 (broche 2) et I/O1 (broche 3) permettent à l'utilisateur d'agir sur le déroulement du programme. Lorsque le bouton poussoir est inactif, l'entrée correspondante est à la masse via une résistance de 10 kΩ. Les résistances R2 et R3 de 10 kΩ associées à la diode D1 permettent, par l'intermédiaire du cordon spécifique (fourni par Lextronic) connecté sur le port imprimante d'un PC, de transférer le programme dans l'eeprom du PicBasic. L'alimentation de l'ensemble est confiée à un traditionnel régulateur de tension 7805 en boîtier TO220 capable de débiter une tension régulée de + 5 V avec une intensité maximale de 1 A. Le condensateur électrolytique de 220 µF assure un filtrage efficace de la tension d'alimentation provenant, par exemple, d'un bloc secteur délivrant une tension maximale continue de 15 V. La diode D1 protégera le montage dans le cas d'une malencontreuse inversion de polarité. Pour terminer, une Led associée à une résistance de 470 Ω signale visuellement la présence de la tension d'alimentation.

Réalisation

L'impression du circuit sur du papier transparent avec une imprimante à jet d'encre ou laser permet d'obtenir un masque de bonne qualité. Pour avoir une opacité correcte des pistes, il est conseillé de superposer deux masques lors de l'insolation. Le temps d'exposition aux UV peut ainsi être augmenté pour une révélation sans surprise. Après gravure, rinçage et perçage, les différents composants seront soudés conformément au schéma d'implantation. Attention à l'orientation des composants polarisés. La borne positive du buzzer correspond au disque gris de plus faible dimension. La borne positive pour chacun des condensateurs est signalée par un signe « + » gravé côté cuivre. Soyez également vigilant sur l'orientation du connecteur du câble de programmation et celui de l'afficheur LCD (voir **figure 5.4**).

Programme du PicBasic : « recep.bas »

Le programme implanté dans la mémoire eeprom du PicBasic est largement aussi important que la partie électronique, puisque c'est de lui que dépend le fonctionnement correct du montage. Il n'est donc pas inutile de le détailler surtout pour ceux qui désireront par la suite ajouter des fonctionnalités au montage.

¹ DECLARATION DES CONSTANTES

'-----'

Pour faciliter la maintenance nous avons déclaré trois constantes : TXD qui correspond à la ligne I/O16 (broche n° 22) du PicBasic, RXD qui correspond à ligne I/O17 (broche n° 21) et BDS pour

INTERFACES GSM

Figure 5.2.
Circuit imprimé.

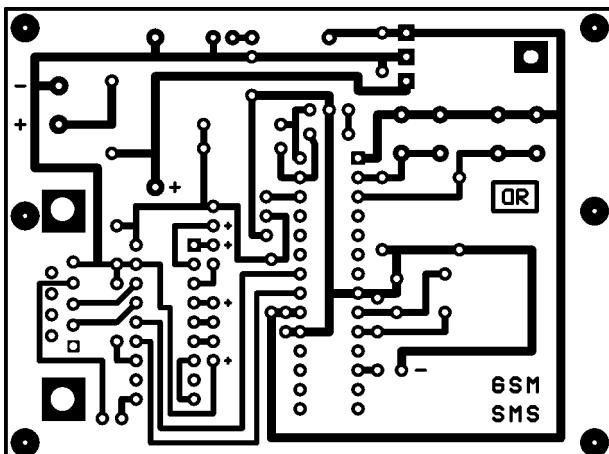
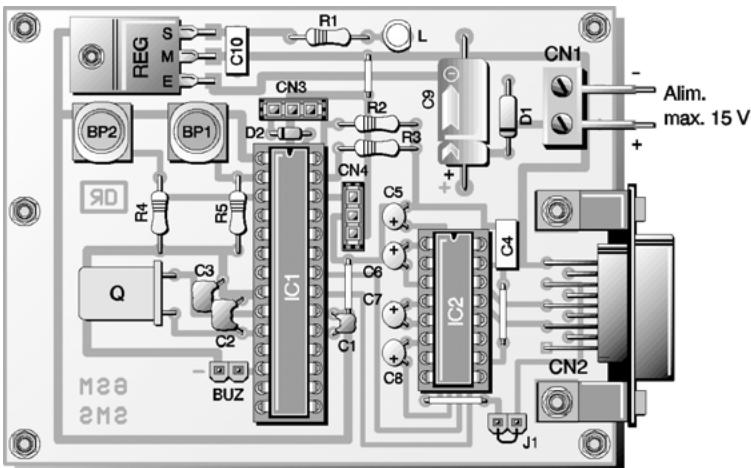


Figure 5.3.
Implantation
des composants.



Liste des composants

R1 : 470 Ω
 R2, R3, R4, R5 : 10 k Ω
 C1 : 100 nF (pas de 2,54 mm)
 C2, C3 : 22 pF / céramique
 C4, C10 : 100 nF / LCC jaune
 C5, C6, C7, C8 : 1 μ F / tantale / 15 V
 C9 : 220 μ F / électrolytique / 15 V
 D1 : diode 1N4002
 D2 : diode 1N4148
 L : Led standard
 Q : quartz 20 MHz
 REG : régulateur 7805

BUZ : buzzer piezzo (sans électronique intégrée)
 BP1, BP2 : bouton poussoir type D6
 J1 : barrette HE10 2 contacts + cavalier
 CN1 : bornier à vis 2 plots
 CN2 : connecteur DB9 mâle pour CI / coudé à 90°
 CN3 : connecteur pour câble de programmation (LEXTRONIC)
 CN4 : connecteur pour écran LCD (LEXTRONIC)
 IC1 : PICBASIC PB-3B (LEXTRONIC)
 + support DIL 28 broches (étroit)
 IC2 : MAX232 + support DIL 16 broches
 Écran LCD série 2 x 16 caractères (LEXTRONIC)

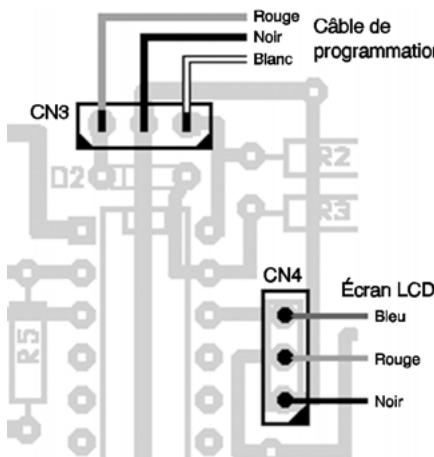


Figure 5.4.

bauds qui définit la vitesse de transmission ici fixée à 9 600 car BDS = 103. L'utilisation de constantes évite de parcourir tout le programme lorsque l'on désire modifier un des paramètres.

```
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16
```

'DECLARATION DES VARIABLES

Comme dans n'importe quel autre programme nous avons besoin de variables pour stocker dans la mémoire RAM des données et les récupérer ultérieurement. La première variable nommée Tampon contient la mémoire et l'index du SMS. Il n'est pas possible d'utiliser des variables de type texte (string), les seuls types supportés par le compilateur sont BYTE qui est un nombre codé sur 8 bits donc compris entre 0 et 255 et INTEGER qui est un nombre codé sur 16 bits donc compris entre 0 et 65 535. Par contre il est possible de configurer une variable en tableau afin de stocker plusieurs valeurs. Sachant qu'un caractère ASCII se code sur 8 bits, il est possible de stocker une chaîne de caractères dans un tableau de type BYTE. Dans ce cas la ligne de commande DIM Tampon(10) as BYTE signifie que la variable Tampon peut contenir jusqu'à 10 valeurs de type BYTE. La deuxième variable nommée SMS stocke le contenu du SMS, limité ici à 16 caractères. Enfin la variable simple i de type BYTE qui sert notamment pour les boucles FOR/NEXT.

```
DIM Tampon(10) AS BYTE
DIM SMS(16) AS BYTE
DIM i AS BYTE
```

'INITIALISATION DE L'ECRAN LCD

Il convient d'initialiser l'écran LCD connecté au PicBasic grâce aux instructions spécifiques à ce type d'afficheur. L'instruction SET PICBUS HIGH ou LOW permet de paramétrier la vitesse de communication du bus spécialisé « PICBUS ». Par défaut ce type d'afficheur est configuré pour travailler à une vitesse de 19 200 bauds, donc l'instruction SET PICBUS sera suivie de l'instruction HIGH (LOW pour une vitesse de 4 800 bauds). L'instruction LCDINIT initialise l'écran LCD.

```
SET PICBUS HIGH  
LCDINIT
```

'TEST LIAISON SERIE

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit : AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte. Les caractères « AT » suivis du caractère <CR>=13_{dec} sont envoyés par la commande SEROUT. L'instruction SERIN permet d'attendre l'éventuelle réponse « OK » pendant 2 000 ms (soit 2 s). Si les caractères OK sont réceptionnés dans le temps donné, le caractère suivant soit <CR> est placé dans la variable i. Dans le cas contraire le programme saute à la ligne repérée par l'étiquette TEST0, i est alors vide. Il suffit de tester le contenu de i pour savoir si la liaison est établie.

```
i=0  
TEST0: SEROUT TXD,BDS,0,1,[ "AT",13 ]  
SERIN RXD,BDS,0,2000,TEST1,[ WAIT( "OK" ),i ]  
TEST1: IF i<>0 THEN  
    LOCATE 0,0  
    PRINT "Liaison OK"  
    LOCATE 0,1  
    PRINT "Test mode..."  
ELSE  
    LOCATE 0,0  
    PRINT "PB liaison !"  
    DELAY 5000  
END IF
```

Tant que la liaison n'est pas établie le programme boucle sur l'étiquette TEST0. L'écran LCD affiche le message « PB liaison ! ». Une fois la liaison établie, le programme suit son cours normal.

```
IF i=0 THEN GOTO TEST0
```

```
'SELECTION DE L'ALPHABET GSM
```

```
'-----  
SEROUT TXD,BDS,0,1,["AT+CSCS=",34,"GSM",34,13]  
DELAY 500
```

```
'CODE PIN
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable. 13_{dec} est le code ASCII du retour chariot <CR>, qui déclenche l'exécution de la commande.

```
SEROUT TXD,BDS,0,1,["AT+CPIN=",34,"7208",34,13]  
DELAY 500
```

```
'INITIALISATION DES VARIABLES
```

Il convient comme dans tout programme d'initialiser les variables, en particulier Tampon et SMS.

```
DEBUT: FOR i=0 TO 9  
      Tampon(i)=0  
      NEXT i  
FOR i=0 TO 15  
      SMS(i)=0  
      NEXT i
```

```
'INITIALISATION ME
```

On considère que la liaison est établie, nous allons configurer le ME en mode TEXT avec la commande « AT+CMGF=1 ». On prendra pour habitude de faire suivre l'envoi d'une commande par une temporisation d'au moins 500 ms (0,5 s) ceci afin de laisser le temps au ME de réceptionner, de traiter et éventuellement de répondre à la commande.

```
CLS  
i=0  
SEROUT TXD,BDS,0,1,["AT+CMGF=1",13]  
SERIN RXD,BDS,0,2000,INIT,[WAIT("OK"),i]  
INIT: IF i<>0 then  
      LOCATE 0,0  
      PRINT "Mode TEXT"
```

```
LOCATE 0,1
PRINT "Attente SMS..."
ELSE
LOCATE 0,0
PRINT "Mode TEXT"
LOCATE 0,1
PRINT "non supporte :("
DELAY 5000
END IF
IF i=0 THEN GOTO INIT
DELAY 1000
```

Le ME doit signaler au TE l'arrivée d'un nouveau SMS, pour cela utilisons la commande « AT+CNMI=1,1 ». Ainsi l'arrivée d'un SMS sera signalée par l'envoi au ME de la commande +CMTI: "SMS",1. Ceci dans le cas où le message serait stocké dans la carte SIM à l'emplacement n° 1.

```
SEROUT TXD,BDS,0,1,[ "AT+CNMI=1,1",13]
DELAY 1000

'ATTENTE RECEPTION SMS
'-----
```

Le µC est placé dans une phase d'attente. Dès que les caractères « TI » sont reçus, les 10 caractères suivants sont stockés dans la variable Tampon, le buzzer retentit et le texte « Message reçu » s'affiche sur la première ligne de l'écran LCD, sur la deuxième ligne apparaît la mémoire contenant le SMS et son index. Notez que l'instruction WAIT ne peut pas être utilisée avec une chaîne de plus de 2 caractères. L'index est obtenu après la concaténation des variables Tampon(7), Tampon(8) et Tampon(9) à condition que leur contenu soit compris entre $48_{dec}=0_{ASCII}$ et $57_{dec}=9_{ASCII}$. Ceci élimine les caractères indésirables du style <CR> ou <LF> qui s'enregistrent dans la variable Tampon lorsque l'index est codé sur un ou deux chiffres.

```
ATTSM: SERIN RXD,BDS,0,10000,ATTSM,[WAIT("TI"),Tampon(0)~10]
CLS
LOCATE 0,0
PRINT "Message reçu !"
LOCATE 0,1
PRINT "Mem:",Tampon(3),Tampon(4)
PRINT ",Index:"
FOR i=7 TO 9
IF Tampon(i)>=48 AND Tampon(i)<=57 THEN PRINT Tampon(i)
NEXT i
```

Un « bip » est émis par le buzzer jusqu'à ce qu'un des boutons poussoirs (BP1 ou BP2) soit actionné.

```
ATTBP: BEEP 9
IF IN(0)=0 AND IN(1)=0 THEN GOTO ATTBP
```

Comme l'index s'incrémente à chaque nouveau message réceptionné, il est nécessaire d'extraire cette donnée pour savoir où aller lire le message en mémoire. Dans l'état actuel des choses admettons que la donnée Tampon contient ce qui est indiqué au tableau 5.1.

Tableau 5.1.

Tampon (0)	Tampon (1)	Tampon (2)	Tampon (3)	Tampon (4)	Tampon (5)	Tampon (6)	Tampon (7)	Tampon (8)	Tampon (9)
:	"	S	M	"	,	1			

On considère dans le programme que la valeur <index> sera codée au maximum sur 3 chiffres. Il est par exemple possible que le stockage des SMS se fasse à partir de l'index 900. Cela dépend du téléphone utilisé et du type de mémoire sélectionné. Dans l'exemple présenté ici, le SMS est stocké dans la carte SIM à l'emplacement n° 1.

'LECTURE DU SMS RECU

Le fait d'actionner BP1 ou BP2 provoque la lecture du SMS dans la mémoire définie par les variables Tampon(3) et Tampon(4) situé à l'emplacement pointé par la valeur obtenue après la concaténation des variables Tampon(7), Tampon(8) et Tampon(9) à condition que leur contenu soit compris entre $48_{dec} = 0_{ASCII}$ et $57_{dec} = 9_{ASCII}$.

```
SEROUT TXD,BDS,0,1,[ "AT+CPMS=",34,Tampon(3),Tampon(4),34,13]
DELAY 500
SEROUT TXD,BDS,0,1,[ "AT+CMGR=" ]
FOR i=7 TO 9
  IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT
    TXD,BDS,0,1,[Tampon(i)]
  NEXT i
  SEROUT TXD,BDS,0,1,[13]
```

Dans notre exemple, les variables Tampon(8) et Tampon(9) étant vides, elles seront ignorées par le programme. La commande envoyée au ME est équivalente à « AT+CMGR=1<CR> ».

En réponse à la commande précédente le ME va transmettre le contenu du SMS. Comme il n'est pas possible au PicBasic de mémoriser l'intégralité du message qui peut atteindre 180 caractères, sans compter l'en-tête, on considère que le message est composé au maximum d'une chaîne de 16 caractères. La mémori-

sation du message dans la variable SMS s'effectuera dès la détection d'une paire de points d'exclamation. *Les SMS envoyés au montage devront donc toujours débuter par les caractères « !! ».* Voici à titre d'exemple un SMS tel qu'il est transmis à notre montage par le ME :

```
+CMGR: "REC READ","+61405809051","03/12/01,20:16:11+44"  
!!Ceci est un test
```

Tous les caractères qui précèdent la paire de points d'exclamations seront ignorés par le PicBasic. La variable SMS contient donc le texte « Ceci est un test ».

```
SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)~16]  
SUITE: IF SMS(0)=0 THEN GOTO RAZ
```

```
'AFFICHAGE SUR L'ECRAN LCD
```

Il reste maintenant à afficher le contenu de la variable SMS sur l'écran LCD du montage. Dans le cas où le SMS contiendrait moins de 16 caractères, le µC va enregistrer des caractères inutiles et notamment les caractères <CR><LF>OK<CR><LF> qui signalent la fin du texte. Pour éviter de les afficher sur l'écran il suffit de sortir de la boucle FOR/NEXT dès que le caractère <CR>=13_{dec} est détecté.

```
CLS  
LOCATE 0,0  
FOR i=0 to 15  
IF SMS(i)=13 THEN  
    GOTO RAZ  
ELSE  
    PRINT SMS(i)  
END IF  
NEXT i
```

```
'EFFACE LE SMS EN MEMOIRE
```

L'appui sur BP1 efface le SMS de la mémoire du téléphone grâce à la commande « AT+CMGD » suivie de l'index du SMS, BP2 permet de retourner au début du programme sans supprimer le SMS, attention tout de même à ne pas saturer la mémoire de votre téléphone.

```
RAZ: LOCATE 0,1  
PRINT "BP1:RAZ BP2:SUIV"  
BP1: IF IN(0)=1 THEN  
    SEROUT TXD,BDS,0,1,[ "AT+CMGD=" ]  
    FOR i=7 TO 9
```

```

IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT
TXD,BDS,0,1,[Tampon(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
DELAY 1000
BEEP 9
CLS
LOCATE 0,0
PRINT "RAZ Message"
LOCATE 0,1
PRINT "BP2:SUIV"
END IF
IF IN(1)=1 THEN GOTO DEBUT
GOTO BP1

```

Toutes les instructions que nous venons de décrire sont rassemblées dans le fichier « recep.bas ». Il reste maintenant à transférer le programme dans l'eprom du PicBasic. Pour raccorder le PC et le PicBasic via le câble imprimante, il vous faudra impérativement couper l'alimentation du montage, puis connecter le câble avant d'allumer le PC et en dernier lieu mettre le montage sous tension. Lancez le logiciel PICBASIC-LAB, ouvrez alors le fichier « recep.bas », cliquez sur le bouton « RUN ». Le programme est compilé en instructions assembleurs qui sont ensuite implantées dans la mémoire du PicBasic. Pour rendre le montage autonome, coupez toujours l'alimentation du montage puis celle du PC. De même, ne déconnectez le cordon de liaison que si le PC et le montage sont tous deux hors tension.

Test du montage

Dans un premier temps il est prudent de tester le montage à l'aide d'un PC avant d'y relier un téléphone. Réalisez un câble RS232 « croisé » à l'aide d'un cordon comportant 3 conducteurs et de 2 connecteurs DB9 femelles à câbler comme le montre la figure 5.5.

Reliez le montage au port série du PC. Ouvrez une session du logiciel Hyper Terminal, vous pouvez reprendre le fichier « InterfacesGSM.lnk » présenté dans le chapitre 4. Rappelons que la

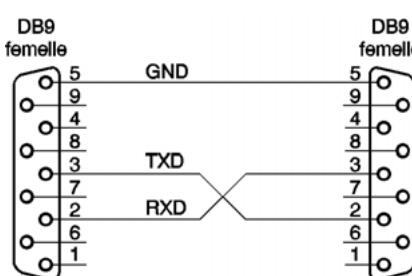


Figure 5.5.

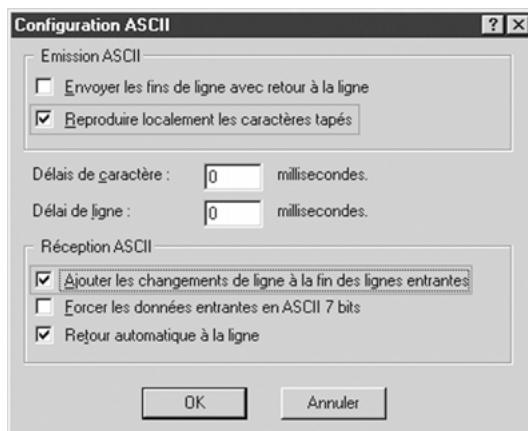


Figure 5.6.
Configuration ASCII.

vitesse de transmission est de 9 600 bauds, 8 bits de données et pas de contrôle de flux. Dans le menu « Fichier » sélectionnez « Propriétés », cliquez sur l'onglet « paramètres » puis finalement sur le bouton « Configuration ASCII... ». Sur la fenêtre qui apparaît (**figure 5.6**), cochez les cases « Reproduire localement les caractères tapés » et « Ajouter les changements de ligne à la fin des lignes entrantes », validez par « OK ».

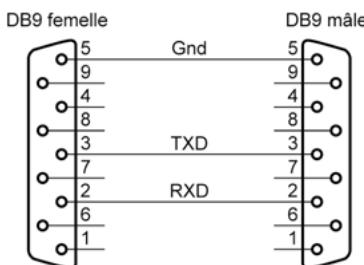
Alimentez le montage, aussitôt les caractères « AT » doivent apparaître sur l'écran du PC. Répondez dans les 2 secondes qui suivent en tapant les caractères « OK », suivis d'un retour chariot. L'écran LCD du montage doit afficher la phrase « Liaison OK ». La commande d'initialisation du mode TEXT AT+CMGF=1 doit s'afficher à l'écran, répondez aussi par OK et un retour chariot. La commande AT+CNMI=1,1 doit ensuite s'afficher. Simulons alors la réception d'un SMS en envoyant la commande +CMTI: "SM",1. Le buzzer doit retentir et l'écran afficher « Message reçu ! ». Actionnez un des deux boutons poussoirs de la platine pour demander la lecture du message. Vous devez voir apparaître la commande AT+CMGR=1, répondez en tapant par exemple « !!TEST ». Comme prévu, les caractères « TEST » qui suivent la chaîne « !! » doivent s'afficher sur l'écran LCD. Actionnez BP1 pour effacer le SMS, en retour vous obtenez à l'écran la commande correspondante AT+CMGD=1 et après une temporisation de 1 s le processus se répète.

Notez qu'il est difficile de saisir les réponses dans le temps donné. Il est possible de les préparer à l'avance dans des fichiers texte grâce au bloc-notes de Windows. L'envoi s'effectue par le menu « Transfert », « Envoyer le fichier texte... ». Une autre solution consiste à rallonger temporairement les temporisations des instructions SERIN.

Les 3 points détaillés ici sont valables pour l'ensemble des montages présentés dans ce chapitre.

Tableau 5.2.

- Pour connecter un téléphone portable à l'aide d'un câble DATA LINK du commerce n'oubliez pas de mettre en place le cavalier J1 sur le montage.
- Pour connecter un téléphone portable à l'aide de l'adaptateur TTL/RS232 ou TTL/FBUS ou M2BUS présenté dans le chapitre « Matériaux utilisés » vous devez utiliser un câble RS232 « droit » (voir figure ci-après). Une alimentation externe (la même que celle du montage) sera utilisée pour alimenter l'adaptateur (cavalier J1 de l'adaptateur en position A).
- Concernant le module TM2 vous pouvez directement le connecter au présent montage ou utiliser un câble RS232 « droit » (voir figure 5.7). L'alimentation du montage est fournie par le module TM2 (via le connecteur CN2).

Figure 5.7.
Câble
« droit ».

Le moment tellement attendu, celui de relier votre téléphone portable au montage, est enfin arrivé !!

Mettez la platine sous tension. Allez, par exemple, sur Internet pour envoyer gratuitement un SMS. N'oubliez pas de commencer le message par les caractères !!! Quelques secondes après l'envoi du SMS l'écran LCD du montage doit indiquer l'arrivée de votre message.

Résumé des points importants

Tableau 5.3.

RECEPTEUR DE MESSAGES SMS	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Eléments du programme PicBasic à modifier	
Code PIN (7208 par défaut)	
Commande SMS reçue	Action du montage
!!texte	Affiche le contenu du paramètre texte sur la première ligne de l'écran LCD (texte 16 caractères)

Émetteur de SMS

Nous présentons ici une deuxième utilisation du montage, la platine est identique, seul le programme est modifié. Le but est d'envoyer quotidiennement un SMS à une heure préalablement programmée. Nous en profiterons donc pour montrer comment bénéficier de l'horloge du téléphone qui fournit la date et l'heure courante. Pour varier les plaisirs le SMS est composé au format PDU.

Programme du PicBasic (version PDU) : « emet_pdu.bas »

```
'DECLARATION DES CONSTANTES
```

Déclarations de nos trois constantes employées dans les instructions SERIN et SEROUT pour configurer le port série.

```
CONST BDS = 103  
CONST RXD = 17  
CONST TXD = 16
```

```
'DECLARATION DES VARIABLES
```

Déclarations des quatre variables utilisées par le programme. La variable tableau CLK permet la mémorisation de la date et de l'heure fournies par le téléphone. La variable nbSMS de type BYTE permet de mémoriser le nombre de SMS envoyés. Nous verrons un peu plus loin l'utilité de la variable old. La variable i est notamment utilisée pour les boucles FOR/NEXT.

```
DIM CLK(20) AS BYTE  
DIM old AS BYTE  
DIM nbSMS AS BYTE  
DIM i AS BYTE
```

```
'INITIALISATION DE L'ECRAN LCD
```

La partie de code assurant l'initialisation de l'écran LCD, le test de la liaison série et l'envoi du code PIN est identique au programme précédent :

```
SET PICBUS HIGH  
LCDINIT
```

```
'TEST LIAISON SERIE
```

```
i=0  
TESTO: SEROUT TXD,BDS,0,1,["AT",13]
```

```

SERIN RXD,BDS,0,2000,TEST1,[WAIT("OK"),i]
TEST1: IF i<>0 THEN
        LOCATE 0,0
        PRINT "Liaison OK"
    ELSE
        LOCATE 0,0
        PRINT "PB liaison !"
        DELAY 5000
    END IF
    IF i=0 THEN GOTO TEST0
    DELAY 5000

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSMS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500

'INITIALISATION ME
'-----

```

■ Exceptionnellement l'envoi des SMS se fera en mode PDU :

```

CLS
i=0
SEROUT TXD,BDS,0,1,[ "AT+CMGF=0",13]
DELAY 1000

```

```
'ATTENTE / AFFICHAGE DATE et HEURE SUR ECRAN LCD
'
```

■ Voici maintenant le programme principal. Dans un premier temps on interroge le téléphone pour savoir qu'elle est la date et l'heure courante. Rappelons que la commande à utiliser est AT+CCLK?, suivie bien entendu d'un retour chariot :

```

old=0
ATT: SEROUT TXD,BDS,0,1,[ "AT+CCLK?"]
DELAY 500
SEROUT TXD,BDS,0,1,[13]

```

■ En retour le téléphone transmet le texte suivant : +CCLK:
"aa/mm/jj,hh:mm:ss", pour placer la date et l'heure dans la variable CLK nous allons attendre les caractères « LK » et mémo-riser les 20 caractères suivants :

```
SERIN RXD,BDS,0,2000,ATT,[WAIT("LK"),CLK(0)~20]
```

Examinons ce que contient désormais la variable tableau CLK (tableau 5.4).

Tableau 5.4.

CLK (0)	CLK (1)	CLK (2)	CLK (3)	CLK (4)	CLK (5)	CLK (6)	CLK (7)	CLK (8)	CLK (9)	CLK (10)	CLK (11)	CLK (12)	CLK (13)	CLK (14)	CLK (15)	CLK (16)	CLK (17)	CLK (18)	CLK (19)
:		"	A	A	/	M	M	/	J	J	,	H	H	:	M	M	:	S	S

AA : année, MM : mois, JJ : jour, HH : heure, MM : minute, SS : seconde.

Nous affichons sur la première ligne de l'écran LCD la date au format JJ/MM :

```
LOCATE 0,0
PRINT CLK(9),CLK(10),"/",CLK(6),CLK(7)," "
```

L'affichage de l'heure se fait à l'aide d'une boucle FOR et NEXT afin de parcourir les champs CLK(12) à CLK(19) :

```
FOR i=12 TO 19
  PRINT CLK(i)
NEXT i
```

Si l'heure courante correspond à l'heure programmée, un SMS doit être envoyé, le programme bascule alors sur le sous-programme EMET. Pour être certain que le montage n'envoie pas plus d'un SMS par jour, le programme teste le champ CLK(10) qui représente l'unité du jour ; tant qu'il n'a pas varié, aucun autre SMS ne peut être envoyé. La valeur précédente est mémorisée dans la variable old. Par défaut l'heure programmée est 09:55, vous pouvez bien entendu modifier ce paramètre selon votre convenance.

```
IF CLK(10)<>old THEN
  IF CLK(12)="0" AND CLK(13)="9" THEN
    IF CLK(15)="5" AND CLK(16)="5" THEN GOSUB EMET
  END IF
END IF

GOTO ATT
```

```
'EMISSION SMS
'-----
```

Voici maintenant la description du sous-programme qui assure l'envoi du SMS. À chaque envoi un compteur s'incrémentera afin d'afficher sur l'écran LCD le nombre total de SMS (limité à 99 pour éviter le débordement du texte de l'écran LCD) envoyés depuis la mise sous tension du montage.

```
EMET: nbSMS=nbSMS+1
IF nbSMS>99 THEN nbSMS=0
LOCATE 0,1
PRINT DEC(nbSMS,2,0)," SMS envoyé(s)"
```

En mode PDU, il suffit de préciser le nombre d'octets que comporte le message, puis d'envoyer la trame au téléphone, le caractère <eof> = 26_{dec} provoque l'envoi sur le réseau GSM :

```
SEROUT TXD,BDS,0,1,[ "AT+CMGS=46",13]
DELAY 1000
SEROUT
TXD,BDS,0,1,[ "0011000A8160572391950000AA25C3F2380D2ACFE9A0BA1BD42
ECFE7E17319442E83E8E5391D0497BFCFF270BB5D06"]
SEROUT TXD,BDS,0,1,[26]
old=CLK(10)
RETURN
```

La composition de votre propre message se fera avec le logiciel « ConvertSMS ». Renseignez le champ DA, c'est-à-dire le numéro de téléphone du destinataire, puis le champ texte du message, cliquez sur le bouton convertir. Faites ensuite un copier/coller de la trame obtenue vers le programme. N'oubliez pas de mettre à jour la longueur qui suit la commande « AT+CMGS ».

Résumé des points importants

Tableau 5.5.

ÉMETTEUR DE MESSAGES SMS (mode PDU)	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none"> Code PIN (7208 par défaut) Heure à laquelle le SMS doit être envoyé (09:55 par défaut) Trame PDU constituant le message à envoyer 	

Programme du PicBasic (version TEXT) : « emet_txt.bas »

Voici une deuxième version du programme, pour ceux qui préfèrent travailler avec le mode TEXT pour envoyer les SMS.

```
'DECLARATION DES CONSTANTES
'
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16
```

```
'DECLARATION DES VARIABLES
```

INTERFACES GSM

```
'-----
DIM CLK(20) AS BYTE
DIM old AS BYTE
DIM nbSMS AS BYTE
DIM i AS BYTE

'INITIALISATION DE L'ECRAN LCD
'-----
SET PICBUS HIGH
LCDINIT

'TEST LIAISON SERIE
'-----
i=0
TESTO: SEROUT TXD,BDS,0,1,[ "AT",13]
SERIN RXD,BDS,0,2000,TEST1,[WAIT("OK"),i]
TEST1: IF i>0 THEN
LOCATE 0,0
PRINT "Liaison OK"
ELSE
LOCATE 0,0
PRINT "PB liaison !"
DELAY 5000
END IF
IF i=0 THEN GOTO TESTO
DELAY 5000

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500

'INITIALISATION ME
'-----
```

L'envoi des SMS se fait désormais en mode TEXT grâce à la commande « AT+CMGF = 1<CR> ».

```
CLS
i=0
SEROUT TXD,BDS,0,1,[ "AT+CMGF=1",13]
DELAY 1000

'ATTENTE / AFFICHAGE DATE et HEURE SUR ECRAN LCD
'-----
old=0
ATT: SEROUT TXD,BDS,0,1,[ "AT+CCLK?" ]
```

```

DELAY 500
SEROUT TXD,BDS,0,1,[13]

SERIN RXD,BDS,0,2000,ATT,[WAIT("LK"),CLK(0)~20]

LOCATE 0,0
PRINT CLK(9),CLK(10),"/",CLK(6),CLK(7),"    "

FOR i=12 TO 19
  PRINT CLK(i)
NEXT i

IF CLK(10)<>old THEN
  IF CLK(12)="0" AND CLK(13)="9" THEN
    IF CLK(15)="5" AND CLK(16)="5" THEN GOSUB EMET
  END IF
END IF

GOTO ATT

'EMISSION SMS
'-----
EMET: nbSMS=nbSMS+1
  IF nbSMS>99 THEN nbSMS=0
  LOCATE 0,1
  PRINT DEC(nbSMS,2,0)," SMS envoyé(s)"

```

Avec le mode TEXT l'envoi du message « Ceci est un message de test programme » se fait à l'aide de la commande « AT+CMGS ». N'oubliez pas de modifier le numéro du destinataire du message.

```

SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34,"06xxxxxxxx",34,13]
DELAY 1000
SEROUT TXD,BDS,0,1,[ "Ceci est un message de test
programme",26]
old=CLK(10)
RETURN

```

Résumé des points importants

Tableau 5.6.

ÉMETTEUR DE MESSAGES SMS (mode TEXT)	
Configuration	Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none"> • Code PIN (7208 par défaut) • Heure à laquelle le SMS doit être envoyé (09:55 par défaut) • Numéro de téléphone pour l'envoi des SMS (06xxxxxxxx par défaut) • Texte du SMS (< Ceci est un message de test programme > par défaut) 	

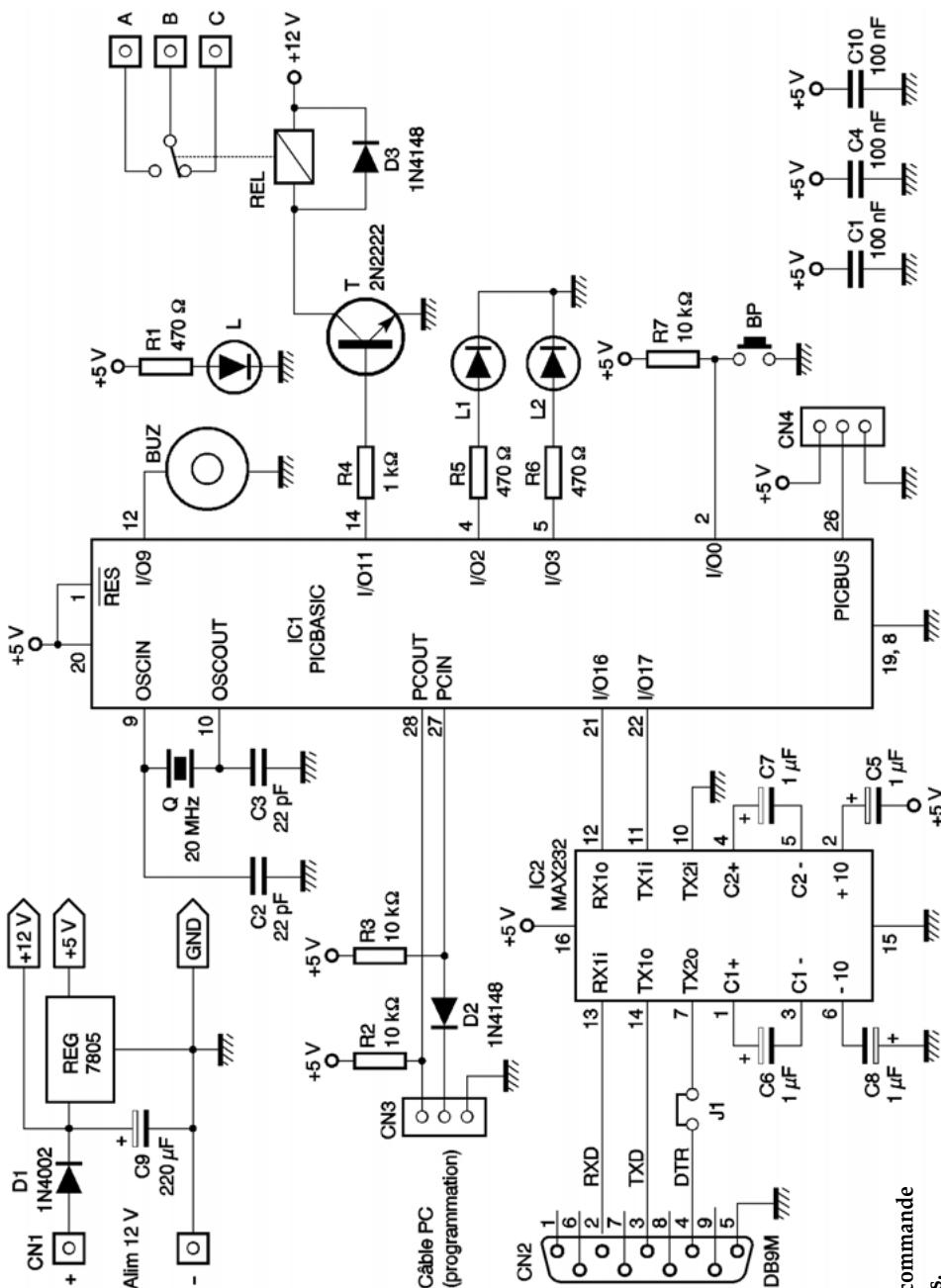


Figure 5.8.
Schéma de la télécommande
à 1 sortie sur relais.

5.2 TÉLÉCOMMANDES PAR GSM

1 sortie sur relais

Cette première télécommande qui possède une seule sortie, n'est, exceptionnellement, pas pilotée par SMS. Le changement d'état du relais se fait lorsque le téléphone reçoit un appel, à condition que le numéro de l'appelant soit autorisé à piloter la carte. L'utilisation de ce montage est totalement gratuite, aucun frais de communication n'est à prévoir du fait qu'il n'y a pas de prise de ligne, c'est la « sonnerie » qui est le vecteur de la commande. Attention, il faut tout de même que l'abonnement du téléphone connecté au montage inclue la présentation du numéro. Si ce n'est pas le cas vous pouvez toujours vous rabattre sur la version 2 du programme.

Schéma électrique

Nous ne reviendrons pas sur la description des parties communes au montage précédent. La sortie I/O11 ne disposant pas d'une puissance suffisante pour faire coller le relais, nous avons fait appel à un étage amplificateur constitué d'un simple transistor 2N2222. La base est reliée à la sortie I/O11 via une résistance de 1 kΩ. Le relais est placé entre le collecteur et l'alimentation du montage (en amont du régulateur) qui ne devra pas dépasser les 12 V. Notez la présence de la diode dite « de roue libre » D3 chargée de court-circuiter la force contre électromotrice générée par la bobine. Lorsque la sortie I/O11 est à l'état haut, T1 est saturé, le relais est actif. Lorsque I/O11 est à l'état bas, T1 est bloqué, le relais est inactif. Le seul capteur de la carte est le bouton poussoir BP1 connecté à l'entrée I/O0. La résistance de rappel R7 impose une tension de + 5 V lorsque le BP n'est pas actionné (**figures 5.9 et 5.10**).

Programme PICBASIC (version 1) : « 1sr_v1.bas »

```
' DECLARATION DES CONSTANTES
```

```
' -----
```

Déclaration des constantes utilisées par les instructions SERIN et SEROUT. TXD correspond à la ligne I/O16 du PicBasic, RXD correspond à ligne I/O17 et BDS pour bauds qui définit la vitesse de transmission ici fixée à 9 600 car BDS = 103.

```
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16
```

```
' DECLARATION DES VARIABLES
```

```
' -----
```

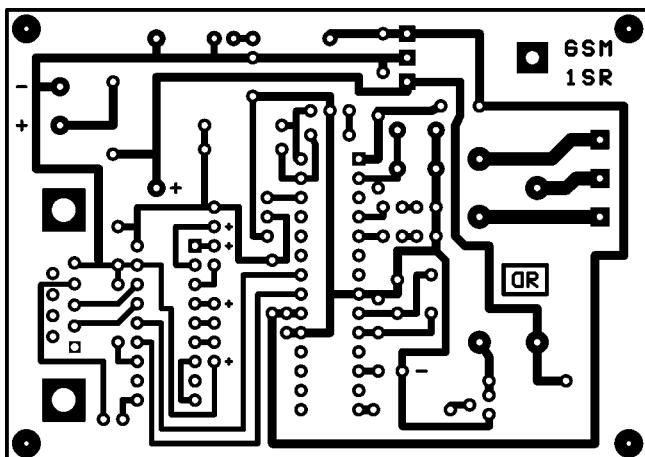


Figure 5.9.
Circuit imprimé.

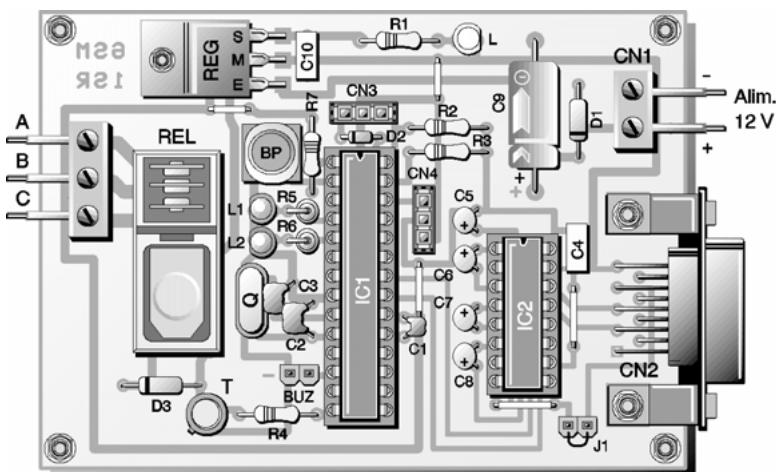


Figure 5.10.
Implantation
des composants.

Liste des composants

R1, R5, R6 : 470 Ω
 R2, R3, R7 : 10 kΩ
 R4 : 1 kΩ
 C1 : 100 nF (pas de 2,54 mm)
 C2, C3 : 22 pF / céramique
 C4, C10 : 100 nF / LCC jaune
 C5, C6, C7, C8 : 1 µF / tantale / 15 V
 C9 : 220 µF / électrolytique / 15 V
 D1 : diode 1N4002
 D2, D3 : diode 1N4148
 L, L1, L2 : Led standard (diamètre 3 mm)
 Q : quartz 20 MHz
 REG : régulateur 7805

BUZ : buzzer piezzo (sans électronique intégrée)
 T : transistor 2N2222
 J1 : barrette HE10 2 contacts + cavalier
 CN1 : bornier à vis 2 plots
 CN2 : connecteur DB9 mâle pour CI / coudé à 90°
 CN3 : connecteur pour câble de programmation (LEXTRONIC)
 CN4 : connecteur pour écran LCD (LEXTRONIC) (facultatif)
 IC1 : PICBASIC PB-3B (LEXTRONIC)
 + support DIL 28 broches (étroit)
 IC2 : MAX232 + support DIL 16 broches
 REL : relais ISKRA TRM 2903 12 V (ou équivalent)
 BP : bouton poussoir type D6

```
DIM Num(16) AS BYTE
DIM NumMem AS BYTE
DIM i AS BYTE
DIM j AS INTEGER
DIM fFlag AS BYTE

'GESTION DU BOUTON POUSSOIR
'-----
```

L'instruction utilisée ici permet de réaliser un accès direct au sous-programme « BP » dès lors qu'un niveau logique bas apparaît sur l'entrée I/O0 où est connecté le bouton poussoir. Cette surveillance est gérée en tâche de fond pendant l'exécution du programme principal. Dès que le bouton poussoir est actionné, le programme principal est interrompu, le sous-programme « BP » est exécuté, le programme principal peut ensuite reprendre son cours normal.

```
ON INT(0)=0 GOSUB BP

'TEST LIAISON SERIE
'-----
```

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit : AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte. Les caractères « AT » suivis du caractère <CR>=13_{dec} sont envoyés par la commande SEROUT. L'instruction SERIN permet d'attendre l'éventuelle réponse « OK » pendant 2 000 ms (soit 2 s). Si les caractères OK sont réceptionnés dans le temps donné, le caractère suivant soit <CR> est placé dans la variable i. Dans le cas contraire le programme saute à la ligne repérée par l'étiquette TEST, i est alors vide, un « bip » est émis par le buzzer. Il suffit de tester le contenu de i pour savoir si la liaison est établie.

```
TEST: BEEP 9
SEROUT TXD,BDS,0,1,[ "AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST

'SELECTION DE L'ALPHABET GSM
'-----
```

SEROUT TXD,BDS,0,1,["AT+CSCS=",34,"GSM",34,13]
DELAY 500

```
'CODE PIN
'-----
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable.

```
SEROUT TXD,BDS,0,1,["AT+CPIN=",34,"7208",34,13]
DELAY 500
```

' INITIALISATION DU ME

Activation de la fonction présentation du numéro. Attention l'utilisation de ce service dépend de votre formule d'abonnement.

```
SEROUT TXD,BDS,0,1,["AT+CLIP=1",13]
DELAY 500
```

' INITIALISATION N° TELEPHONE

Le numéro de téléphone autorisé à piloter le relais est initialisé dans la mémoire eeprom du PicBasic. Pour ne pas interférer avec la partie programme, le stockage se fait dans les 11 derniers emplacements de la mémoire de FF5_{hex} à FFF_{hex}. Cette mémorisation ne se réalise qu'une seule fois car le programme teste avant si l'adresse FF5_{hex} est vide (notez qu'un emplacement vide contient la donnée FF_{hex}). Attention le numéro est en notation internationale, mais sans le signe « + », on a donc un numéro de la forme « 33xxxxxxxxx » (+33 correspond en fait au chiffre 0).

```
IF EEREAD(&HFF5)=&HFF THEN
EEWRITE &HFF5,"3"
EEWRITE &HFF6,"3"
EEWRITE &HFF7,"x"
EEWRITE &HFF8,"x"
EEWRITE &HFF9,"x"
EEWRITE &HFFA,"x"
EEWRITE &HFFB,"x"
EEWRITE &HFFC,"x"
EEWRITE &HFFD,"x"
EEWRITE &HFFE,"x"
EEWRITE &HFFF,"x"
END IF
```

' INITIALISATION DES VARIABLES

```

DEBUT: FOR i=0 TO 15
    Num(i)=0
NEXT i

'ATTENTE SONNERIE (Appel entrant)
'-----

```

Lors d'un appel les sonneries sont matérialisées par le message « RING » qui est envoyé sur la sortie série du téléphone à destination du TE. Lorsque la présentation du numéro est active, le message complémentaire suivant apparaît à chaque sonnerie : +CLIP : <number>,<type>. Le paramètre <number> contient le numéro de téléphone de l'appelant. Le µC va donc scruter l'entrée RXD dans l'attente des caractères « IP ». Dès leur réception les 16 caractères suivants sont placés dans la variable tableau Num.

```

ATT:    SERIN RXD,BDS,0,10000,ATT,[WAIT("IP"),Num(0)~16]
        GOSUB BUZ

```

Ce que contient la variable Num lorsque le téléphone reçoit un appel est montré **tableau 5.7.**

Tableau 5.7.

NUM (0)	NUM (1)	NUM (2)	NUM (3)	NUM (4)	NUM (5)	NUM (6)	NUM (7)	NUM (8)	NUM (9)	NUM (10)	NUM (11)	NUM (12)	NUM (13)	NUM (14)	NUM (15)
:		"	+	3	3	x	x	x	x	x	x	x	x	x	"

Le numéro de l'appel entrant est contenu par les variables Num(3) à Num(12).

```

'COMPARAISON AVEC NUMERO EN EEPROM
'-----

```

Le programme compare le numéro de l'appel entrant au numéro stocké dans l'eeprom du PicBasic. Si au moins un des chiffres qui composent le numéro ne correspond pas, la variable flag est positionnée à 0.

```

flag=1
j=&HFF5

FOR i=4 TO 14
    NumMem=EEREAD(j)
    IF Num(i)<>NumMem THEN flag=0
    j=j+1
NEXT i

```

```

'ACTIVATION / DESACTIVATION DU RELAIS
'-----

```

Si la variable flag est à 1, le relais change d'état. Si flag est à 0 le relais conserve son état, la Led L2 s'illumine pendant 5 s pour signaler que l'appel est rejeté. L'instruction TOGGLE change l'état de la sortie qui lui est associée.

```
IF flag=1 THEN
    TOGGLE 11
    TOGGLE 2
    OUT 3,0
    GOSUB BUZ
ELSE
    OUT 3,1
END IF

DELAY 5000
OUT 3,0
GOTO ATT
```

```
'GESTION DU BP
'-----
```

Sous-programme relatif à la gestion du bouton poussoir. Dès lors que le BP est actionné, le relais et la Led L1 changent d'état et le buzzer est activé.

```
BP:      TOGGLE 11
        TOGGLE 2
        GOSUB BUZ
        DELAY 1000
        RETURN
```

```
'ACTIVATION DU BUZZER
'-----
```

```
BUZ:    FOR i=0 TO 10
        BEEP 9
        NEXT i
        RETURN
```

Tableau 5.8.

Résumé des points importants

1 SORTIE SUR RELAIS (version 1)	
Configuration	Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none">• Code PIN (7208 par défaut)• Numéro de téléphone autorisé à activer/désactiver le relais, attention le numéro est en notation internationale mais sans le signe +, exemple : 33xxxxxxxx (valeur par défaut)	

Programme PICBASIC (version 2) : « 1sr_v2.bas »

Pour ceux qui n'ont pas la chance de posséder la fonctionnalité présentation du numéro, voici une deuxième version du programme. Le montage se contente de détecter l'arrivée d'un appel pour changer l'état du relais. L'inconvénient est que n'importe qui peut piloter votre montage.

```
'DECLARATION DES CONSTANTES
'-----
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16

'DECLARATION DES VARIABLES
'-----
DIM RING(2) AS BYTE
DIM i AS BYTE

'GESTION DU BOUTON POUSSOIR
'-----
ON INT(0)=0 GOSUB BP

'TEST LIAISON SERIE
'-----
TEST: BEEP 9
SEROUT TXD,BDS,0,1,[ "AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500

'INITIALISATION DES VARIABLES
'-----
DEBUT: RING(0)=0
RING(1)=0

'ATTENTE SONNERIE (Appel entrant)
'-----
```

Lors d'un appel les sonneries sont matérialisées par le message « RING » qui est envoyé sur la sortie série du téléphone à destination du TE. C'est ce texte qui va être reconnu par le µC, pour

ce faire celui-ci attend les caractères « RI », les 2 caractères suivants sont placés dans la variable RING.

```
ATT: SERIN RXD,BDS,0,10000,ATT,[WAIT("RI"),RING(0)~2]
```

Ce que contient la variable RING lorsque le téléphone reçoit un appel est montré **tableau 5.9**.

Tableau 5.9.

RING(0)	RING(1)
N	G

```
'ACTIVATION / DESACTIVATION DU RELAIS
```

```
'-----
```

Si les variables RING(0) et RING(1) contiennent respectivement les lettres « N » et « G », l'état du relais est modifié.

```
IF RING(0)<>"N" OR RING(1)<>"G" THEN GOTO ATT
TOGGLE 11
TOGGLE 2
GOSUB BUZ
DELAY 5000
GOTO ATT
```

```
'GESTION DU BP
```

```
'-----
```

```
BP: TOGGLE 11
TOGGLE 2
GOSUB BUZ
DELAY 1000
RETURN
```

```
'ACTIVATION DU BUZZER
```

```
'-----
```

```
BUZ: FOR i=0 TO 10
BEEP 9
NEXT i
RETURN
```

Tableau 5.10.

Résumé des points importants

1 SORTIE SUR RELAIS (version 2)	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none"> • Code PIN (7208 par défaut) 	

4 sorties sur relais

Ce montage associé à un téléphone portable ou à un terminal GSM permet le pilotage de 4 sorties tout ou rien de puissance. L'activation des relais se fait par l'envoi d'un message SMS à partir d'un téléphone portable voire même d'un ordinateur disposant d'une connexion à Internet (SMS gratuits !). Le message doit contenir le numéro du relais à activer (ou à désactiver). À tout moment il est possible de demander l'état des 4 sorties.

Schéma électrique

Quatre lignes du PicBasic I/O8 à I/O11 configurées pour l'occasion en sorties pilotent les 4 relais. Le circuit ULN2803A sert d'amplificateur afin de fournir l'intensité suffisante pour activer les bobines, les 4 sorties restantes sont mises à profit pour signaler visuellement l'état de chacun des relais. IC3 est un amplificateur inverseur de tension, par exemple si l'entrée D0 est à l'état haut, la sortie Q0 est à l'état bas, le relais REL1 est alors actif. Chacune des sorties de l'amplificateur est capable de délivrer une intensité de 500 mA, plus qu'il en faut pour faire coller le relais. Les relais choisis pour cette réalisation sont au format DIL, ils se caractérisent par une petite taille, cependant ils sont capables de véhiculer une intensité permanente de 1,25 A.

Programme PICBASIC (version 1) : « 4sr_v1.bas »

Le début du programme est similaire au montage précédent. Les lignes de programme concernant la gestion de l'écran LCD sont retirées.

'DECLARATION DES CONSTANTES

Déclaration des constantes utilisées par les instructions SERIN et SEROUT. TXD correspond à la ligne I/O16 du PicBasic, RXD correspond à ligne I/O17 et BDS pour Bauds qui définit la vitesse de transmission ici fixée à 9 600 car BDS = 103.

```
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16
```

'DECLARATION DES VARIABLES

```
DIM Tampon(10) AS BYTE
DIM SMS(7) AS BYTE
DIM i AS BYTE
DIM n AS BYTE
```

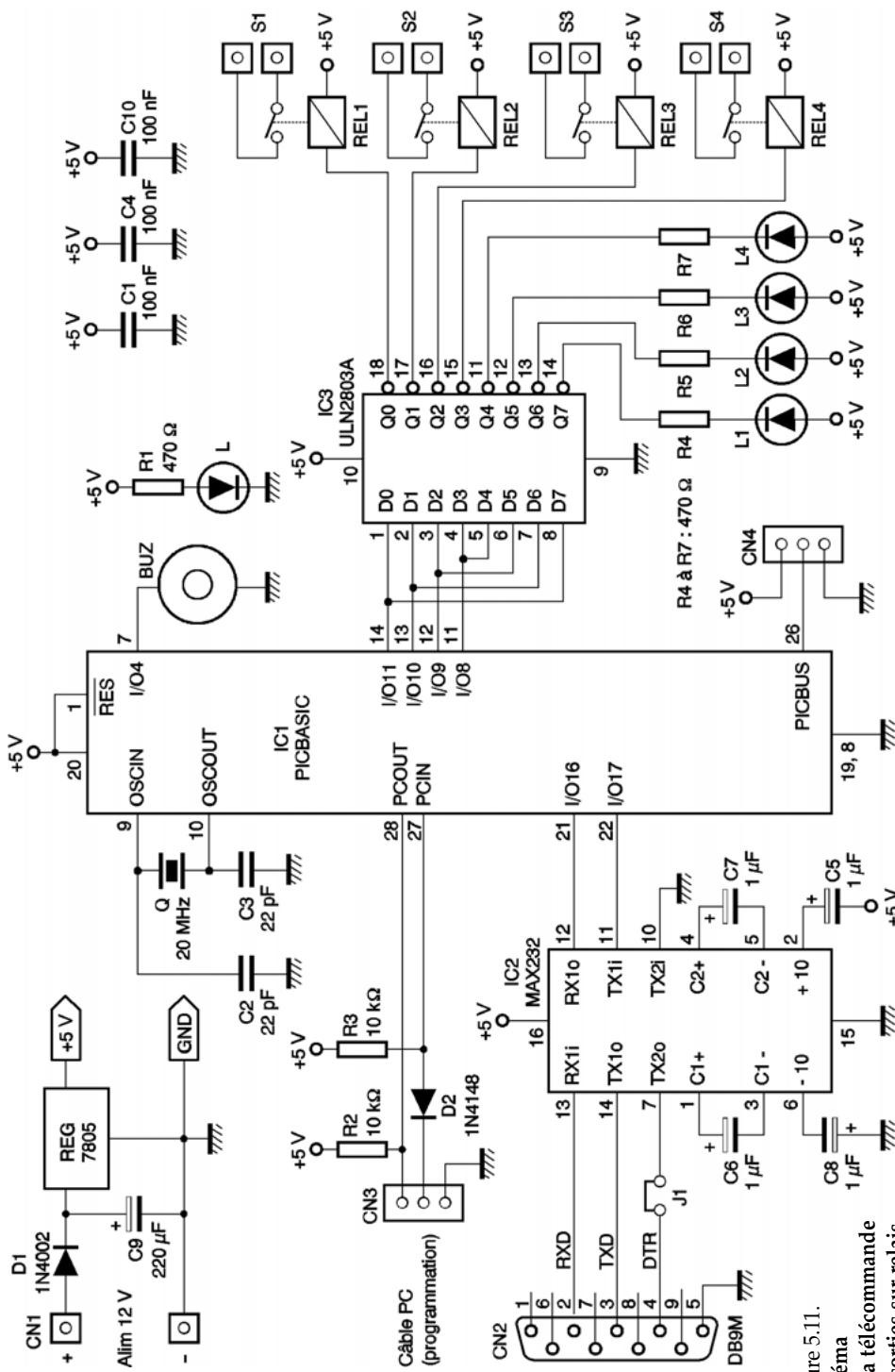


Figure 5.11.
Schéma
de la télécommande
à 4 sorties sur relais.

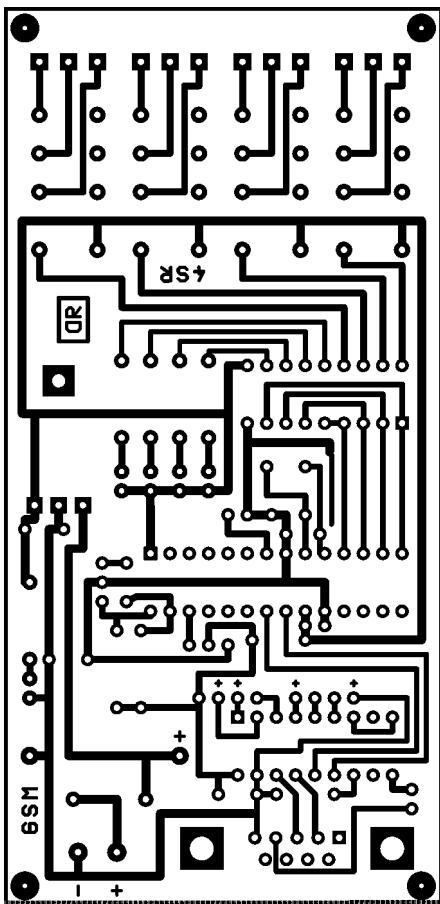


Figure 5.12.
Circuit imprimé.

Liste des composants

R1, R4 à R7 : 470 Ω

R2, R3 : 10 kΩ

C1 : 100 nF (pas de 2,54 mm)

C2, C3 : 22 pF / céramique

C4, C10 : 100 nF / LCC jaune

C5, C6, C7, C8 : 1 µF / tantale / 15 V

C9 : 220 µF / électrolytique / 15 V

D1 : diode 1N4002

D2 : diode 1N4148

L, L1 à L4 : Led standard (diamètre 3 mm)

Q : quartz 20 MHz

REG : régulateur 7805

BUZ : buzzer piezzo (sans électronique intégrée)

J1 : barrette HE10 2 contacts + cavalier

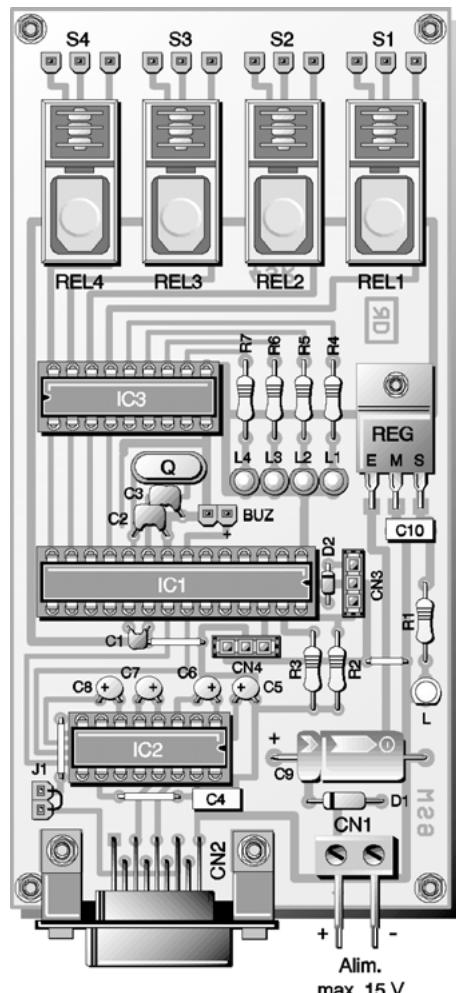


Figure 5.13.
Implantation des composants.

CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle pour CI / coudé à 90°

CN3 : connecteur pour câble de programmation (LEXTRONIC)

CN4 : connecteur pour écran LCD (LEXTRONIC) (facultatif)

IC1 : PICBASIC PB-3B (LEXTRONIC) + support DIL 28 broches (étroit)

IC2 : MAX232 + support DIL 16 broches

IC3 : ULN2803A + support DIL 18 broches

REL1 à REL4 : relais FINDER au format DIL 30.22S modèle 3995 (Arquié composants)

S1 à S4 : cosse poignard

'INITIALISATION DES RELAIS

Au départ du programme tous les relais doivent être inactifs. L'instruction BYTEOUT port,val permet de sortir la valeur binaire de la donnée (val) sur 8 sorties du PicBasic. Chaque sortie est l'image de chaque bit de la valeur binaire donnée. Dans notre cas les relais sont reliés sur les sorties I/O8 à I/O11, il s'agit donc des 4 bits du bloc 1. Le LSB correspond à la broche I/O8, le MSB à la broche I/O15. Les bits 0 à 3 sont donc positionnés à zéro, l'état des autres bits n'a aucune importance car les sorties correspondantes ne sont pas utilisées (mieux vaut tout de même les positionner à l'état bas).

```
BYTEOUT 1,&b00000000
```

'TEST LIAISON SERIE

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit : AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte. Les caractères « AT » suivis du caractère <CR>=13_{dec} sont envoyés par la commande SEROUT. L'instruction SERIN permet d'attendre l'éventuelle réponse « OK » pendant 2 000 ms (soit 2 s). Si les caractères OK sont réceptionnés dans le temps donné, le caractère suivant soit <CR> est placé dans la variable i. Dans le cas contraire le programme saute à la ligne repérée par l'étiquette TEST, i est alors vide, un « bip » est émis par le buzzer. Il suffit de tester le contenu de i pour savoir si la liaison est établie.

```
i=0  
TEST: BEEP 4  
SEROUT TXD,BDS,0,1,[ "AT",13]  
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]  
IF i=0 THEN GOTO TEST
```

'SELECTION DE L'ALPHABET GSM

```
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]  
DELAY 500
```

'CODE PIN

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est

plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable.

```
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500
```

'INITIALISATION DU ME

Le ME est configuré en mode TEXT par la commande « AT+CMGF=1 ». La commande « AT+CNMI=1,1 » indique au ME que chaque nouveau SMS reçu doit être signalé au TE. Ainsi l'arrivée d'un SMS sera signalée par l'envoi au ME de la commande +CMTI: <mem1>,<index>.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGF=1",13]
DELAY 500
SEROUT TXD,BDS,0,1,[ "AT+CNMI=1,1",13]
DELAY 500
```

'INITIALISATION DES VARIABLES

```
DEBUT: FOR i=0 TO 9
      Tampon(i)=0
NEXT i
FOR i=0 TO 6
      SMS(i)=0
NEXT i
```

'ATTENTE RECEPTION SMS

Désormais le µC scrute l'entrée RXD dans l'attente des caractères « TI ». Dès leur réception les 10 caractères suivants sont placés dans la variable Tampon. Une série de 11 bips signale l'arrivée du SMS.

```
ATT: SERIN RXD,BDS,0,10000,ATT,[WAIT("TI"),Tampon(0)~10]
FOR i=0 TO 10
      BEEP 4
NEXT i
```

Comme l'index s'incrémente à chaque nouveau message réceptionné, il est nécessaire d'extraire cette donnée pour savoir où aller lire le message en mémoire. Dans l'état actuel des choses admettons que la donnée Tampon contient ce qui est indiqué **tableau 5.11**.

Tableau 5.11.

Tampon (0)	Tampon (1)	Tampon (2)	Tampon (3)	Tampon (4)	Tampon (5)	Tampon (6)	Tampon (7)	Tampon (8)	Tampon (9)
:	"	M	E	"	,	9	0	0	

On considère dans le programme que la valeur <index> sera codée au maximum sur 3 chiffres. Il est possible, comme le montre cet exemple, que le stockage des SMS se fasse dans la mémoire ME à partir de l'index 900.

```
'LECTURE DU SMS RECU
'-----
```

Le TE configure le ME pour que la lecture soit faite dans la mémoire définie par Tampon(3) et Tampon(4).

```
SEROUT TXD,BDS,0,1,[ "AT+CPMS=",34,Tampon(3),Tampon(4),34,13]
DELAY 500
```

La lecture du SMS est provoquée par la commande « AT+CMGR=<index> ». Si la donnée <index> est codée sur un ou deux chiffres, on récupère des données indésirables (<CR><LF>). Pour les éliminer lors de la reconstitution de l'index du message on s'assure que les données Tampon(7) à Tampon(9) contiennent un caractère compris entre $0_{\text{ASCII}}=48_{\text{dec}}$ et $9_{\text{ASCII}}=57_{\text{dec}}$.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGR=" ]
FOR i=7 TO 9
  IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT TXD,BDS,
  0,1,[Tampon(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
```

Dès la réception des caractères « !! » les 7 caractères suivants sont placés dans la variable SMS. Dans le cas où les caractères « !! » ne sont pas détectés dans les 5 s, le programme passe au label SUITE, comme la variable SMS(0) est vide le SMS est effacé. Si le SMS contient moins de 7 caractères le programme passe également au label SUITE mais comme SMS(0) est dans ce cas différent de zéro le programme suit son cours.

```
SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)=7]
SUITE: SMS(0)=0 THEN GOTO RAZ
```

En l'état actuel du programme, si l'on considère que le SMS envoyé était de la forme « !!REL1,ON », la variable tableau SMS doit contenir ce qui est indiqué **tableau 5.12**.

SMS(0)	SMS(1)	SMS(2)	SMS(3)	SMS(4)	SMS(5)	SMS(6)
R	E	L	1	,	O	N

Tableau 5.12.

SMS(3) contient le numéro du relais à activer, compris entre 1 et 4
 SMS(5) et SMS(6) contiennent l'état que doit prendre le relais à l'issue de la commande, ON ou OF. Il n'est pas interdit de saisir le dernier F de OFF mais celui-ci sera ignoré par le programme.

'GESTION DES RELAIS

La partie gestion des relais est traitée uniquement si la variable SMS(3) est comprise entre 1 et 4, codes ASCII 49_{dec} et 52_{dec}. Si tel est le cas, le contenu des variables SMS(5) et SMS(6) est testé pour savoir si le relais en question doit être activé ou désactivé. Si SMS(5)+SMS(6)= « ON » alors le relais est activé par la commande OUT x,1. Si SMS(5)+SMS(6)= « OF » le relais est désactivé par la commande OUT x,0. Le paramètre x étant le numéro de la broche sur laquelle le relais est connecté.

```

IF SMS(3)<=52 AND SMS(3)>=49 THEN
  IF SMS(5)="0" AND SMS(6)="N" THEN
    IF SMS(3)="1" THEN OUT 8,1
    IF SMS(3)="2" THEN OUT 9,1
    IF SMS(3)="3" THEN OUT 10,1
    IF SMS(3)="4" THEN OUT 11,1
  END IF
  IF SMS(5)="0" AND SMS(6)="F" THEN
    IF SMS(3)="1" THEN OUT 8,0
    IF SMS(3)="2" THEN OUT 9,0
    IF SMS(3)="3" THEN OUT 10,0
    IF SMS(3)="4" THEN OUT 11,0
  END IF
END IF

```

'ENVOI D'UN SMS CONTENANT L'ETAT DES 4 RELAIS

Si la variable SMS(3) ne contient pas un chiffre compris entre 1 et 4 mais un point d'interrogation « ? », ceci dans le cas où le SMS envoyé est de la forme « !!REL? », le montage doit rédiger et envoyer un SMS contenant l'état actuel des 4 relais. Notez la commande OUTSTAT qui permet de consulter l'état des sorties qui pilotent les relais sans en modifier l'état.

La variable n contient successivement les codes ASCII 49_{dec} à 52_{dec}, correspondant aux numéros 1 à 4 des relais.

Pour obtenir systématiquement un accusé de réception pour chaque commande envoyée il suffirait de supprimer la ligne de code « IF SMS(3)="?" THEN » et le « END IF » correspondant.

```
n=0
IF SMS(3)="?" THEN
    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34,"06xxxxxxxx",34,13]
    DELAY 1000
    SEROUT TXD,BDS,0,1,[ "ETAT DES RELAIS : "]
    FOR i=8 TO 11
        n=i+41
        IF OUTSTAT(i)=1 THEN
            SEROUT TXD,BDS,0,1,[ "REL",n,"=ON "]
        ELSE
            SEROUT TXD,BDS,0,1,[ "REL",n,"=OFF "]
        END IF
        NEXT i
        SEROUT TXD,BDS,0,1,[26]
        DELAY 5000
    END IF

'EFFACE LE SMS EN MEMOIRE
'-----
```

Pour terminer, le SMS est systématiquement effacé à l'aide de la commande « AT+CMGD » suivie de l'index, pour éviter une saturation de la mémoire utilisée. Du fait chaque SMS reçu aura le même index.

```
RAZ:   SEROUT TXD,BDS,0,1,[ "AT+CMGD=" ]
       FOR i=7 TO 9
           IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT TXD,BDS,
               0,1,[Tampon(i)]
           NEXT i
           SEROUT TXD,BDS,0,1,[13]
           DELAY 1000
           GOTO DEBUT
```

Résumé des points importants

Voir **Tableau 5.13.**

Programme PICBASIC (version 2) : « 4sr_v2.bas »

Dans le programme précédent, lorsque l'on utilise la commande !!REL? un SMS contenant l'état des 4 relais est envoyé à un numéro qui est précisé en dur dans le programme, ce qui limite l'utilisation du montage à un seul numéro. Si vous utilisez un autre téléphone portable pour envoyer la commande !!REL? vous ne recevrez aucun SMS en retour. Nous allons avec cette deuxième version du programme remédier à ce petit inconvénient.

Tableau 5.13.

4 SORTIES SUR RELAIS (version 1)	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none"> Code PIN (7208 par défaut) Numéro de téléphone pour l'envoi des SMS (06xxxxxxxx par défaut) 	
Commande SMS reçue	Action du montage
!!RELx,ON	Activation du relais numéro x ($4 \geq x \geq 1$)
!!RELx,OF	Désactivation du relais numéro x ($4 \geq x \geq 1$)
!!REL?	Un SMS contenant l'état des 4 relais est envoyé au numéro spécifié dans le programme

Il suffit de faire suivre la commande du numéro où doit être expédié le SMS : !!REL?,06xxxxxxxx.

'DECLARATION DES CONSTANTES

```
'-----  
CONST BDS = 103  
CONST RXD = 17  
CONST TXD = 16
```

'DECLARATION DES VARIABLES

La taille de la variable tableau SMS doit désormais pouvoir contenir jusqu'à 15 caractères.

```
'-----  
DIM Tampon(10) AS BYTE  
DIM SMS(15) AS BYTE  
DIM i AS BYTE  
DIM n AS BYTE  
  
'INITIALISATION DES RELAIS  
'-----  
BYTEOUT 1,&b00000000  
  
'TEST LIAISON SERIE  
'-----  
TEST: BEEP 4  
SEROUT TXD,BDS,0,1,[ "AT",13]  
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]  
IF i=0 THEN GOTO TEST
```

INTERFACES GSM

```
'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,["AT+CSCS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
SEROUT TXD,BDS,0,1,["AT+CPIN=",34,"7208",34,13]
DELAY 500

'INITIALISATION DU ME
'-----
SEROUT TXD,BDS,0,1,["AT+CMGF=1",13]
DELAY 500
SEROUT TXD,BDS,0,1,["AT+CNMI=1,1",13]
DELAY 500

'INITIALISATION DES VARIABLES
'-----
DEBUT: FOR i=0 TO 9
    Tampon(i)=0
NEXT i
FOR i=0 TO 14
    SMS(i)=0
NEXT i

'ATTENTE RECEPTION SMS
'-----
ATT: SERIN RXD,BDS,0,10000,ATT,[WAIT("TI"),Tampon(0)~10]
FOR i=0 TO 10
    BEEP 4
NEXT i

'LECTURE DU SMS RECU
'-----
```

Le TE configure le ME pour que la lecture soit faite dans la mémoire définie par Tampon(3) et Tampon(4). La lecture du SMS est provoquée par la commande « AT+CMGR=<index> ». Dès la réception des caractères « !! » les 15 caractères suivants sont placés dans la variable SMS.

```
SEROUT TXD,BDS,0,1,["AT+CPMS=",34,Tampon(3),Tampon(4),34,13]
DELAY 500
SEROUT TXD,BDS,0,1,["AT+CMGR="]
FOR i=7 TO 9
    IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT TXD,BDS,
        0,1,[Tampon(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
SERIN RXD,BDS,0,5000,ATT,[WAIT("!!"),SMS(0)~15]
ATT: IF SMS(0)=0 THEN GOTO RAZ
```

En l'état actuel du programme, si l'on considère que le SMS envoyé était de la forme « !!REL?,0601234567 », la variable tableau SMS doit contenir ce qui est indiqué **tableau 5.14.**

Tableau 5.14.

SMS (0)	SMS (1)	SMS (2)	SMS (3)	SMS (4)	SMS (5)	SMS (6)	SMS (7)	SMS (8)	SMS (9)	SMS (10)	SMS (11)	SMS (12)	SMS (13)	SMS (14)
R	E	L	?	,	0	6	0	1	2	3	4	5	6	7

SMS(3) contient un point d'interrogation qui signale au PicBasic qu'il doit envoyer un SMS contenant l'état des 4 relais au numéro spécifié par SMS(5) à SMS(14).

'GESTION DES RELAIS

'-----'

Noter que les autres commandes, !!RELx,ON et !!RELx,OF, fonctionnent de la même manière que dans la version 1 du programme.

```
IF SMS(3)<=52 AND SMS(3)>=49 THEN
    IF SMS(5)="0" AND SMS(6)="N" THEN
        IF SMS(3)="1" THEN OUT 8,1
        IF SMS(3)="2" THEN OUT 9,1
        IF SMS(3)="3" THEN OUT 10,1
        IF SMS(3)="4" THEN OUT 11,1
    END IF
    IF SMS(5)="0" AND SMS(6)="F" THEN
        IF SMS(3)="1" THEN OUT 8,0
        IF SMS(3)="2" THEN OUT 9,0
        IF SMS(3)="3" THEN OUT 10,0
        IF SMS(3)="4" THEN OUT 11,0
    END IF
END IF
```

'ENVOI UN SMS CONTENANT L'ETAT DES 4 RELAIS

'-----'

Si la variable SMS(3) ne contient pas un chiffre compris entre 1 et 4 mais un point d'interrogation « ? », ceci dans le cas où le SMS envoyé est de la forme « !!REL?,06xxxxxxx », le montage doit rédiger et envoyer un SMS contenant l'état actuel des 4 relais. Cette fois le numéro du téléphone n'est pas figé dans le programme, il est contenu par les variables SMS(5) à SMS(14). La boucle FOR/NEXT permet de balayer les 10 variables pour reconstituer le numéro. Si aucun numéro n'est spécifié, ce qui est vrai si SMS(4) ne contient pas une virgule, le message est envoyé au numéro par défaut indiqué en dur dans le programme.

```
n=0
IF SMS(3)="?" THEN
  IF SMS(4)=".," THEN
    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
    FOR i=5 TO 14
      SEROUT TXD,BDS,0,1,[SMS(i)]
    NEXT i
    SEROUT TXD,BDS,0,1,[34,13]
  ELSE
    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34,"06xxxxxxxx",34,13]
  END IF

  DELAY 1000
  SEROUT TXD,BDS,0,1,[ "ETAT DES RELAIS : "]
  FOR i=8 TO 11
    n=i+41
    IF OUTSTAT(i)=1 THEN
      SEROUT TXD,BDS,0,1,[ "REL",n,"=ON "]
    ELSE
      SEROUT TXD,BDS,0,1,[ "REL",n,"=OFF "]
    END IF
    NEXT i
    SEROUT TXD,BDS,0,1,[26]
    DELAY 5000
  END IF

'EFFACE LE SMS EN MEMOIRE
'-----
RAZ:  SEROUT TXD,BDS,0,1,[ "AT+CMGD="]
FOR i=7 TO 9
  IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT TXD,BDS,
0,1,[Tampon(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
DELAY 1000
GOTO DEBUT
```

Résumé des points importants

Voir Tableau 5.15.

4 sorties sur triacs

Ce montage associé à un téléphone portable ou à un terminal GSM permet le pilotage de 4 charges alimentées par la tension du secteur, la partie puissance fait cette fois appel à des triacs. Contrairement aux relais, les triacs ont un fonctionnement purement électronique, donc statique, ce qui est gage d'une durée de vie plus importante.

Comme pour le montage précédent, l'activation des triacs se fait par l'envoi d'un message SMS à partir d'un téléphone portable

Tableau 5.15.

4 SORTIES SUR RELAIS (version 2)	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none">Code PIN (7208 par défaut)Numéro de téléphone par défaut pour l'envoi des SMS	
Commande SMS reçue	Action du montage
!!RELx,ON	Activation du relais numéro x ($4 \geq x \geq 1$)
!!RELx,OF	Désactivation du relais numéro x ($4 \geq x \geq 1$)
!!REL?	Un SMS contenant l'état des 4 relais est envoyé au numéro spécifié en dur dans le programme
!!REL?,06xxxxxxxx	Un SMS contenant l'état des 4 relais est envoyé au numéro indiqué

ou d'un ordinateur. Le message doit contenir le numéro du triac à activer (ou à désactiver). À tout moment il est possible de demander l'état des 4 sorties.

Schéma électrique

Quatre lignes du PicBasic I/O8 à I/O11 configurées en sorties pilotent les 4 triacs. L'utilisation de la tension du secteur nous conduit à effectuer une isolation galvanique afin de protéger correctement l'électronique placée en amont. Cette isolation est réalisée à l'aide d'un optocoupleur (du type MOC3041), un tel circuit se compose de deux parties distinctes (isolation galvanique de 7 500 V) : la première est constituée d'une diode infrarouge qui va venir mettre en conduction le triac contenu dans la deuxième partie. Il dispose également d'un dispositif qui détecte le passage à zéro de la tension du secteur afin d'éviter de générer des parasites lors de l'alimentation de la charge. Le courant de l'ordre de 10 mA, nécessaire à l'activation de la diode infrarouge, est généré par la sortie du PicBasic, la limitation de l'intensité est assurée par une résistance de 470 Ω. La faible puissance du triac interne à l'optocoupleur ($I_{max} = 100$ mA) ne permet pas l'alimentation directe d'une charge importante. Un deuxième triac mis en cascade permet de disposer d'une puissance beaucoup plus importante. Toutefois, compte tenu de la largeur des pistes de la carte, il est conseillé de ne pas dépasser 200 W par sortie. Le composant référencé VR est une varistance qui permet de protéger le

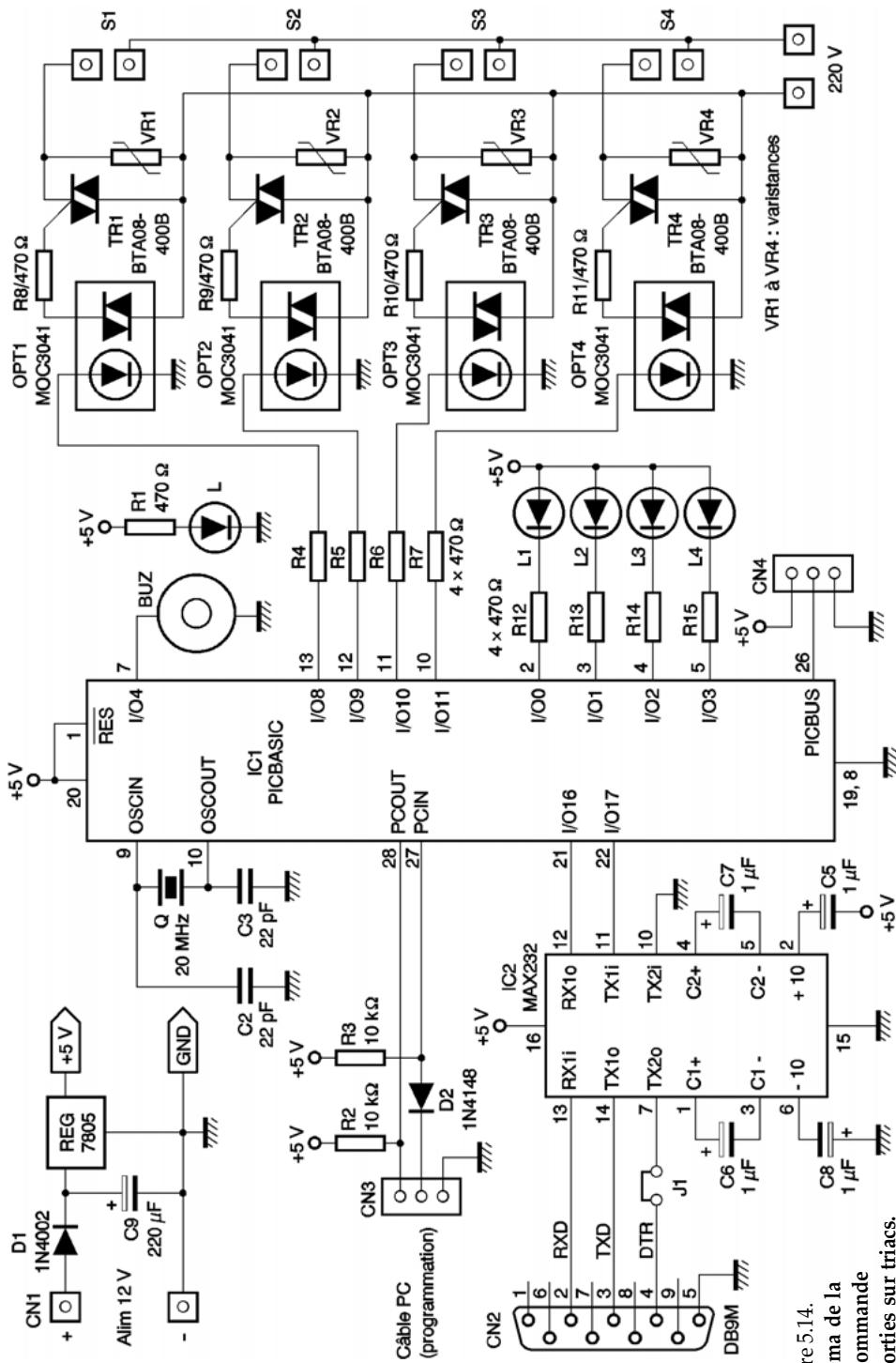


Figure 5.14.
Schéma de la
télécommande
à 4 sorties sur triacs.

montage lors du pilotage d'une charge inductive, les phénomènes d'auto-induction lors de l'établissement et la coupure du courant peuvent détériorer le triac. Chaque triac possède donc une varistance montée en parallèle. Ce composant voit son impédance chuter très fortement en présence d'une surtension (tension > tension nominale de 250 V), protégeant ainsi le circuit placé en aval, en l'occurrence le triac. Notez la présence de 4 Led sur les broches I/O0 à I/O3 qui signalent visuellement l'état des triacs.

Programme PICBASIC : « 4st.bas »

Il est bien entendu possible de réutiliser le programme du montage « 4 sorties sur relais », cependant nous allons en profiter pour mettre en œuvre ici une deuxième méthode de programmation qui va éviter au PicBasic d'attendre en permanence que le ME lui signale l'arrivée d'un SMS. Périodiquement, c'est le TE qui va consulter la mémoire du ME pour savoir si un nouveau SMS y est stocké. Cette méthode est plus simple et permet surtout au PicBasic d'effectuer d'autres tâches entre deux consultations. Le seul petit inconvénient est qu'il faut savoir à l'avance dans quelle mémoire va être stocké le prochain SMS et quel sera son index.

Nous allons également ajouter la commande : TRI.,DATA qui positionne simultanément les 4 triacs à l'état spécifié par la donnée DATA comprise entre 00 et 15_{dec}.

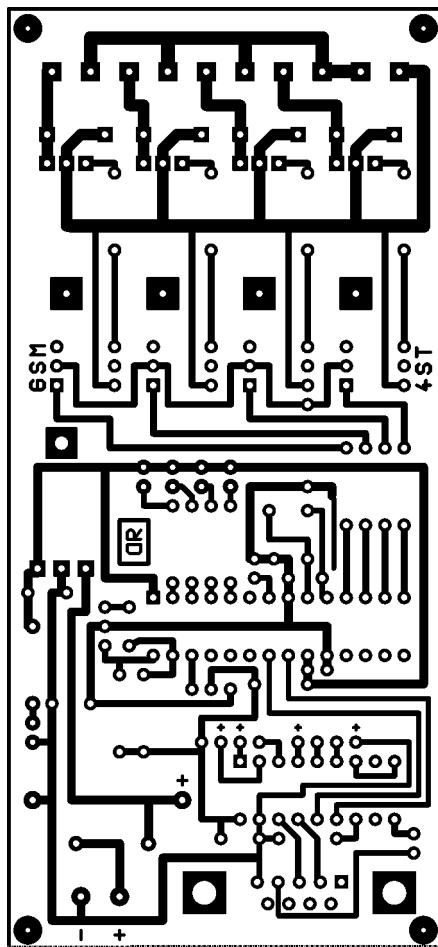
```
' DECLARATION DES CONSTANTES
'-----
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16

'DECLARATION DES VARIABLES
'-----
DIM index(3) AS BYTE
DIM SMS(15) AS BYTE
DIM DATA AS BYTE
DIM i AS BYTE
DIM n AS BYTE

'INITIALISATION DES TRIACS ET DES LEDS
'-----
BYTEOUT 1,&b00000000
OUT 0,1
OUT 1,1
OUT 2,1
OUT 3,1

'TEST LIAISON SERIE
'-----
i=0
```

Figure 5.15.
Circuit imprimé.



```
TEST: BEEP 4
SEROUT TXD,BDS,0,1,["AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,["AT+CSCS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone

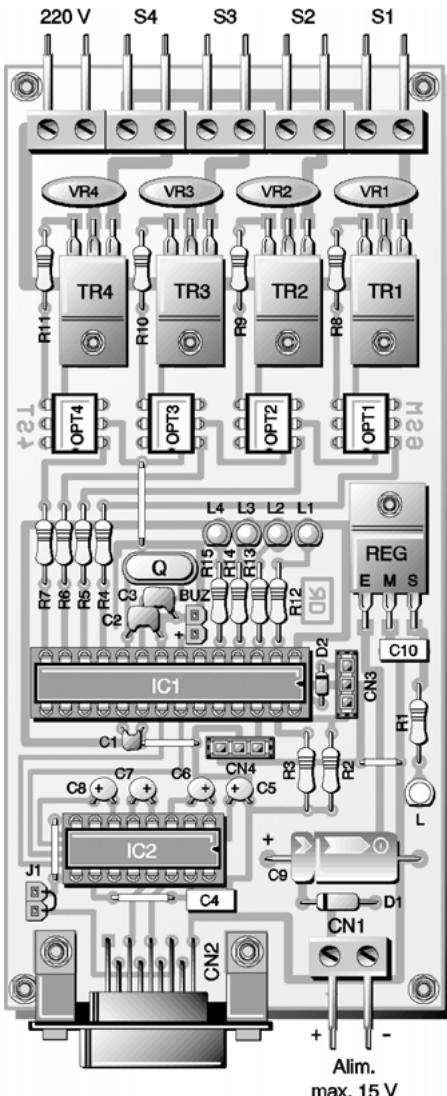


Figure 5.16.
Implantation
des composants.

Liste des composants

R1, R4 à R15 : 470 Ω

R2, R3 : 10 k Ω

C1 : 100 nF (pas de 2,54 mm)

C2, C3 : 22 pF / céramique

C4, C10 : 100 nF / LCC jaune

C5, C6, C7, C8 : 1 μ F / tantalé / 15 V

C9 : 220 μ F / électrolytique / 15 V

D1 : diode 1N4002

D2 : diode 1N4148

L, L1 à L4 : Led standard
(diamètre 3 mm)

Q : quartz 20 MHz

REG : régulateur 7805

BUZ : buzzer piezzo
(sans électronique intégrée)

J1 : barrette HE10 2 contacts + cavalier

CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle
pour CI / coudé à 90°

CN3 : connecteur pour câble
de programmation (LEXTRONIC)

CN4 : connecteur pour écran LCD
(LEXTRONIC) (facultatif)

IC1 : PICBASIC PB-3B (LEXTRONIC)
+ support DIL 28 broches (étroit)

IC2 : MAX232 + support DIL 16 broches

OPT1 à OPT4 : optocoupleur MOC3041

TR1 à TR4 : TRIAC BTA08-400B

VR1 à VR4 : varistance 220 V

classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable.

```
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500
```

```
'INITIALISATION DU ME EN MODE TEXT
'-----
```

Comme nous n'avons pas besoin que le ME avertisse le TE de l'arrivée d'un nouveau SMS, la commande « AT+CNMI=1,1 » est retirée. Ne subsiste que la commande qui sélectionne le mode TEXT.

```
SEROUT TXD,BDS,0,1,["AT+CMGF=1",13]
DELAY 500
```

```
'SELECTION MEMOIRE ET INDEX POUR LECTURE SMS
```

Comme nous l'avons dit plus haut, la mémoire de stockage et l'index sont figés. Il faut donc déterminer ces deux paramètres à l'avance. Concernant la mémoire <mem1>, on considère que les SMS envoyés par les particuliers ne possèdent pas de classe. Cela signifie que le mobile qui reçoit ce genre de SMS le stocke dans la mémoire ME. Les autres types de mémoire sont surtout utilisés par les opérateurs. Si votre téléphone le supporte vous pouvez utiliser le paramètre MT qui permet aux commandes de lecture de SMS de travailler avec toutes les mémoires (possible sur le SIEMENS MC35 et TC35).

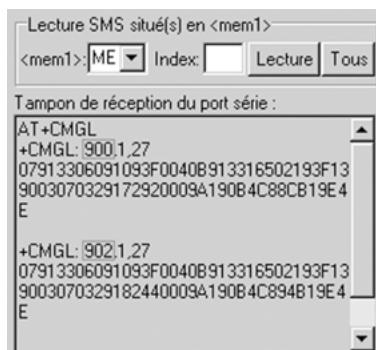


Figure 5.17.
Index.

Ici nous avons choisi la mémoire du téléphone d'où « AT+CPMS= "ME" ». De même, la variable index doit être initialisée avec l'index que portera le prochain SMS réceptionné. Il correspond au premier emplacement de libre dans la mémoire sélectionnée. Pour le déterminer, vous pouvez utiliser le logiciel « convertSMS2 », après avoir sélectionné la mémoire, soit « ME » dans notre cas, cliquez sur le bouton « Tous », il suffit de relever l'index du premier emplacement de libre.

Dans l'exemple ci-contre on constate que la mémoire « ME » contient 2 messages, le premier est situé à l'index 900, le deuxième est situé à l'index 902. L'index 901 est donc libre, c'est là que le prochain message reçu sera stocké.

Remarque : Si vous avez un index codé sur un ou deux chiffres, il suffit de mettre les variables non utilisées à nul. Par exemple si index = 1 (index de base pour le MC35 et TC35 de SIEMENS), il faudra modifier le programme comme ceci : index(0)="" ; index(1)="" ; index(2)="1".

```
SEROUT TXD,BDS,0,1,[ "AT+CPMS=",34,"ME",34,13]
DELAY 500
index(0)="9":index(1)="0":index(2)="1"

'INITIALISATION DES VARIABLES
'-----
DEBUT: FOR i=0 TO 14
SMS(i)=0
NEXT i

'REGARDE SI RECEPTION D'UN SMS
'-----
```

Le programme grâce à la commande « AT+CMGR » regarde si l'emplacement mémoire indiqué par l'index contient un SMS. Si l'emplacement est vide le programme saute à l'étiquette RAZ. Dans le cas contraire si le texte contient les caractères « !! », les 15 caractères suivants, qui contiennent la commande, sont stockés dans la variable SMS.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGR=" ]
FOR i=0 to 2
IF index(i)>=48 AND index(i)<=57 THEN SEROUT TXD,BDS,
0,1,[index(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
SERIN RXD,BDS,0,5000,ATT,[WAIT("!!"),SMS(0)~15]
ATT: IF SMS(0)=0 THEN GOTO RAZ
FOR i=0 TO 10
BEEP 4
NEXT i
```

```
'GESTION DES TRIACS
'-----
```

La commande contenue dans la variable SMS est décodée pour savoir s'il faut activer/désactiver un triac ou envoyer un SMS contenant l'état des 4 triacs. Exemple : la commande !!TRI4,OF désactive le triac numéro 4.

```
IF SMS(3)<=52 AND SMS(3)>=49 THEN
IF SMS(5)="0" AND SMS(6)="N" THEN
IF SMS(3)="1" THEN OUT 8,1
IF SMS(3)="2" THEN OUT 9,1
```

```

IF SMS(3)="3" THEN OUT 10,1
IF SMS(3)="4" THEN OUT 11,1
END IF
IF SMS(5)="0" AND SMS(6)="F" THEN
    IF SMS(3)="1" THEN OUT 8,0
    IF SMS(3)="2" THEN OUT 9,0
    IF SMS(3)="3" THEN OUT 10,0
    IF SMS(3)="4" THEN OUT 11,0
END IF
END IF

```

Ajout de la commande qui permet de piloter simultanément les 4 triacs. Notez qu'à la place du numéro du triac il faut saisir un point. Les données SMS(5) et SMS(6) contiennent la valeur à appliquer sur les sorties qui pilotent les triacs (**tableau 5.16**).

Tableau 5.16.

Commande	DATA (dec)	TRIAC n°			
		4	3	2	1
!!TRI.,00	0	0	0	0	0
!!TRI.,01	1	0	0	0	1
!!TRI.,02	2	0	0	1	0
!!TRI.,03	3	0	0	1	1
!!TRI.,04	4	0	1	0	0
!!TRI.,05	5	0	1	0	1
!!TRI.,06	6	0	1	1	0
!!TRI.,07	7	0	1	1	1
!!TRI.,08	8	1	0	0	0
!!TRI.,09	9	1	0	0	1
!!TRI.,10	10	1	0	1	0
!!TRI.,11	11	1	0	1	1
!!TRI.,12	12	1	1	0	0
!!TRI.,13	13	1	1	0	1
!!TRI.,14	14	1	1	1	0
!!TRI.,15	15	1	1	1	1
1 = ON, 0 = OFF					

Notez que la commande `!!TRI,,00` désactive simultanément les 4 triacs, alors que la commande `!!TRI,,15` active simultanément les 4 triacs.

`SMS(5)` et `SMS(6)` contiennent la valeur de consigne que doivent prendre les 4 triacs. Cette valeur ne peut pas être utilisée directement, car elle est codée en ASCII. Malheureusement il n'existe aucune instruction en langage PicBasic pour convertir une valeur de type texte en donnée de type BYTE. Attention il ne suffit pas de multiplier `SMS(5)` par 10, `SMS(6)` par 1 et d'additionner ces deux valeurs pour obtenir la valeur numérique souhaitée. Avant de faire ce calcul il faut retrancher 48_{dec} à chaque donnée (48 correspond au code ASCII du chiffre 0).

```
IF SMS(3)=". ." THEN
  SMS(5)=SMS(5)-48
  SMS(5)=SMS(5)*10
  SMS(6)=SMS(6)-48
  DATA = SMS(5)+SMS(6)
  BYTEOUT 1,DATA
END IF
```

Mise à jour des Led en fonction de l'état des triacs :

```
IF OUTSTAT(8) =0 THEN OUT 0,1 ELSE OUT 0,0
IF OUTSTAT(9) =0 THEN OUT 1,1 ELSE OUT 1,0
IF OUTSTAT(10)=0 THEN OUT 2,1 ELSE OUT 2,0
IF OUTSTAT(11)=0 THEN OUT 3,1 ELSE OUT 3,0
```

'ENVOI D'UN SMS CONTENANT L'ETAT DES 4 TRIACS

Si la variable `SMS(3)` ne contient pas un chiffre compris entre 1 et 4 mais un point d'interrogation « ? », ceci dans le cas où le SMS envoyé est de la forme « `!!TRI?,06xxxxxxxx` », le montage doit rédiger et envoyer un SMS contenant l'état actuel des 4 triacs. Cette fois le numéro du téléphone n'est pas figé dans le programme, il est contenu par les variables `SMS(5)` à `SMS(14)`. La boucle `FOR/NEXT` permet de balayer les 10 variables pour reconstituer le numéro. Si aucun numéro n'est spécifié, ce qui est vrai si `SMS(4)` ne contient pas le code ASCII du symbole virgule, le message est envoyé au numéro par défaut indiqué en dur dans le programme.

```
n=0
IF SMS(3)="? ." THEN
  IF SMS(4)=", ." THEN
    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
    FOR i=5 TO 14
      SEROUT TXD,BDS,0,1,[SMS(i)]
```

```
NEXT i
SEROUT TXD,BDS,0,1,[34,13]
ELSE
SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34,"06xxxxxxxx",34,13]
END IF
DELAY 1000
SEROUT TXD,BDS,0,1,[ "ETAT DES TRIACS : "]
FOR i=8 TO 11
n=i+41
IF OUTSTAT(i)=1 THEN
SEROUT TXD,BDS,0,1,[ "TRIAC",n,"=ON "]
ELSE
SEROUT TXD,BDS,0,1,[ "TRIAC",n,"=OFF "]
END IF
NEXT i
SEROUT TXD,BDS,0,1,[26]
DELAY 5000
END IF

'EFFACE LE SMS EN MEMOIRE
'-----
```

Cette partie du programme permet d'effacer systématiquement le SMS en mémoire, ainsi le prochain SMS réceptionné aura toujours le même index. Ceci évite de prévoir une incrémentation de la variable index et surtout de saturer la mémoire utilisée. Notez que cette partie de programme est dans certains cas appelée même si aucun SMS n'est à effacer, le ME répond par un message d'erreur qui est ignoré par le programme.

```
RAZ: SEROUT TXD,BDS,0,1,[ "AT+CMGD="]
FOR i=0 TO 2
IF index(i)>=48 AND index(i)<=57 THEN SEROUT
TXD,BDS,0,1,[index(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
DELAY 1000
GOTO DEBUT
```

Résumé des points importants

Voir Tableau 5.17.

4 sorties analogiques

Le présent montage n'utilise pas de réseau R-2R comme on aurait pu s'y attendre mais des potentiomètres numériques. Le potentiomètre numérique remplit exactement la même fonction que son ancêtre mécanique. La différence est que pour faire varier sa résistance on ne fait plus tourner un axe mais on envoie un mot

Tableau 5.17.

4 SORTIES SUR TRIACS	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Éléments du programme PicBasic à modifier	
Commande SMS reçue	Code PIN (7208 par défaut)
	Mémoire lecture SMS <mem1> (ME par défaut)
Action du montage	Index du prochain SMS reçu (900 par défaut)
	Numéro de téléphone par défaut pour l'envoi des SMS
!!TRIx,ON	Activation du triac numéro x ($4 \geq x \geq 1$)
!!TRIx,OF	Désactivation du triac numéro x ($4 \geq x \geq 1$)
!!TRI,data	Positionne les 4 triacs suivant la valeur contenue par data ($15 \geq data \geq 00$)
!!TRI?	Un SMS contenant l'état des 4 triacs est envoyé au numéro spécifié en dur dans le programme
!!TRI?,06xxxxxxxx	Un SMS contenant l'état des 4 triacs est envoyé au numéro indiqué

de commande par le biais d'une liaison série synchrone nécessitant que 3 fils. On peut donc facilement interfaçer ce type de composant avec un PicBasic et envisager de piloter par SMS tout montage utilisant un potentiomètre (alimentation variable, filtre programmable, timer, gradateur...). L'ajout de straps permet de transformer le montage en Convertisseur Numérique Analogique, on obtient dans ce cas sur les sorties une tension comprise entre 0 et + 5 V que l'on peut faire varier par pas de 20 mV.

Schéma électrique

Voir Figures 5.18, 5.19 et 5.20..

Caractéristiques

Dans la famille des potentiomètres numériques proposés par le constructeur Analog Devices nous trouvons le AD8400 qui dispose d'un canal, le AD8402 de deux canaux et le AD8403 de quatre canaux. Bien que le montage présenté ici possède 4 sorties, nous avons opté pour l'utilisation de 4 circuits AD8400 indépendants. Il est ainsi possible de choisir des valeurs de résistance nominale différentes sur chaque sortie. Une sortie correspond à un potentiomètre que l'on peut contrôler numériquement. Les valeurs nominales disponibles sont 1 kΩ, 10 kΩ, 50 kΩ ou 100 kΩ.

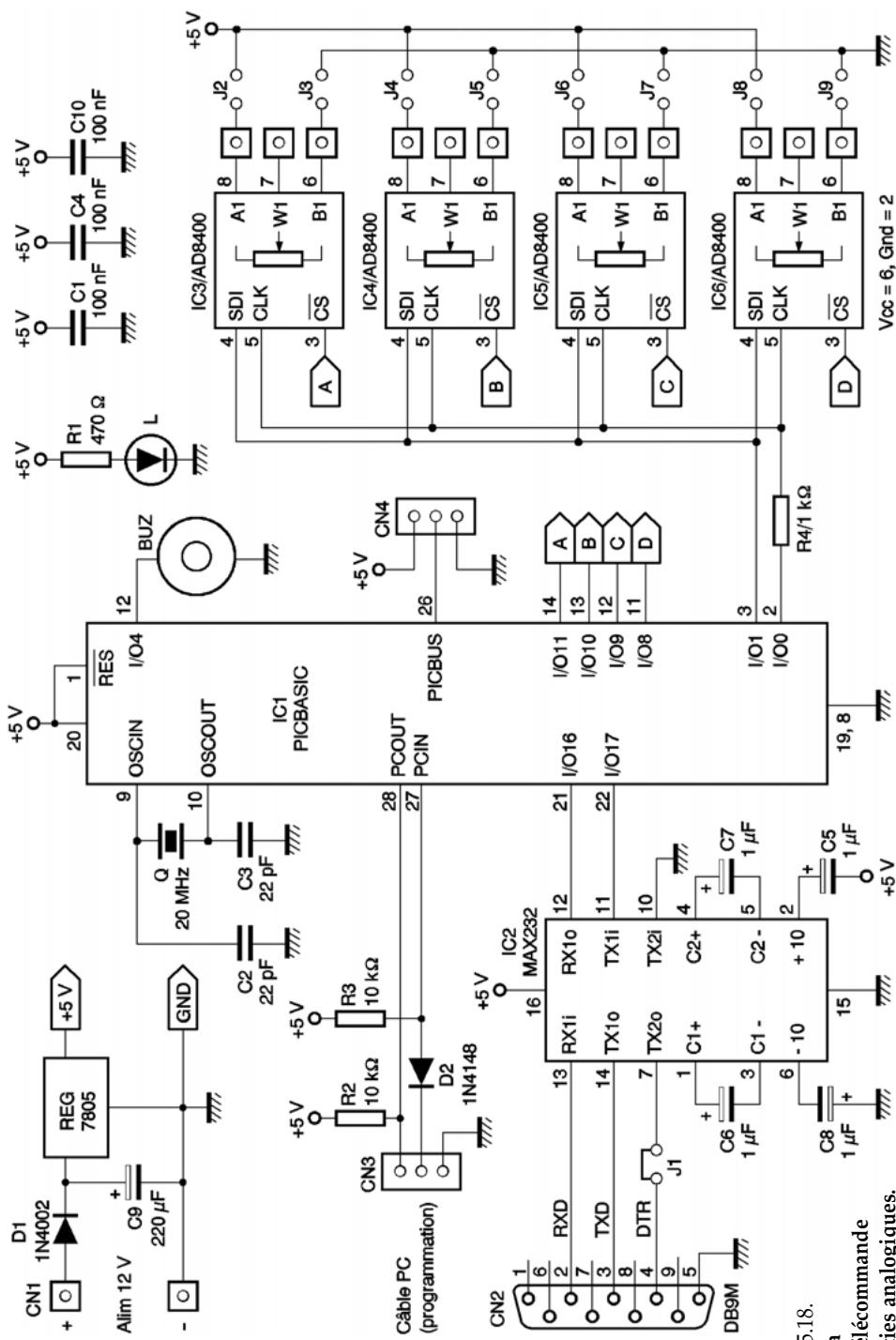


Figure 5.18.
Schéma
de la télécommande
à 4 sorties analogiques.

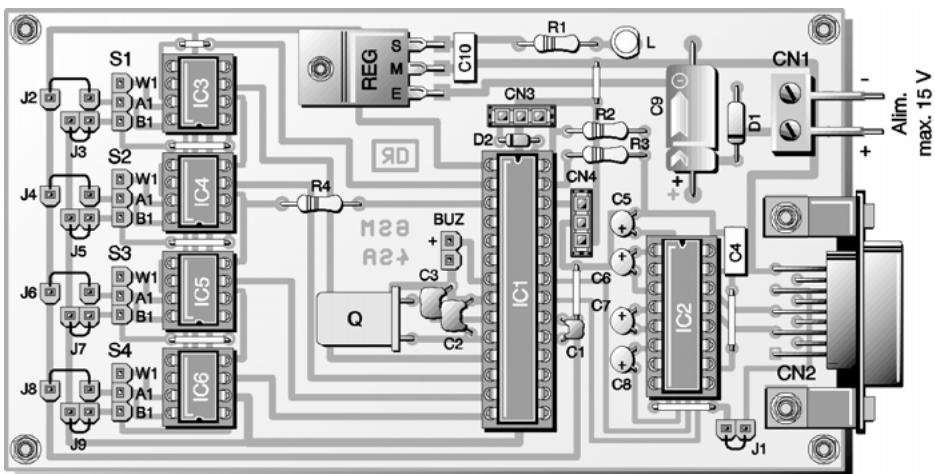
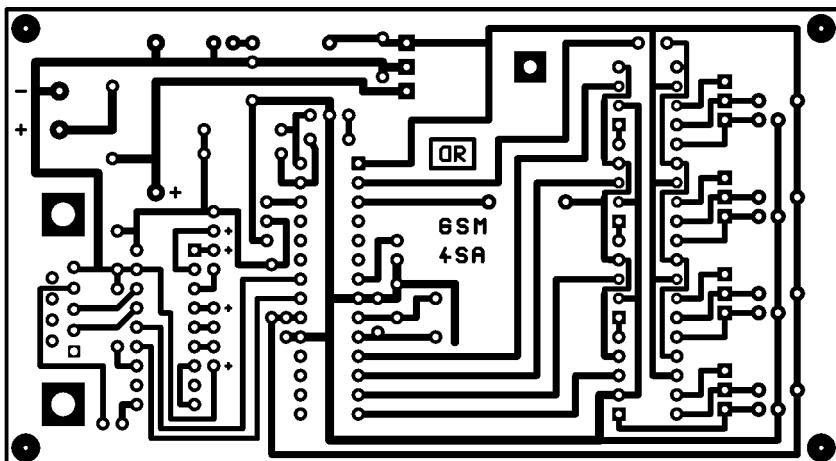


Figure 5.19. (en haut)

Circuit imprimé.

 Figure 5.20. (en bas)
 Implantation des composants.

Liste des composants

- R1 : 470 Ω
- R2, R3 : 10 k Ω
- R4 : 1 k Ω
- C1 : 100 nF (pas de 2,54 mm)
- C2, C3 : 22 pF / céramique
- C4, C10 : 100 nF / LCC jaune
- C5, C6, C7, C8 : 1 μ F / tantale / 15 V
- C9 : 220 μ F / électrolytique / 15 V
- D1 : diode 1N4002
- D2 : diode 1N4148
- L : Led standard
- Q : quartz 20 MHz
- REG : régulateur 7805

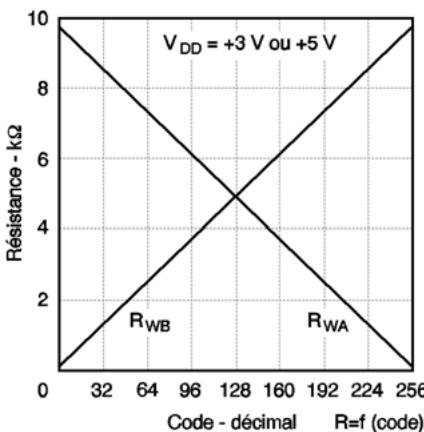
Nous avons choisi pour notre montage 4 potentiomètres de $10\text{ k}\Omega$. La résolution du potentiomètre est de 8 bits, c'est-à-dire que l'on peut accéder à $2^8 = 256$ positions élémentaires. Chacune de ces 256 positions est séparée par une valeur égale à $10\,000\,\Omega / 256 = 39\,\Omega$. Il suffit de transférer un mot de 8 bits dans un registre pour déterminer la valeur de la résistance du potentiomètre. Soit A et B les deux points fixes et W le point variable (curseur). Entre les points A et B on retrouve la valeur nominale du potentiomètre soit $10\text{ k}\Omega$. La position du curseur W est pilotée par la valeur contenue dans le registre de commande. La mise à jour de ce registre se fait par un bus de type SPI piloté par le PicBasic. Il s'agit d'une liaison série synchrone. On retrouve donc un signal d'horloge (CLK) connecté à la broche I/O0, une ligne pour le transfert des données (SDI) connectée à la broche I/O1 et une ligne de validation (CS), les 4 sorties I/O8 à I/O11 reliées aux entrées CS permettent de sélectionner indépendamment chacun des circuits AD8400. Chaque mot transféré comporte 10 bits. Le niveau présenté sur la ligne de donnée est transféré dans le registre à décalage sur un front montant du signal d'horloge alors que la ligne CS est à l'état bas. Ce processus est répété 10 fois puis la ligne CS est mise à l'état haut ce qui déclenche la mise à jour de la position de W. Les deux premiers bits transférés déterminent l'adresse du potentiomètre. Dans le cadre d'une utilisation de l'AD8400 qui ne dispose que d'un canal, les bits d'adresse A0 et A1 seront toujours égaux à zéro. Les 8 autres bits déterminent la position du curseur W, le transfert du mot s'effectue par bit de poids décroissant. Le premier point est accessible en envoyant la valeur 0. Le curseur est alors en butée sur le point B, il subsiste entre ces deux points une résistance résiduelle d'environ $50\,\Omega$. La seconde position est obtenue en envoyant le mot 01_{dec} on obtient entre les points W et B une résistance équivalente de $89\,\Omega$. Chaque position est en fait calculée par la relation suivante : $R_{WB}(D_x) = (D_x / 255) \times R_{BA} + R_w$ avec $R_w = 50\,\Omega$. La résistance maximale est obtenue en transférant la valeur 255_{dec} on obtient alors entre le point B et W une résistance égale à $10\,011\,\Omega$. On peut évidemment en déduire aussi la résistance entre les points W et A en utilisant la relation suivante : $R_{WA}(D_x) = ((255 - D_x) / 255) \times R_{BA} + R_w$ (**tableau 5.18**).

Utilisation du circuit en convertisseur numérique-analogique

Il est aussi très facile d'obtenir en sortie du potentiomètre non plus une résistance mais une tension, simplement en reliant le point A au + 5 V (straps J2, J4, J6 et J8) et le point B au Gnd (straps J1, J3, J5 et J7). La tension comprise entre le point W et le point B se calcule de la manière suivante : $V_{wB}(D_x) = D_x / 255 \times V_{AB} + V_B$.

Code (Dec)	$R_{WB}(\Omega)$	$R_{WA}(\Omega)$
255	10 011	89
128	5 050	5 050
1	89	10 011
0	50	10 050

Tableau 5.18.

Figure 5.21.
Graphique $R = f(D)$.

On obtient ni plus ni moins qu'un convertisseur numérique-analogique (CNA). Concernant notre montage, nous avons $V_{AB} = +5 \text{ V}$ et $V_B = 0 \text{ V}$ (on néglige la résistance résiduelle), la formule précédente devient :

$$V_{WB}(D_x) = D_x/256 \times 5 \text{ et } V_{WA} = 5 - V_{WB}.$$

Si vous désirez générer une intensité convenable, il suffit d'ajouter un amplificateur opérationnel câblé en suiveur de tension (tableau 5.19).

Code (Dec)	$V_{WB} (\text{V})$	$V_{WA} (\text{V})$
255	5	0
128	2,5	2,5
1	0,02	4,98
0	0	5

Tableau 5.19.

■ Programme PICBASIC : « 4sa.bas »

```
'DECLARATION DES CONSTANTES
'-----
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16

'DECLARATION DES VARIABLES
'-----
DIM Tampon(10) AS BYTE
DIM SMS(13) AS BYTE
DIM i AS BYTE
DIM n AS BYTE
DIM D(4) AS BYTE

'INITIALISATION DES 4 CIRCUITS AD8400
'-----
```

Au départ du programme il faut que les sorties CS0 à CS3 soient à l'état haut afin que les circuits AD8400 ignorent les éventuelles informations circulant sur les lignes SDI et CLK. L'instruction BYTEOUT permet de sortir la valeur binaire de la donnée (val) sur 8 sorties du PicBasic. Chaque sortie est l'image de chaque bit de la valeur binaire donnée. Dans notre cas les entrées CS sont reliées sur les sorties I/O8 à I/O11, il s'agit donc des 4 bits du bloc 1. Le LSB correspond à la broche I/O8, le MSB à la broche I/O15. Les bits 0 à 3 sont donc positionnés à l'état logique haut, l'état des autres bits n'a aucune importance (ici positionnés à zéro). Une boucle FOR/NEXT permet la mise à zéro de la variable tableau D qui contient l'image de la valeur de consigne pour chacun des 4 potentiomètres. La valeur de consigne zéro est alors envoyée simultanément aux 4 potentiomètres.

```
BYTEOUT 1,&b00001111
FOR i=0 TO 3
  D(i)=0
NEXT i

BYTEOUT 1,&b00000000
SHIFTOUT 0,1,1,0,10
BYTEOUT 1,&b00001111

'TEST LIAISON SERIE
'-----
```

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit : AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte. Les caractères « AT » suivi du caractère <CR>=13_{dec}

sont envoyés par la commande SEROUT. L'instruction SERIN permet d'attendre l'éventuelle réponse « OK » pendant 2 000 ms (soit 2 s). Si les caractères OK sont réceptionnés dans le temps donné, le caractère suivant soit <CR> est placé dans la variable i. Dans le cas contraire le programme saute à la ligne repérée par l'étiquette TEST i est alors vide, un « bip » est émis par le buzzer. Il suffit de tester le contenu de i pour savoir si la liaison est établie.

```
TEST: BEEP 4
SEROUT TXD,BDS,0,1,[ "AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST
```

```
'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500
```

```
'CODE PIN
'-----
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable.

```
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500
```

```
'INITIALISATION DU ME
'-----
```

Le ME est configuré en mode TEXT par la commande « AT+CMGF=1 ». La commande « AT+CNMI=1,1 » indique au ME que chaque nouveau SMS reçu doit être signalé au TE.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGF=1",13]
DELAY 500
SEROUT TXD,BDS,0,1,[ "AT+CNMI=1,1",13]
DELAY 500
```

```
'INITIALISATION DES VARIABLES
'-----
```

```
DEBUT:FOR i=0 TO 9
    Tampon(i)=0
NEXT i
FOR i=0 TO 12
    SMS(i)=0
```

NEXT i

'ATTENTE RECEPTION SMS

'-----

Désormais le µC scrute l'entrée RXD dans l'attente des caractères « TI ». Dès leur réception les 10 caractères suivants sont placés dans la variable Tampon. Une série de 11 bips signale l'arrivée du SMS.

```
ATT: SERIN RXD,BDS,0,10000,ATT,[WAIT("TI"),Tampon(0)~10]
FOR i=0 TO 10
    BEEP 4
NEXT i
```

'LECTURE DU SMS RECU

'-----

Le TE configure le ME pour que la lecture soit faite dans la mémoire définie par Tampon(3) et Tampon(4). La lecture du SMS est provoquée par la commande « AT+CMGR=<index> ». Dès la réception des caractères « !! » les 13 caractères suivants sont placés dans la variable SMS.

```
SEROUT TXD,BDS,0,1,["AT+CPMS=",34,Tampon(3),Tampon(4),
34,13]
DELAY 500
SEROUT TXD,BDS,0,1,["AT+CMGR="]
FOR i=7 TO 9
    IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT
        TXD,BDS,0,1,[Tampon(i)]
    NEXT i
    SEROUT TXD,BDS,0,1,[13]
    SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)~13]
SUITE: IF SMS(0)=0 THEN GOTO RAZ
```

En l'état actuel du programme, si l'on considère que le SMS envoyé est de la forme « !!S1,127 », la variable tableau SMS doit contenir ce qui est indiqué **tableau 5.20**.

Tableau 5.20.

SMS(0)	SMS(1)	SMS(2)	SMS(3)	SMS(4)	SMS(5)
S	1	,	1	2	7
83 _{dec}	49 _{dec}	44 _{dec}	49 _{dec}	50 _{dec}	55 _{dec}

SMS(1) contient le numéro de la sortie, compris entre 1 et 4.

SMS(3), SMS(4) et SMS(5) contiennent la valeur comprise entre 0 et 255 que doit prendre la sortie concernée. Cette valeur ne peut

pas être utilisée directement, car elle est codée en ASCII alors que le circuit AD8400 réclame une valeur numérique. Malheureusement il n'existe aucune instruction en langage PicBasic pour convertir une valeur de type texte en donnée de type BYTE. Attention il ne suffit pas de multiplier SMS(3) par 100, SMS(4) par 10, SMS(5) par 1 et d'additionner ces trois valeurs pour obtenir la valeur souhaitée. Voici la preuve : $49 \times 100 + 50 \times 10 + 55 \times 1 = 5455$.

```
'CONVERSION ASCII -> DECIMAL
'oooooooooooooooooooooo
```

En fait, avant de faire ce calcul il faut retrancher 48 à chaque donnée.

- $SMS(3) - 48 = 49 - 48 = 1$
- $SMS(4) - 48 = 50 - 48 = 2$
- $SMS(5) - 48 = 55 - 48 = 7$

Finalement en reprenant le calcul précédent, on obtient bien la valeur souhaitée : $1 \times 100 + 2 \times 10 + 7 \times 1 = 127$.

La formule générale est donc :

$$D = (SMS(3) - 48) \times 100 + (SMS(4) - 48) \times 10 + (SMS(5) - 48) \times 1$$

Traduit en langage PicBasic :

```
IF SMS(1)<=52 AND SMS(1)>=49 THEN

    SMS(3)=SMS(3)-48
    SMS(3)=SMS(3)x100
    SMS(4)=SMS(4)-48
    SMS(4)=SMS(4)x10
    SMS(5)=SMS(5)-48

    SMS(1)=SMS(1)-49
    D(SMS(1))=SMS(3)+SMS(4)+SMS(5)
```

Attention la valeur de consigne doit être codée sur 3 chiffres, veillez à compléter par des zéros si nécessaire. Par exemple : !!S1,009.

```
'GESTION DES SORTIES
'ooooooooooooooo
```

La variable n (soit SMS(1)) contient le numéro de la sortie en cours compris entre 0 et 3. Il suffit d'ajouter 8 à SMS(1) pour obtenir la sortie du PicBasic qui sélectionne le circuit AD8400. L'instruction SHIFTOUT permet d'envoyer la donnée D(n) au potentiomètre numérique. Cette instruction génère un signal d'horloge de synchronisation sur la sortie I/O0, tout en venant

écrire sérielement les données présentes sur l'entrée I/O1. Le paramètre suivant définit le mode d'écriture, placé ici à 1, il indique que le MSB est prioritaire. Le dernier paramètre positionné à 10, indique le nombre de bits de la donnée D(n), 2 bits pour l'adresse (toujours à 0) et 8 bits pour la donnée (comprise entre 0 et 255).

```
n=SMS(1)
SMS(1)=SMS(1)+8
OUT SMS(1),0
SHIFTOUT 0,1,1,D(n),10
OUT SMS(1),1
END IF
```

'ENVOI D'UN SMS CONTENANT LA VALEUR POUR CHAQUE SORTIE

Si la variable SMS(3) ne contient pas un chiffre compris entre 1 et 4 mais un point d'interrogation « ? », ceci dans le cas où le SMS envoyé est de la forme « !!S?,0xxxxxxxx », le montage doit rédiger et envoyer un SMS contenant les 4 valeurs de consigne. Si la commande est suivie d'une virgule et d'un numéro du téléphone, la boucle FOR/NEXT permet de balayer les 10 variables SMS(3) à SMS(12) pour reconstituer le numéro. Si aucun numéro n'est spécifié, ce qui est vrai si SMS(2)<>"", le message est envoyé au numéro par défaut indiqué en dur dans le programme.

```
n=0
IF SMS(1)="?" THEN
  IF SMS(2)=",," THEN
    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
    FOR i=3 TO 12
      SEROUT TXD,BDS,0,1,[SMS(i)]
    NEXT i
    SEROUT TXD,BDS,0,1,[34,13]
  ELSE
    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34,"06xxxxxxxx",34,13]
  END IF
  DELAY 1000
  SEROUT TXD,BDS,0,1,[ "ETAT DES SORTIES : "]
  FOR i=0 TO 3
    n=i+49
    SEROUT TXD,BDS,0,1,[ "$",n,"=",DEC(D(i),3,0),32]
  NEXT i
  SEROUT TXD,BDS,0,1,[26]
  DELAY 5000
END IF
```

'EFFACE LE SMS EN MEMOIRE

Pour terminer, le SMS est systématiquement effacé à l'aide de la commande AT+CMGD, pour éviter une saturation de la mémoire utilisée. Du fait chaque SMS reçu aura le même index.

```
RAZ: SEROUT TXD,BDS,0,1,[ "AT+CMGD=" ]
FOR i=7 TO 9
  IF Tampon(i)>=48 AND Tampon(i)<=57 THEN SEROUT
    TXD,BDS,0,1,[Tampon(i)]
  NEXT i
SEROUT TXD,BDS,0,1,[13]
DELAY 1000
GOTO DEBUT
```

Résumé des points importants

Tableau 5.21.

4 SORTIES ANALOGIQUES	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Parties du programme PicBasic à modifier	
Commande SMS reçue	Action du montage
!!Sx,data	La valeur de consigne data exprimée en décimal, <i>toujours codée sur 3 chiffres</i> , (000 ≥ data > 255) est envoyée sur la sortie pointée par x (4 ≥ x ≥ 1)
!!S?	La valeur de consigne associée pour chaque sortie est envoyée par SMS au numéro spécifié en dur dans le programme
!!S?,06xxxxxxxx	La valeur de consigne associée pour chaque sortie est envoyée par SMS au numéro indiqué

5.3 TÉLÉMESURES PAR GSM

4 entrées logiques

Voici une carte comportant 4 entrées logiques tout ou rien. Sur demande de l'utilisateur l'état logique des entrées est envoyé par SMS. Il est possible de programmer la carte pour qu'un envoi se déclenche sur un état logique précis des 4 entrées. On peut envisager d'utiliser ce montage comme système d'alarme protégeant 4 zones.

Schéma électrique

Voir Figure 5.22.

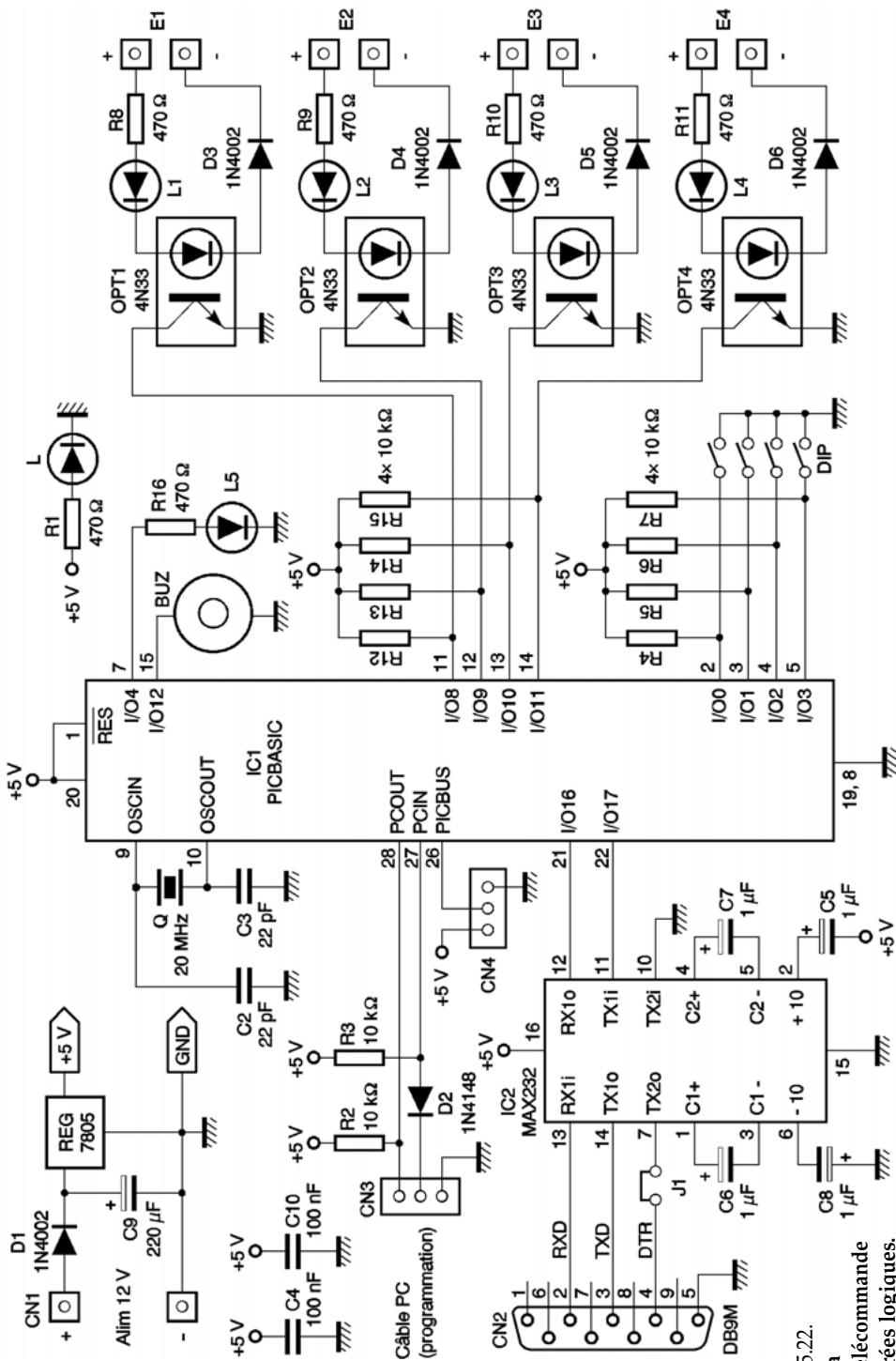


Figure 5.22.
Schéma
de la télécommande
à 4 entrées logiques.

Les lignes I/O8 à I/O11 du PicBasic sont utilisées comme des entrées logiques. Pour une protection efficace du PicBasic les 4 entrées sont isolées électriquement des tensions externes qui lui seront appliquées. Le composant chargé de cet isolement est un optocoupleur (ou photocoupleur). Comme son nom le laisse supposer le transfert de l'information binaire se fait optiquement ; un tel circuit se compose de deux parties distinctes : la première est constituée d'une diode infrarouge qui va venir mettre en conduction le phototransistor contenu dans la deuxième partie. Le 4N33 choisi ici possède un isolement électrique de 2 500 V. En théorie cela signifie qu'il faudrait appliquer une tension d'au moins 2 500 V en amont du circuit pour arriver à endommager la partie située en aval. Dans notre cadre d'utilisation on considère que les tensions « normales » appliquées sur les entrées seront comprises entre 0 et + 5 V. Sachant qu'une intensité de 10 mA traversant la diode suffit à saturer le phototransistor, les résistances R8 à R11 ont une valeur de $470\ \Omega$. Une diode externe au boîtier est utilisée pour signaler visuellement à l'utilisateur l'état de chaque entrée. On notera la présence des diodes 1N4002, D3 à D6, qui protégeront les optocoupleurs d'éventuelles tensions inverses importantes. Lorsque la tension d'entrée est nulle, voir négative, la diode interne à l'optocoupleur est éteinte, par conséquent le phototransistor est bloqué, sur l'entrée correspondante du PicBasic on obtient une tension de + 5 V du fait de la présence de la résistance de rappel (R12 à R15) au + 5 V du montage. Lorsque la tension d'entrée est égale à + 5 V, la diode interne à l'optocoupleur est allumée et vient saturer le phototransistor, on obtient par conséquent une tension d'environ 0,6 V considérée comme un état logique bas par le PicBasic. On remarque que l'état logique lu par le PicBasic est inversé par rapport à celui des entrées de la carte, il faudra en tenir compte dans le programme. Une barrette de 4 mini-interrupteurs est connectée aux entrées I/O0 à I/O3 du PicBasic. Les résistances de rappel R4 à R7 imposent une tension de + 5 V aux entrées lorsque les interrupteurs sont ouverts. Lorsque l'état des 4 interrupteurs est égal à l'état des 4 entrées E1 à E4, un SMS est envoyé au destinataire de votre choix. La Led L5 signale l'envoi du SMS (**Figures 5.23 et 5.24**).

Programme PICBASIC : « 4el.bas »

Comme le PicBasic doit à la fois traiter l'arrivée d'un éventuel SMS, et envoyer un SMS dans le cas où l'état logique des interrupteurs est identique aux entrées, nous allons utiliser la méthode de programmation mise en œuvre pour le montage « 4 sorties sur triacs ». Comme le PicBasic ne possède pas d'interruption programme lors de l'arrivée d'une donnée sur son entrée série, il est nécessaire qu'il scrute en permanence la ligne RxD

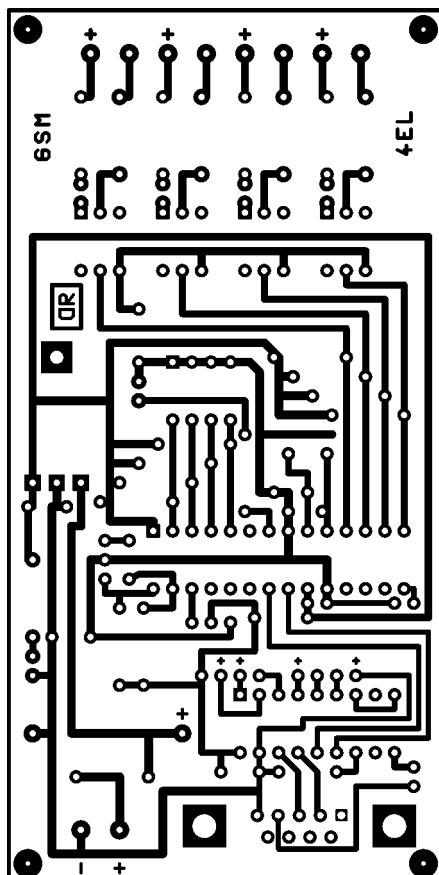


Figure 5.23.
Circuit imprimé.

dans l'attente du signal envoyé par le ME concernant l'arrivée d'un SMS, ce qui n'est pas possible dans notre application. Périodiquement, c'est le TE qui va consulter la mémoire du ME pour savoir si un nouveau SMS y est stocké. Entre deux consultations le PicBasic vérifiera l'égalité entre les entrées et les interrupteurs, si celle-ci est avérée, un SMS d'alerte sera envoyé.

```
'DECLARATION DES CONSTANTES
'-----
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16
```

```
'DECLARATION DES VARIABLES
'-----
```

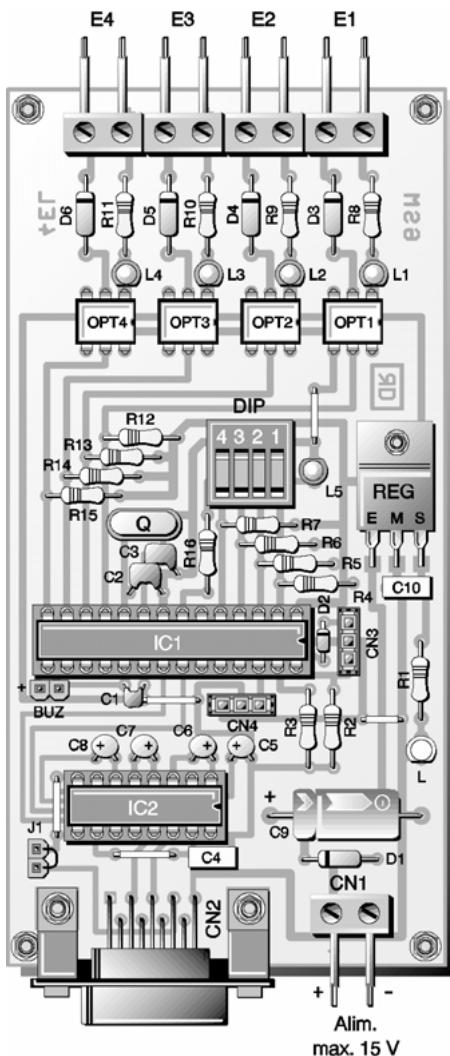


Figure 5.24.
Implantation
des composants.

Liste des composants

R1, R8 à R11, R16 : 470 Ω
 R2 à R7, R12 à R15 : 10 k Ω
 C1 : 100 nF (pas de 2,54 mm)
 C2, C3 : 22 pF / céramique
 C4, C10 : 100 nF / LCC jaune
 C5, C6, C7, C8 : 1 μ F / tantale / 15 V
 C9 : 220 μ F / électrolytique / 15 V
 D1, D3 à D6 : diode 1N4002
 D2 : diode 1N4148
 L, L1 à L5 : Led standard
 Q : quartz 20 MHz
 REG : régulateur 7805
 J1 : barrette HE10 2 contacts + cavalier

BUZ : buzzer piezzo (sans électronique intégrée)
 CN1 : bornier à vis 2 plots
 CN2 : connecteur DB9 mâle pour CI / coudé à 90°
 CN3 : connecteur pour câble de programmation (LEXTRONIC)
 CN4 : connecteur pour écran LCD (LEXTRONIC) (facultatif)
 IC1 : PICBASIC PB-3B (LEXTRONIC) + support DIL 28 broches (étroit)
 IC2 : MAX232 + support DIL 16 broches
 OPT1 à OPT4 : optocoupleur 4N33
 DIP : interrupteur mini dip 4 contacts

```
DIM index(3) AS BYTE
DIM SMS(13) AS BYTE
DIM i AS BYTE
DIM j AS INTEGER
DIM n AS BYTE
DIM num AS BYTE

' TEST LIAISON SERIE
'-----
i=0
TEST: BEEP 12
SEROUT TXD,BDS,0,1,["AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,["AT+CSGS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable.

```
SEROUT TXD,BDS,0,1,["AT+CPIN=",34,"7208",34,13]
DELAY 500
```

Comme nous n'avons pas besoin que le ME avertisse le TE de l'arrivée d'un nouveau SMS, la commande « AT+CNMI=1,1 » est retirée.

```
'INITIALISATION DU ME EN MODE TEXT
'-----
SEROUT TXD,BDS,0,1,["AT+CMGF=1",13]
DELAY 500

'SELECTION MEMOIRE ET INDEX POUR LECTURE SMS
'-----
```

Comme nous l'avons dit plus haut, la mémoire de stockage et l'index sont figés. Il faut donc déterminer ces deux paramètres à l'avance. Concernant la mémoire <mem1>, on considère que les SMS envoyés par les particuliers ne possèdent pas de classe. Cela

signifie que le mobile qui reçoit ce genre de SMS le stocke dans la mémoire ME. Les autres types de mémoire sont surtout utilisés par les opérateurs. Si votre téléphone le permet vous pouvez utiliser le paramètre MT qui permet aux commandes de lecture de SMS de travailler avec toutes les mémoires. Ici nous avons choisi la mémoire du téléphone d'où « AT+CPMS= ME ». De même, la variable index doit être initialisée avec l'index que portera le prochain SMS réceptionné. Il correspond au premier emplacement de libre dans la mémoire sélectionnée. Pour le déterminer, vous pouvez utiliser le logiciel « convertSMS2 », après avoir sélectionné la mémoire soit « ME » dans notre cas, cliquez sur le bouton « Tous » il suffit de relever l'index du premier emplacement de libre. L'index par défaut utilisé ici est fixé à 900. Si vous avez un index codé sur un ou deux chiffres, il suffit de mettre les variables non utilisées à nul. Par exemple si index = 1, il faudra modifier le programme comme ceci :

```
index(0)=""":index(1)=""":index(2)="1".
```

```
SEROUT TXD,BDS,0,1,[ "AT+CPMS=",34,"ME",34,13]
DELAY 500
index(0)="9":index(1)="0":index(2)="0"

'INITIALISATION N° TÉLÉPHONE UTILISÉ PAR DÉFAUT POUR L'ENVOI DES
SMS
'-----
```

Le numéro de téléphone utilisé par défaut pour envoyer des SMS est initialisé dans la mémoire eeprom du PicBasic. Pour ne pas interférer avec la partie programme, le stockage se fait dans les 10 derniers emplacements de la mémoire de FF6_{hex} à FFF_{hex}. Cette mémorisation ne se réalise qu'une seule fois car le programme teste avant si l'adresse FF6_{hex} est vide (notez qu'un emplacement vide contient la donnée FF_{hex}).

```
IF EEREAD(&HFF6)=&HFF THEN
EEWRITE &HFF6,"0"
EEWRITE &HFF7,"6"
EEWRITE &HFF8,"x"
EEWRITE &HFF9,"x"
EEWRITE &HFFA,"x"
EEWRITE &HFFB,"x"
EEWRITE &HFFC,"x"
EEWRITE &HFFD,"x"
EEWRITE &HFFE,"x"
EEWRITE &HFFF,"x"
END IF

'INITIALISATION DES VARIABLES
'-----
```

```
        OUT4,0
DEBUT: FOR i=0 TO 12
      SMS(i)=0
      NEXT i

'COMPARAISON INTERRUPEURS / ENTREES
'-----
```

En premier lieu, l'état logique des 4 interrupteurs est lu, puis comparé avec l'état des 4 entrées de la carte, en cas d'égalité le sous-programme ALERT est exécuté.

```
IF IN(0)=IN(8) AND IN(1)=IN(9) THEN
  IF IN(2)=IN(10) AND IN(3)=IN(11) THEN GOSUB ALERT
END IF
```

```
'REGARDE SI RECEPTION D'UN SMS
'-----
```

Le programme, grâce à la commande « AT+CMGR », regarde si l'emplacement mémoire indiqué par l'index contient un SMS. Si l'emplacement est vide le programme saute à l'étiquette RAZ. Dans le cas contraire si le texte contient les caractères « !! », les 13 caractères suivants, qui définissent la commande, sont stockés dans la variable SMS, sinon le programme saute à l'étiquette RAZ.

```
SEROUT TXD,BDS,0,1,["AT+CMGR="]
FOR i=0 to 2
  IF index(i)>=48 AND index(i)<=57 THEN SEROUT
    TXD,BDS,0,1,[index(i)]
  NEXT i
  SEROUT TXD,BDS,0,1,[13]
  SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)~13]
SUITE: IF SMS(0)=0 THEN GOTO RAZ
  FOR i=0 TO 10
    BEEP 12
  NEXT i
```

```
'GESTION DES SMS RECEPTIONNES
'-----
```

Dans un premier temps, seules les deux premières lettres constituant la commande reçue par SMS sont vérifiées par le programme. Si les lettres « E? » sont reconnues, le sous-programme ETAT est appelé. Si les lettres « N, » sont détectées, le sous-programme MAJNUM est appelé. Si ce sont les lettres « RA », la sortie I/O4 qui signale et mémorise le fait qu'un message d'alerte a été envoyé, est remise à zéro. La Led L5 est alors éteinte.

```
IF SMS(0)="E" AND SMS(1)="?" THEN GOSUB ETAT
IF SMS(0)="N" AND SMS(1)="," THEN GOSUB MAJNUM
```

```
IF SMS(0)="R" AND SMS(1)="A" THEN OUT4,0
GOTO RAZ
```

'MESSAGE D'ALERTE EN CAS D'EGALITE

'-----

Si la sortie I/O4 est à zéro, le sous-programme ENV qui compose la première partie du SMS est appelé. Ensuite vient se greffer à la fin du message, le texte «=> ALERTE ». Le caractère 26_{dec} soit <EOF> déclenche l'envoi du SMS. La sortie I/O4 est positionnée à 1 pour éviter que d'autres SMS d'alerte soient envoyés. L'utilisateur devra envoyer un SMS avec la commande !!RA pour positionner la sortie I/O4 à zéro.

```
ALERT: IF OUTSTAT(4)=0 THEN
    GOSUB ENV
    SEROUT TXD,BDS,0,1,[ "> ALERTE",26]
    OUT4,1
    DELAY 5000
END IF
RETURN
```

'ENVOI ETAT DES ENTREES

'-----

Appel du sous-programme ENV pour la composition du SMS spécifiant l'état des 4 entrées. Le caractère 26_{dec} soit <EOF> déclenche l'envoi du SMS.

```
ETAT: GOSUB ENV
SEROUT TXD,BDS,0,1,[26]
DELAY 5000
RETURN
```

'Modification du numéro utilisé pour l'envoi des SMS

'-----

Le numéro contenu dans la commande !!N,06xxxxxxxx est sauvegardé dans la mémoire eeprom du PicBasic aux adresses FF6_{hex} à FFF_{hex}. C'est ce numéro qui sera utilisé pour l'expédition des SMS.

MAJNUM:

```
i=2
FOR j=&HFF6 TO &FFFF
    EEWRITE j,SMS(i)
    i=i+1
NEXT j
```

'ENVOI D'UN SMS CONTENANT L'ETAT DES 4 ENTREES

'-----

Si la variable SMS(2) contient une virgule, c'est qu'il s'agit de la commande !!E?,06xxxxxxxx, le programme envoie alors le SMS au numéro indiqué par les variables SMS(3) à SMS(12). Pour tous les autres cas, le numéro utilisé est celui situé dans la mémoire eeprom du PicBasic, aux adresses FF6_{hex} à FFF_{hex}. Le message expédié sur le réseau GSM contient l'état logique en cours des 4 entrées de la carte.

```
ENV:   SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
       IF SMS(2)="," THEN
           FOR i=3 TO 12
               SEROUT TXD,BDS,0,1,[SMS(i)]
           NEXT i
       ELSE
           FOR j=&HFF6 TO &HFFF
               num=EEREAD(j)
               SEROUT TXD,BDS,0,1,[num]
           NEXT j
       END IF
       SEROUT TXD,BDS,0,1,[34,13]
       DELAY 1000
       SEROUT TXD,BDS,0,1,[ "ETAT DES ENTREES : "]
       FOR i=8 TO 11
           n=i+41
           IF IN(i)=0 THEN
               SEROUT TXD,BDS,0,1,[ "E",n,"=ON "]
           ELSE
               SEROUT TXD,BDS,0,1,[ "E",n,"=OFF "]
           END IF
       NEXT i
       RETURN

'EFFECTUE LE NETTOYAGE DU MEMOIRE
-----
```

Cette partie du programme permet de systématiquement effacer le SMS en mémoire, ainsi le prochain SMS réceptionné aura toujours le même index. Ceci évite de prévoir une incrémentation de la variable index et surtout de saturer la mémoire utilisée. Notez que cette partie de programme est appelée même si aucun SMS n'est à effacer, le ME répond par un message d'erreur qui est ignoré par le programme.

```
RAZ:   SEROUT TXD,BDS,0,1,[ "AT+CMGD=" ]
       FOR i=0 TO 2
           IF index(i)>=48 AND index(i)<=57 THEN SEROUT
               TXD,BDS,0,1,[index(i)]
           NEXT i
       SEROUT TXD,BDS,0,1,[13]
       DELAY 1000
       GOTO DEBUT
```

Test du montage

Dans un premier temps il est prudent de tester le montage à l'aide d'un PC avant d'y relier un téléphone. Réalisez un câble RS232 à l'aide d'un cordon comportant 3 conducteurs et de 2 connecteurs DB9 femelles à câbler.

Reliez le montage au port série du PC. Ouvrez une session du logiciel Hyper Terminal, vous pouvez reprendre le fichier « Interfaces GSM.lnk » présenté dans le chapitre 4. Rappelons que la vitesse de transmission est de 9 600 bauds, 8 bits de données et pas de contrôle de flux. Alimentez le montage, aussitôt les caractères « AT » doivent apparaître sur l'écran du PC. Répondez dans les 2 secondes qui suivent en tapant les caractères OK, suivis d'un retour chariot. La commande d'initialisation du mode TEXT AT+CMGF=1 doit s'afficher à l'écran ; répondez aussi par OK et un retour chariot. Vous devez voir apparaître la commande AT+CMGR=900 (en admettant que index est fixé à 900), répondez dans les 5 secondes qui suivent en tapant la commande « !!E ? ». Vous devez voir apparaître à l'écran les instructions permettant l'envoi d'un SMS contenant l'état logique des 4 entrées. Appliquez maintenant un état logique sur les entrées correspondant à celui des interrupteurs. Aussitôt la Led L5 doit s'allumer. Vous devez voir apparaître à l'écran les instructions permettant l'envoi d'un SMS contenant l'état logique des 4 entrées suivi cette fois de la phrase « => ALERTE ». À la prochaine apparition de la commande AT+CMGR=900 vous pouvez tester la commande « !!RA » qui doit provoquer l'extinction de la Led L5.

Si tout fonctionne correctement vous pouvez relier votre téléphone ou terminal GSM au montage.

Résumé des points importants

Voir Tableau 5.22.

4 entrées analogiques

Voici une carte capable de convertir 4 tensions analogiques en 4 données numériques codées sur 8 bits et d'envoyer le résultat par SMS sur demande de l'utilisateur. La carte envoie également un SMS d'alerte lorsqu'une tension dépasse un seuil préalablement programmé.

Schéma électrique

Voir Figure 5.25.

Le PicBasic 3B possède 5 entrées analogiques disponibles sur les broches AD0 à AD4. Dans le cadre de notre application nous nous limiterons à l'acquisition de 4 tensions analogiques sur les

Tableau 5.22.

4 ENTRÉES LOGIQUES	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce 4 mini-interrupteurs, lorsque l'état logique est égal à celui des 4 entrées un SMS d'alerte est envoyé	
Parties du programme PicBasic à modifier	
<ul style="list-style-type: none"> • Code PIN (7208 par défaut) • Mémoire lecture SMS <mem1> (ME par défaut) • Index du prochain SMS reçu (900 par défaut) • Numéro de téléphone utilisé par défaut pour l'envoi des SMS 	
Commande SMS reçue	Action du montage
!!E?	Un SMS contenant l'état logique des 4 sorties est envoyé au numéro présent dans la mémoire eeprom du PicBasic
!!N,06xxxxxxxx	Modifie dans la mémoire eeprom du PicBasic le numéro utilisé pour l'envoi des SMS
!!E?,06xxxxxxxx	Un SMS contenant l'état logique des 4 sorties est envoyé au numéro indiqué
!!RA	Autorise le montage à envoyer d'autres SMS d'alerte

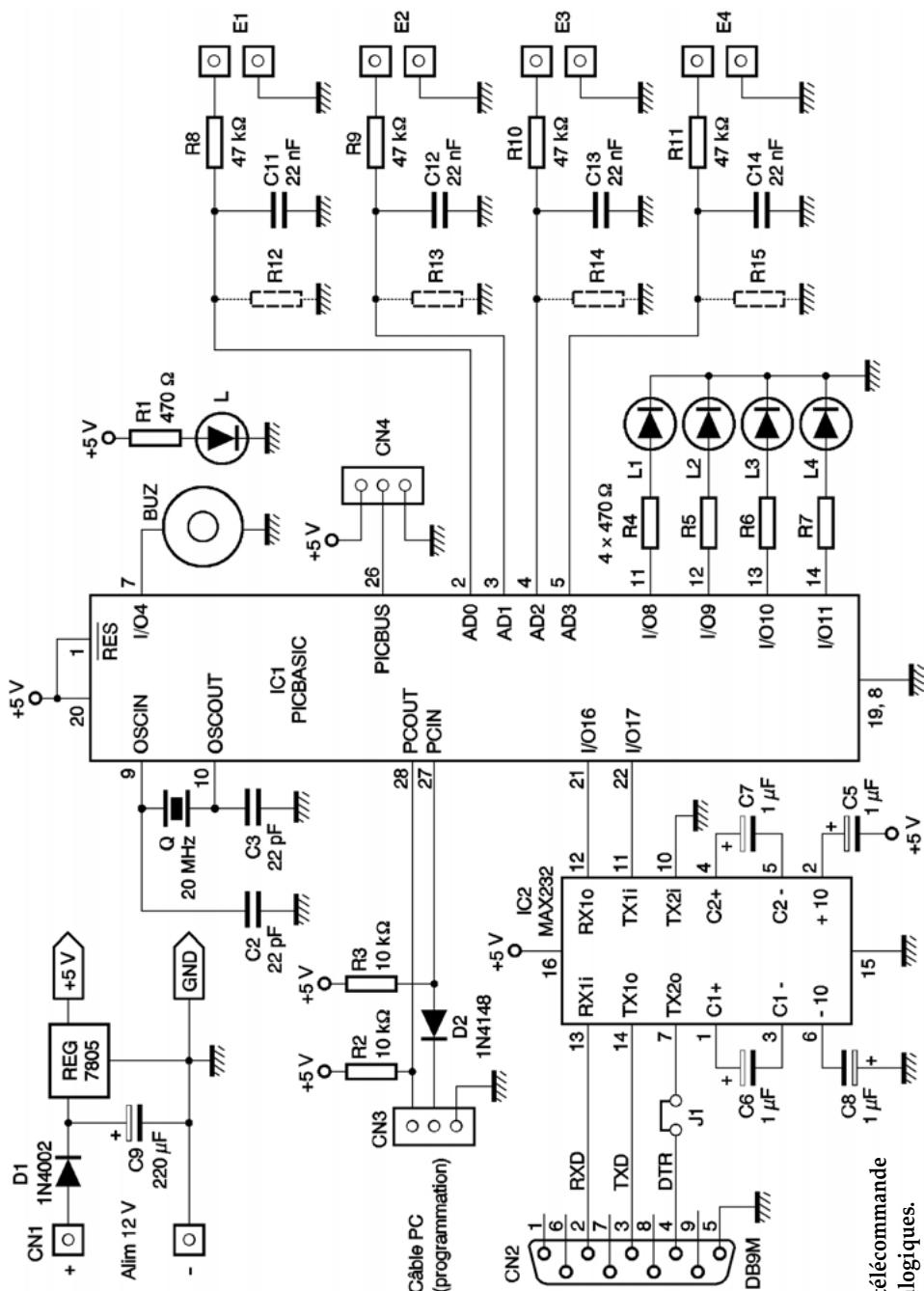
entrées AD0 (broche2) à AD3 (broche5). La valeur de la tension à lire doit être comprise entre 0 et + 5 V. Il est impératif que la tension ne dépasse pas la barre des + 5 V, sous peine d'endommager le PicBasic. Pour l'acquisition de tensions supérieures à + 5 V il est prévu sur la carte des emplacements pour ajouter les résistances R12 à R15, chaque résistance associée à celle existante forme un pont diviseur de tension.

À titre d'exemple étudions le cas de l'entrée E1 (**figure 5.26**). Le raisonnement est bien entendu similaire pour les trois autres entrées de la carte. La tension AD0 appliquée au CAN du PicBasic se calcule ainsi : $AD0 = [R12/(R12 + R8)] \times E1$. Pour chaque valeur de E1 en entrée on souhaite connaître la résistance R12 à utiliser, il nous faut donc une fonction de la forme $R12 = f(E1)$. On sait que lorsque E1 est au maximum on doit toujours avoir $AD0 = + 5$ V. On en déduit la formule suivante :

$$R12 = (5 \times R8) / (E1 - 5).$$

Par exemple si l'on souhaite mesurer une tension comprise entre 0 et + 10 V, on aura $R12 = (5 \times R8) / (10 - 5)$ d'où $R12 = R8$. Pour une tension comprise entre 0 et + 15 V on aura $R12 = R8 / 10$.

Notez la présence d'un condensateur de 22 nF qui élimine les variations brusques de la tension à mesurer (filtre passe bas).



© DUNOD - La photocopie non autorisée est un délit.

Figure 5.25.
Schéma de la télécommande
à 4 entrées analogiques.

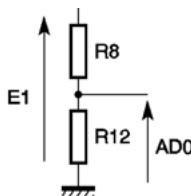


Figure 5.26.
Entrée E1.

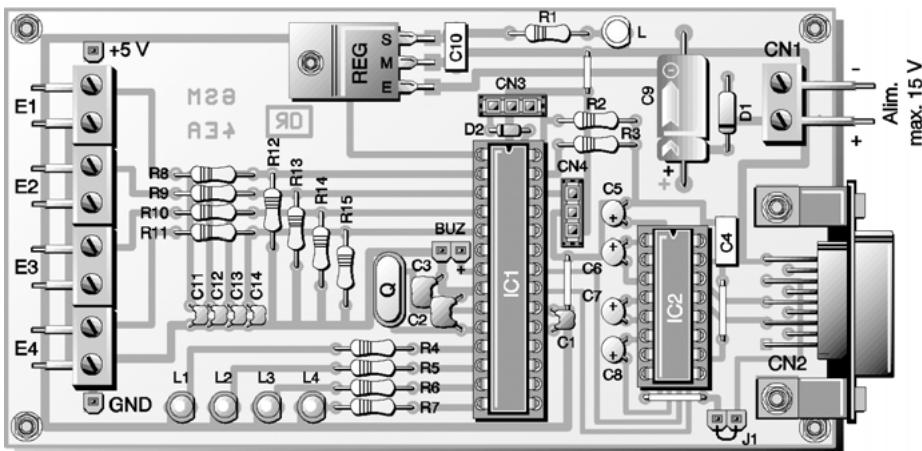
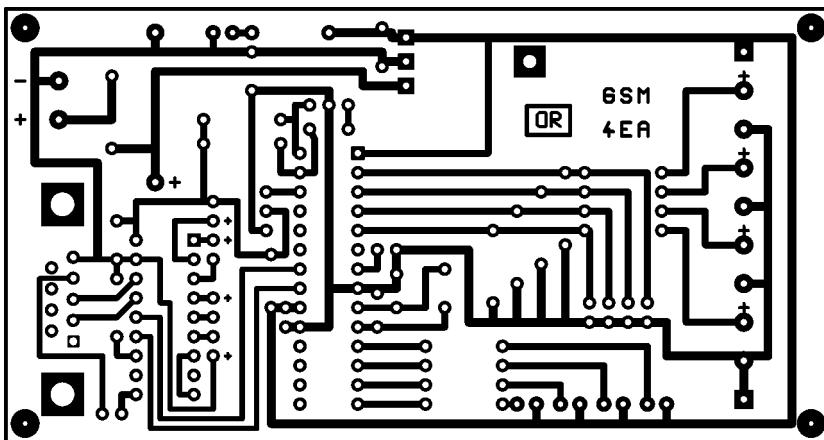
Avec la version du PicBasic que nous utilisons, la résolution du convertisseur est de 10 bits ; cela signifie que le PicBasic convertit une tension en un nombre binaire composé de 10 bits. La précision de la mesure est donc égale à $5/2^{10} = 0,005$ V. Rappelons la relation qui permet à partir de la valeur binaire de calculer la tension : $V = (D \times 5)/2^{10}$, V est la tension mesurée exprimée en volts, D est la donnée exprimée en décimal calculée par le PicBasic. Par exemple si $D = 512$ cela signifie que la tension mesurée est égale à 2,5 V. L'instruction basic qui permet de réaliser une conversion est ADIN(port), avec port compris entre 0 et 3.

Comme nous l'avons dit dans l'introduction, la carte doit envoyer un SMS dès lors que la tension mesurée dépasse un seuil préalablement programmé par l'utilisateur. Dans un premier temps pour signaler le dépassement du seuil sur chacune des entrées, nous utilisons 4 Led, associées bien entendu à des résistances de limitation, sur les broches I/O8 à I/O11 utilisées en sorties.

Programme PICBASIC : « 4ea.bas »

Comme le PicBasic doit à la fois traiter l'arrivée d'un éventuel SMS, et envoyer un SMS dans le cas où une tension dépasse le seuil défini par l'utilisateur, nous allons utiliser la méthode de programmation mise en œuvre pour le montage « 4 sorties sur triacs ». Comme le PicBasic ne possède pas d'interruption programme lors de l'arrivée d'une donnée sur son entrée série, il est nécessaire qu'il scrute en permanence la ligne RxD dans l'attente du signal envoyé par le ME concernant l'arrivée d'un SMS, ce qui n'est pas possible dans notre application. Périodiquement, c'est le TE qui va consulter la mémoire du ME pour savoir si un nouveau SMS y est stocké. Entre deux consultations le PicBasic vérifiera que chacune des tensions mesurées est inférieure au seuil correspondant.

Le programme ci-après est prévu pour l'acquisition de tensions comprises entre 0 et + 5 V. Il est donc inutile d'implanter les résistances R12 à R15 sur la carte. Toutefois si les résistances en question sont en place il suffirait de multiplier le résultat de la conversion par un coefficient. Par exemple si vous mesurez une tension comprise entre 0 et + 10 V il faut multiplier par 2 le résultat obtenu par le CAN du PicBasic pour retrouver la tension d'entrée.



Liste des composants

R1, R4 à R7 : 470 Ω

R2, R3 : 10 k Ω

R8 à R11 : 47 k Ω

R12 à R15 : résistance à prévoir si la tension à mesurer est supérieure à + 5 V

C1 : 100 nF (pas de 2,54 mm)

C2, C3 : 22 pF / céramique

C4, C10 : 100 nF / LCC jaune

C5, C6, C7, C8 : 1 μ F / tantale / 15 V

C9 : 220 μ F / électrolytique / 15 V

C11 à C14 : 22 nF / céramique

D1 : diode 1N4002

D2 : diode 1N4148

L, L1 à L4 : Led standard

Q : quartz 20 MHz

REG : régulateur 7805

Figure 5.27. (en haut)
Circuit imprimé.

Figure 5.28. (en bas)
Implantation des composants.

BUZ : buzzer piezzo (sans électronique intégrée)

J1 : barrette HE10 2 contacts + cavalier

CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle pour CI / coudé à 90°

CN3 : connecteur pour câble de programmation (LEXTRONIC)

CN4 : connecteur pour écran LCD (LEXTRONIC) (facultatif)

IC1 : PICBASIC PB-3B (LEXTRONIC) + support DIL 28 broches (étroit)

IC2 : MAX232 + support DIL 16 broches

```
'DECLARATION DES CONSTANTES
'-----
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16

'DECLARATION DES VARIABLES
'-----
DIM index(3) AS BYTE
DIM SMS(13) AS BYTE
DIM i AS BYTE
DIM j AS INTEGER
DIM n AS BYTE
DIM FLAG AS BYTE
DIM seuil(4) AS BYTE
DIM num AS BYTE

DIM D AS INTEGER
DIM V AS INTEGER
DIM V1 AS INTEGER
DIM V2 AS INTEGER
DIM V3 AS INTEGER

'TEST LIAISON SERIE
'-----
i=0
TEST: BEEP 4
SEROUT TXD,BDS,0,1,["AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500

'INITIALISATION DU ME EN MODE TEXT
'-----
SEROUT TXD,BDS,0,1,[ "AT+CMGF=1",13]
DELAY 500

'INITIALISATION DES VARIABLES
'-----
```

Comme nous l'avons dit plus haut, la mémoire de stockage et l'index sont figés. Il faut donc déterminer ces deux paramètres à l'avance. Concernant la mémoire <mem1>, on considère que les SMS envoyés par les particuliers ne possèdent pas de classe. Cela signifie que le mobile qui reçoit ce genre de SMS le stocke dans la mémoire ME. Les autres types de mémoire sont surtout utilisés

par les opérateurs. Si votre téléphone le permet vous pouvez utiliser le paramètre MT qui permet aux commandes de lecture de SMS de travailler avec toutes les mémoires. Ici nous avons choisi la mémoire du téléphone d'où « AT+CPMS= ME ». De même, la variable index doit être initialisée avec l'index que portera le prochain SMS réceptionné. Il correspond au premier emplacement de libre dans la mémoire sélectionnée. Pour le déterminer, vous pouvez utiliser le logiciel « convertSMS2 », après avoir sélectionné la mémoire soit « ME » dans notre cas, cliquez sur le bouton « Tous » il suffit de relever l'index du premier emplacement de libre. L'index par défaut utilisé ici est fixé à 900. Si vous avez un index codé sur un ou deux chiffres, il suffit de mettre les variables non utilisées à zéro. Par exemple si index = 1, il faudra modifier le programme comme ceci : index(0)="" ; index(1)="" ; index(2) = "1".

Il faut dans cette partie définir également les seuils des tensions qui déclencheront l'envoi d'un SMS. Chaque entrée possède son propre seuil exprimé en décimal. Attention comme il n'est pas possible de déclarer une variable de type tableau en INTEGER, les valeurs doivent être comprises entre 0 et 255. Voici la relation qui permet de calculer la valeur décimale codée sur 8 bits en fonction de la tension : $D = (V \times 2^8)/5$, par exemple si vous désirez qu'un SMS d'alerte soit envoyé si la tension mesurée sur l'entrée E1 dépasse 2,5 V, il vous faut initialiser la variable seuil(0) à 127.

```
SEROUT TXD,BDS,0,1,[AT+CPMS=,34,"ME",34,13]
DELAY 500
index(0)="9":index(1)="0":index(2)="0"
seuil(0)=127:seuil(1)=127:seuil(2)=127:seuil(3)=127
BYTEOUT 1,&b00001111
```

```
'INITIALISATION N° TÉLÉPHONE UTILISÉ PAR DÉFAUT POUR L'ENVOI DES
SMS
```

Le numéro de téléphone utilisé par défaut pour envoyer des SMS est initialisé dans la mémoire eeprom du PicBasic. Pour ne pas interférer avec la partie programme, le stockage se fait dans les 10 derniers emplacements de la mémoire de FF6_{hex} à FFF_{hex}. Cette mémorisation ne se réalise qu'une seule fois car le programme teste avant si l'adresse FF6_{hex} est vide (notez qu'un emplacement vide contient la donnée FF_{hex}).

```
IF EEREAD(&FF6)=&HFF THEN
EEWRITER &HFF6,"0"
EEWRITER &HFF7,"6"
EEWRITER &HFF8,"x"
```

```
EEWRITE &HFF9,"x"
EEWRITE &HFFA,"x"
EEWRITE &HFFB,"x"
EEWRITE &HFFC,"x"
EEWRITE &HFFD,"x"
EEWRITE &HFFE,"x"
EEWRITE &HFFF,"x"
END IF

'INITIALISATION DES VARIABLES (suite)
'-----
DEBUT: FLAG=0
FOR i=0 TO 12
SMS(i)=0
NEXT i

'COMPARAISON TENSIONS
'-----
```

Dans un premier temps le PicBasic regarde si au moins un des 4 seuils programmés est dépassé. Si cette condition est vérifiée et que la Led de signalement correspondante est inactive, le sous-programme ENV est appelé.

```
FOR i=0 TO 3
n=i+8
D=ADIN(i)
D=D/4
IF D>=SEUIL(i) AND OUTSTAT(n)=1 THEN FLAG=1
NEXT i
IF FLAG=1 THEN GOSUB ENV

'REGARDE SI RECEPTION D'UN SMS
'-----
```

Le programme grâce à la commande « AT+CMGR » regarde si l'emplacement mémoire indiqué par l'index contient un SMS. Si l'emplacement est vide le programme saute à l'étiquette RAZ. Dans le cas contraire si le texte contient les caractères « !! », les 13 caractères suivants, qui contiennent la commande, sont stockés dans la variable SMS, sinon le programme saute à l'étiquette RAZ.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGR=" ]
FOR i=0 to 2
IF index(i)>=48 AND index(i)<=57 THEN SEROUT
TXD,BDS,0,1,[index(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)~13]
SUITE: IF SMS(0)=0 THEN GOTO RAZ
```

```

FOR i=0 TO 10
BEEP 4
NEXT i

'GESTION DES SMS RECEPTIONNES
'-----
```

Seules les deux premières lettres constituant la commande reçue par SMS sont vérifiées par le programme. Si les lettres « E? » sont reconnues, le sous-programme ENV est appelé. Si les lettres « N, » sont reconnues, le sous-programme MAJNUM est appelé. Si ce sont les lettres « RA », les 4 Led qui mémorisent le fait qu'un message d'alerte a été envoyé sont remises à zéro.

```

IF SMS(0)="E" AND SMS(1)="?" THEN GOSUB ENV
IF SMS(0)="N" AND SMS(1)=", " THEN GOSUB MAJNUM
IF SMS(0)="R" AND SMS(1)="A" THEN BYTEOUT 1,&b00001111
GOTO RAZ
```

```
'ENVOI D'UN SMS CONTENANT L'ETAT DES 4 ENTREES
'-----
```

Si la variable SMS(2) contient une virgule, c'est qu'il s'agit de la commande !!E?,06xxxxxxxx, le programme envoie alors le SMS au numéro indiqué par les variables SMS(3) à SMS(12). Pour tous les autres cas, le numéro utilisé est celui situé dans la mémoire eeprom du PicBasic, aux adresses FF6_{hex} à FFF_{hex}. Le message expédié sur le réseau GSM indique la valeur de la tension mesurée en volts (0 à + 5 V) sur chacune des 4 entrées de la carte. Dans le cas où la tension dépasse le seuil programmé le message « => seuil atteint » est ajouté. La Led de signalisation correspondante est activée pour éviter l'envoi d'autres SMS. Rappelons que la commande « !!RA » permet de remettre à zéro les 4 Led.

```

ENV: SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
      IF SMS(2)=", " THEN
          FOR i=3 TO 12
              SEROUT TXD,BDS,0,1,[SMS(i)]
          NEXT i
      ELSE
          FOR j=&HFF6 TO &HFFF
              num=EEREAD(j)
              SEROUT TXD,BDS,0,1,[num]
          NEXT j
      END IF
      SEROUT TXD,BDS,0,1,[34,13]
      DELAY 1000
      SEROUT TXD,BDS,0,1,[ "TENSIONS MESUREES : "]
      FOR i=0 TO 3
```

```
n=i+48
D=ADIN(i)
GOSUB CONV
SEROUT TXD,BDS,0,1,["E",n,"+",V1,",",V2,V3,"v"]
n=i+8
IF D>=SEUIL(i) AND OUTSTAT(n)=1 THEN
    SEROUT TXD,BDS,0,1,["=> ALERTE "]
    OUT n,0
ELSE
    IF OUTSTAT(n)=0 THEN SEROUT TXD,BDS,0,1,["(seuil
        atteint) "]
END IF
NEXT i
SEROUT TXD,BDS,0,1,[26]
DELAY 5000
RETURN

'Conversion DECIMAL -> TENSION -> ASCII
'-----
```

Ce sous-programme permet de convertir la valeur décimale fournie par l'instruction D=ADIN(i) en trois caractères ASCII. Le résultat contenu dans la variable D est codé sur 10 bits. Pour faciliter l'écriture de notre programme nous allons travailler sur 8 bits, ce qui revient à diviser le résultat par 4. La précision de la mesure est donc égale à $5/2^8 = 0,02$ V ce qui n'est déjà pas si mal. La formule qui permet de calculer la tension correspondante est de la forme $V = (D \times 5)/2^8$. Comme le PicBasic ne peut pas travailler avec des nombres à virgule, nous allons multiplier le résultat par 100, en simplifiant, la relation devient $V = (100 \times D)/51$. On obtient alors un nombre entier compris entre 0 et 500. Pour extraire le chiffre des unités nommé V1 on divise le résultat par 100, d'où la relation $V1 = V/100$. La première décimale nommée V2 est obtenue par la formule $V2 = (V - 100 \times V1)/10$. Enfin la deuxième décimale est obtenue par la formule $V3 = V - (100 \times V1 + 10 \times V2)$, notez qu'il est nécessaire de découper cette formule en 3 sous formules pour que le PicBasic puisse effectuer le calcul de V3. Finalement on ajoute à chacun des chiffres le nombre 48_{dec} pour obtenir le caractère ASCII correspondant. Le sous-programme ENV se chargera d'intercaler une virgule entre V1 et V2 lors de la composition du SMS.

```
CONV:  D=D/4
      V=(100*D)/51
      V1=V/100
      V3=100*V1
      V2=(V-V3)/10
```

```
V3=V3+(10*V2)
V3=V - V3
V1=V1+48
V2=V2+48
V3=V3+48
RETURN
```

'MISE À JOUR DU NUMÉRO POUR L'ENVOI DES SMS

Le numéro contenu dans la commande !!N,06xxxxxxxx est sauvegardé dans la mémoire eeprom du PicBasic aux adresses FF6_{hex} à FFF_{hex}. C'est ce numéro qui sera utilisé pour l'expédition des SMS.

MAJNUM:

```
i=2
FOR j=&HFF6 TO &HFFF
EEWRITE j,SMS(i)
i=i+1
NEXT j
```

'EFFACE LE SMS EN MEMOIRE

Cette partie du programme permet de systématiquement effacer le SMS en mémoire, ainsi le prochain SMS réceptionné aura toujours le même index. Ceci évite de prévoir une incrémentation de la variable index et surtout de saturer la mémoire du téléphone. Notez que cette partie de programme est appelée même si aucun SMS n'est à effacer, le MÉ répond par un message d'erreur qui est ignoré par le programme.

```
RAZ: SEROUT TXD,BDS,0,1,[AT+CMGD="]
FOR i=0 TO 2
IF index(i)>=48 AND index(i)<=57 THEN SEROUT
TXD,BDS,0,1,[index(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
DELAY 1000
GOTO DEBUT
```

Résumé des points importants

Voir Tableau 5.23.

Thermomètre

Voici certainement le premier thermomètre GSM ! Vous pouvez à tout instant demander la température ambiante en degré Celsius que vous recevrez sous forme d'un SMS. Le montage vous avertit automatiquement lorsque la température est négative, idéal donc

Tableau 5.23.

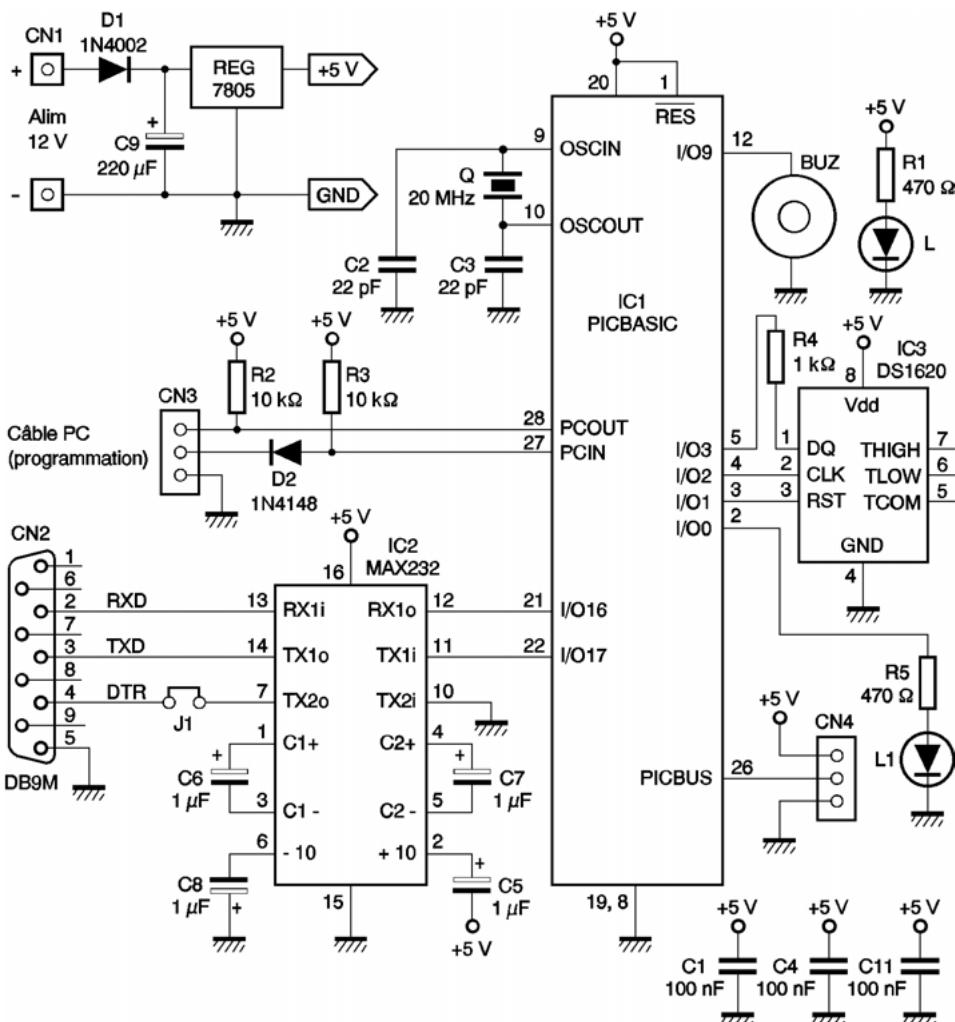
4 ENTRÉES ANALOGIQUES	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce Les résistances R12 à R15 sont à planter sur la carte si la tension appliquée aux entrées peut dépasser + 5 V	
Parties du programme PicBasic à modifier	
<ul style="list-style-type: none"> • Code PIN (7208 par défaut) • Mémoire lecture SMS <mem1> (ME par défaut) • Index du prochain SMS reçu (900 par défaut) • Numéro de téléphone utilisé par défaut pour l'envoi des SMS • Seuils de déclenchement de l'envoi d'un SMS (seuil(0), seuil(1), seuil(2), seuil(3)) 	
Commande SMS reçue	Action du montage
!!E?	Un SMS contenant les 4 tensions mesurées est envoyé au numéro spécifié dans la mémoire eeprom du PicBasic
!!N,06xxxxxxxx	Modifie dans la mémoire eeprom du PicBasic le numéro utilisé pour l'envoi des SMS
!!E?,06xxxxxxxx	Un SMS contenant les 4 tensions mesurées est envoyé au numéro spécifié
!!RA	Autorise le montage à envoyer d'autres SMS d'alerte

pour réaliser un détecteur de gel. Il est aussi possible de définir une température maximale, lorsque celle-ci est atteinte, un SMS d'avertissement vous est envoyé, idéal pour détecter un incendie.

Schéma électrique

Ce montage utilise le circuit DS1620 du constructeur Dallas. Il s'agit d'un capteur de température ambiante contenu dans un boîtier DIL 8 broches. La liaison avec le PicBasic s'effectue via un bus SPI nécessitant seulement trois lignes pour dialoguer : l'horloge CLK qui synchronise les données circulant sur DQ et RST qui permet une remise à zéro du circuit. Ces trois lignes sont respectivement reliées aux broches I/O2 (broche 4), I/O3 (broche 5) et I/O1 (broche 3) du PicBasic. La température comprise entre - 55 °C et + 125 °C avec une précision de 0,5 °C est transmise sous la forme d'un mot de 9 bits en complément à 2 sur la ligne DQ, le LSB (bit de poids le plus faible) est transmis en premier. Comme le PicBasic ne peut pas traiter des nombres à virgule, la précision de la mesure sera de 1 °C.

On remarque que le 9^e bit indique le signe, s'il est égal à 0 la température mesurée est positive, s'il est égal à 1 la température


 Figure 5.29.
 Schéma du thermomètre GSM.

Température	Donnée binaire	Donnée Hex.	Donnée Déc.
+ 125 °C	0 11111010	00FA	250
+ 25 °C	0 00110010	0032	50
0,5 °C	0 00000001	0001	1
0 °C	0 00000000	0000	0
- 0,5 °C	1 11111111	01FF	511
- 25 °C	1 11001110	01CE	462
- 55 °C	1 10010010	0192	402

 Tableau 5.24.
 Relation entre la donnée et la température.

INTERFACES GSM

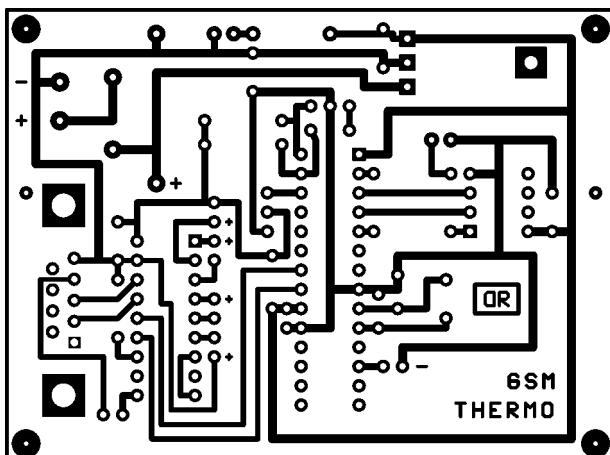


Figure 5.30.
Circuit imprimé.

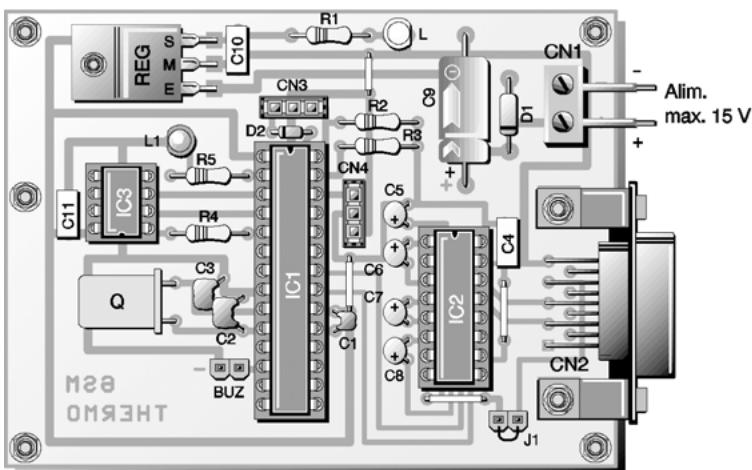


Figure 5.31.
Implantation
des composants.

Liste des composants

R1, R5 : 470 Ω

R2, R3 : 10 k Ω

R4 : 1 k Ω

C1 : 100 nF (pas de 2,54 mm)

C2, C3 : 22 pF / céramique

C4, C10, C11 : 100 nF / LCC jaune

C5, C6, C7, C8 : 1 μ F / tantale / 15 V

C9 : 220 μ F / électrolytique / 15 V

D1 : diode 1N4002

D2 : diode 1N4148

L, L1 : Led standard

Q : quartz 20 MHz

REG : régulateur 7805

BUZ : buzzer piezzo (sans électronique intégrée)

J1 : barrette HE10 2 contacts + cavalier

CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle pour CI / coudé à 90°

CN3 : connecteur pour câble de programmation (LEXTRONIC)

CN4 : connecteur pour écran LCD (LEXTRONIC) (facultatif)

IC1 : PICBASIC PB-3B (LEXTRONIC) + support DIL 28 broches (étroit)

IC2 : MAX232 + support DIL 16 broches

IC3 : DS1620 + support DIL 8 broches

est négative. Dans le cas d'une température positive le calcul est très simple, il suffit de diviser la donnée par deux. Dans le cas d'une température négative, il faut prendre en compte les 8 premiers bits et les soustraire à 255 puis diviser le résultat par deux. Prenons à titre d'exemple la donnée 1 11001110, le 9^e bit est à 1, donc il s'agit d'une température négative. On prend les 8 autres bits soit $11001110_{\text{bin}} = 206_{\text{dec}}$. On effectue l'opération $255 - 206 = 49$, $49/2 = 24,5$ soit 25.

Programme PicBasic : « thermo.bas »

```
'DECLARATION DES CONSTANTES
```

```
'-----
```

```
CONST BDS = 103
CONST RXD = 17
CONST TXD = 16
```

```
'DECLARATION DES VARIABLES
```

```
'-----
```

```
DIM index(3) AS BYTE
DIM SMS(13) AS BYTE
DIM i AS BYTE
DIM j AS INTEGER
DIM num AS BYTE
```

La variable T mémorise la température courante. TH mémorise le seuil de température haute. FLAG_TH et FLAG_TB indiquent si un SMS de dépassement de seuil haut ou bas a été envoyé.

```
DIM T AS INTEGER
DIM TH AS INTEGER
DIM FLAG_TH AS BYTE
DIM FLAG_TB AS BYTE
```

```
'INITIALISATION DES VARIABLES
```

```
'-----
```

La donnée index n'est pas déterminée automatiquement, il faudra donc l'initialiser (900 par défaut), il en va de même pour la mémoire (« ME » par défaut). Le seuil de température haute TH est ici fixé à 25 °C, vous pouvez bien entendu le modifier dans une plage allant de 1 à 125 °C. La ligne qui suit effectue un décalage à gauche de la donnée TH, ce qui correspond à une multiplication par deux. Les indicateurs de dépassement de seuil sont initialisés à zéro.

```
index(0)="9":index(1)="0":index(2)="0"
TH=25
TH=(TH<<1)
FLAG_TH=0
FLAG_TB=0
```

'INITIALISATION N° TELEPHONE UTILISE PAR DEFAUT POUR L'ENVOI DES
SMS

Le numéro de téléphone utilisé par défaut pour envoyer des SMS est initialisé dans la mémoire eeprom du PicBasic. Pour ne pas interférer avec la partie programme, le stockage se fait dans les 10 derniers emplacements de la mémoire de FF6_{hex} à FFF_{hex}. Cette mémorisation ne se réalise qu'une seule fois car le programme teste avant si l'adresse FF6_{hex} est vide (notez qu'un emplacement vide contient la donnée FF_{hex}).

```
IF EEREAD(&HFF6)=&HFF THEN
    EEWRITE &HFF6,"0"
    EEWRITE &HFF7,"6"
    EEWRITE &HFF8,"x"
    EEWRITE &HFF9,"x"
    EEWRITE &HFFA,"x"
    EEWRITE &HFFB,"x"
    EEWRITE &HFFC,"x"
    EEWRITE &HFFD,"x"
    EEWRITE &HFFE,"x"
    EEWRITE &HFFF,"x"
END IF

'TEST LIAISON SERIE
'-----
j=0
TEST: BEEP 9
SEROUT TXD,BDS,0,1,[ "AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSGS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable.

```
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500
```

```
'INITIALISATION DU ME (MODE TEXT)
'-----
SEROUT TXD,BDS,0,1,[ "AT+CMGF=1",13]
DELAY 500
SEROUT TXD,BDS,0,1,[ "AT+CPMS=",34,"ME",34,13]
DELAY 500

'PROGRAMME PRINCIPAL
'-----

DEBUT:
'Initialisation variable tableau SMS
'oooooooooooooooooooooooooooo
FOR i=0 TO 12
SMS(i)=0
NEXT i

'Mesure de la température
'oooooooooooooooooooo
```

Dans un premier temps le PicBasic demande quelle est la température, au circuit DS1620, en envoyant sur le bus SPI l'instruction AA_{hex} à l'aide de l'instruction spécifique SHIFTOUT. Cette instruction génère un signal d'horloge de synchronisation sur la sortie I/O2, tout en venant écrire sérielement les données présentent sur l'entrée I/O3. L'avant dernier paramètre de la commande définit le mode d'écriture, placé à zéro il indique que le LSB est prioritaire. Pour lire la réponse donnée par le DS1620 il faut ensuite utiliser l'instruction SHIFTIN, les lignes d'horloge et de donnée sont identiques, le dernier paramètre indique la taille en nombre de bits de la donnée récupérée ici positionnée à 9. Les instructions OUT 1,1 et OUT 1,0 assure l'initialisation du capteur de température avant et après sa consultation.

```
OUT 1,1
SHIFTOUT 2,3,0,&HAA
T=SHIFTIN(2,3,0,9)
OUT 1,0

'Regarde si un des seuils de T° est franchi
'oooooooooooooooooooooooooooo
```

Si la température mesurée est supérieure à 255, cela signifie que le 9^e bit est positionné à 1, donc qu'il s'agit d'une température négative, le sous-programme ALERT est alors appelé. Si la température mesurée est positive et supérieure au seuil défini par TH, le sous-programme ALERT est aussi appelé.

```
IF T>255 THEN
GOSUB ALERT
```

```
ELSE  
IF T>TH THEN GOSUB ALERT  
END IF
```

```
'Regarde si nouveau SMS  
'ooooooooooooooooooo
```

La mémoire du ME spécifiée par la donnée index est consultée pour savoir si un nouveau SMS est arrivé.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGR=" ]  
FOR i=0 TO 2  
IF index(i)>=48 AND index(i)<=57 THEN SEROUT  
TXD,BDS,0,1,[index(i)]  
NEXT i  
SEROUT TXD,BDS,0,1,[13]  
SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)~13]  
SUITE: IF SMS(0)=0 THEN GOTO RAZ  
FOR i=0 TO 10  
BEEP 9  
NEXT i
```

```
'Envoi de la température courante  
'ooooooooooooooooooooooooooo
```

Si les deux premières lettres de la commande réceptionnée sont « T » et « ? » un SMS contenant la température courante est envoyé. S'il s'agit des lettres « T » et « R » la mémorisation des seuils de dépassement FLAG_TH et FLAG_TB sont mis à zéro.

Le code ci-dessous génère le SMS qui contient la température ambiante mesurée, le seuil de détection température haute et le numéro de téléphone utilisé pour l'envoi des SMS d'alertes. Il fait notamment appel aux sous-programmes THM pour la conversion d'une température négative ($T > 255$) ou THP pour la conversion d'une température positive ($T < 255$).

```
IF SMS(0)="T" AND SMS(1)="?" THEN  
GOSUB NUMERO  
IF T>255 THEN GOSUB THM ELSE GOSUB THP  
j=(TH>>1)  
SEROUT TXD,BDS,0,1,[ " / Seuil Haut = +",DEC(j,3,1),"'C"]  
SEROUT TXD,BDS,0,1,[ " / Numero d'alerte : "]  
FOR j=&HFF6 TO &HFFF  
num=EEREAD(j)  
SEROUT TXD,BDS,0,1,[num]  
NEXT j  
SEROUT TXD,BDS,0,1,[26]  
END IF
```

```
'Remise à zéro des flags  
'ooooooooooooooooooo
```

Si les deux premières lettres de la commande réceptionnée sont « T » et « R » la mémorisation des seuils de dépassement FLAG_TH et FLAG_TB sont mis à zéro. La Led de signalement est éteinte.

```
IF SMS(0)="T" AND SMS(1)="R" THEN
    FLAG_TH=0
    FLAG_TB=0
    OUT 0,0
END IF

'Mise à jour du seuil de température haut
'oooooooooooooooooooooooooooooo
```

Il est possible de programmer par SMS la valeur du seuil haut qui déclenche l'envoi d'un message d'alerte. La commande est de la forme !!TH,temp, la donnée temp contient la valeur de seuil comprise entre + 001 et + 125 °C.

```
IF SMS(0)="T" AND SMS(1)="H" THEN
    SMS(3)=SMS(3)-48
    SMS(3)=SMS(3)*100
    SMS(4)=SMS(4)-48
    SMS(4)=SMS(4)*10
    SMS(5)=SMS(5)-48
    TH=SMS(3)+SMS(4)+SMS(5)
    TH=(TH<<1)
END IF

'Mise à jour du numéro utilisé pour l'envoi des SMS
'oooooooooooooooooooooooooooooo
```

Le numéro contenu dans la commande !!N,06xxxxxxxx est sauvegardé dans la mémoire eeprom du PicBasic aux adresses FF6_{hex} à FFF_{hex}. C'est ce numéro qui sera utilisé pour l'expédition des SMS.

```
IF SMS(0)="N" THEN
    i=2
    FOR j=&HFF6 TO &HFFF
        EEWRITE j,SMS(i)
        i=i+1
    NEXT j
END IF

'Efface le SMS dans la mémoire du téléphone
'oooooooooooooooooooooooooooooo
```

Cette partie du programme permet de systématiquement effacer le SMS en mémoire, ainsi le prochain SMS réceptionné aura toujours le même index. Ceci évite de prévoir une incrémentation de la variable index et surtout de saturer la mémoire du téléphone.

```
RAZ: SEROUT TXD,BDS,0,1,[ "AT+CMGD=" ]
FOR i=0 to 2
  IF index(i)>=48 AND index(i)<=57 THEN SEROUT
    TXD,BDS,0,1,[index(i)]
NEXT i
SEROUT TXD,BDS,0,1,[13]
DELAY 5000
GOTO DEBUT

'ENVOI DU MESSAGE D'ALERTE GEL ou INCENDIE
'-----
```

Le sous-programme ALERT génère le SMS d'alerte de dépassement des seuils. Il fait appel aux sous-programmes GEL si la température mesurée est inférieure ou égale à zéro, INC si la température est supérieure au seuil TH fixé en début de programme (ou par SMS), à condition qu'aucun des flags (FLAG_TB ou FLAG_TH) ne soit positionné à 1.

```
ALERT: IF FLAG_TB<>0 OR FLAG_TH<>0 THEN RETURN
  GOSUB NUMERO
  IF T>255 THEN
    GOSUB GEL
  ELSE
    IF T>TH THEN GOSUB INC
  END IF
  SEROUT TXD,BDS,0,1,[26]
  DELAY 5000
  RETURN
```

```
'NUMEROTATION POUR L'ENVOI D'UN SMS
'-----
```

Si la commande envoyée est de la forme !!T?,06xxxxxxxx le numéro utilisé pour l'envoi des SMS est celui spécifié par la commande. Pour les autres commandes le numéro utilisé est celui inscrit dans la mémoire eeprom du PicBasic.

NUMERO:

```
SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
IF SMS(1)="?" AND SMS(2)="." THEN
  FOR i=3 TO 12
    SEROUT TXD,BDS,0,1,[SMS(i)]
  NEXT i
ELSE
  FOR j=&HFF6 TO &HFFF
    num=EEREAD(j)
    SEROUT TXD,BDS,0,1,[num]
  NEXT j
END IF
SEROUT TXD,BDS,0,1,[34,13]
```

```
DELAY 1000  
RETURN  
  
'MESSAGES D'ALERTES  
'-----
```

Voici les sous-programmes GEL et INC qui font eux-mêmes appel aux sous-programmes THM et THP qui réalisent la conversion de la température. On notera la mise à 1 des variables FLAG_TB et FLAG_TH afin d'éviter que d'autres SMS d'alertes ne soient envoyés tant que ces mêmes variables ne seront pas remises à zéro par la commande !!TR.

```
GEL:  GOSUB THM  
      SEROUT TXD,BDS,0,1,[ " => RISQUE DE GEL"]  
      FLAG_TB=1  
      OUT 0,1  
      RETURN  
  
INC:   GOSUB THP  
      SEROUT TXD,BDS,0,1,[ " => INCENDIE"]  
      FLAG_TH=1  
      OUT 0,1  
      RETURN
```

```
'CALCUL DE LA TEMPERATURE  
'-----
```

Voici les sous-programmes, THM qui est chargé de la conversion d'une température négative et THP chargé de la conversion d'une température positive.

```
THM:   T=255 AND T  
      T=(255-T)  
      T=(T>>1)  
      SEROUT TXD,BDS,0,1,[ "TEMPERATURE : - ",DEC(T,3,1),"'C"]  
      RETURN  
  
THP:   T=(T>>1)  
      SEROUT TXD,BDS,0,1,[ "TEMPERATURE : + ",DEC(T,3,1),"'C"]  
      RETURN
```

Résumé des points importants

Voir Tableau 5.25.

Tableau 5.25.

THERMOMÈTRE	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Éléments du programme PicBasic à modifier	
<ul style="list-style-type: none"> • Code PIN (7208 par défaut) • Mémoire lecture SMS <mem1> (ME par défaut) • Index du prochain SMS reçu (900 par défaut) • Numéro de téléphone par défaut pour l'envoi des SMS • Seuil de température haute TH. Lorsque température > TH un SMS d'alerte est envoyé. 	
Commande SMS reçue	Action du montage
!!T?	Un SMS contenant la température mesurée est envoyé au numéro spécifié dans l'eeprom du PicBasic à l'aide de la commande !!N,06xxxxxx
!!T?,06xxxxxx	Un SMS contenant la température mesurée est envoyé au numéro indiqué
!!N, 06xxxxxx	Enregistre le numéro de téléphone indiqué dans l'eeprom du PicBasic. C'est ce numéro qui sera utilisé pour l'envoi des SMS d'alertes. Par défaut le numéro utilisé est celui en dur dans le programme
!!TH,temp	Fixe le seuil de température haute à la valeur spécifiée par la donnée temp (125 ≥ temp ≥ 001)
!!TR	Autorise le montage à envoyer d'autres SMS d'alerte, si la température est négative ou supérieure au seuil TH programmé

5.4 CARTE ENTRÉES/SORTIES PILOTÉE PAR GSM

La carte présentée ici est en quelque sorte une compilation des montages précédents. Elle permet la commande de 32 sorties logiques réparties sur 4 ports ainsi que la lecture de 32 entrées logiques réparties aussi sur 4 ports et de 8 entrées analogiques réparties sur 1 port. Le pilotage de ces 72 lignes est entièrement réalisé par l'envoi et la réception de SMS sur le réseau de téléphonie mobile.

PicBasic

Le cœur de notre carte est là encore un microcontrôleur PicBasic du constructeur Coréen COMFILE TECHNOLOGY. Il existe 3 familles de PicBasic, celui que nous avons choisi ici, le PIC-BASIC-2S, appartient à la deuxième famille, il est un bon compromis entre le coût et les possibilités offertes. Ce petit module hybride au format DIP 34 broches est constitué d'un PIC 16C74A-04, d'un quartz de 4,19 MHz, d'une mémoire eeprom

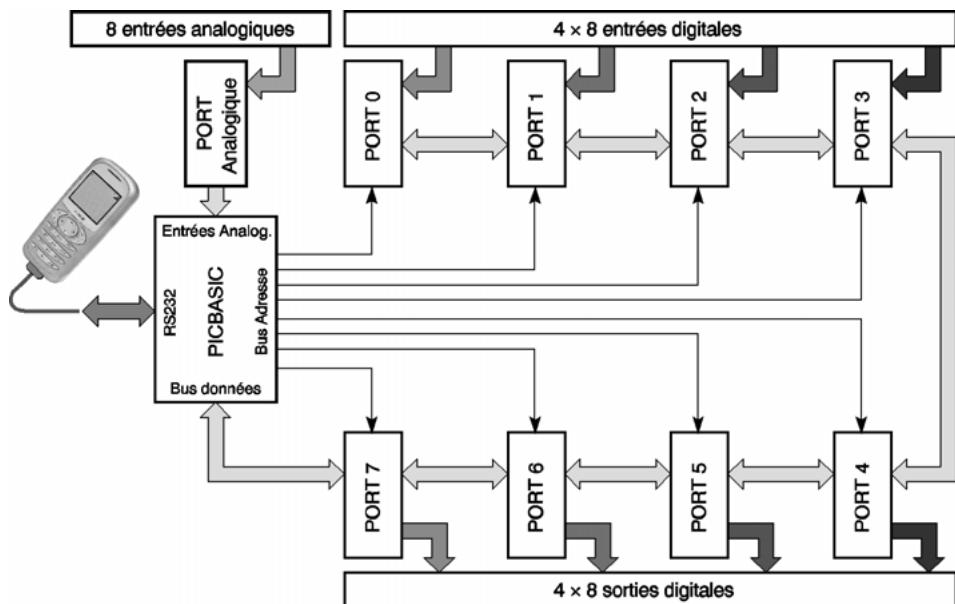


Figure 5.32.
Synoptique.

24LC64 d'une capacité de 8 Ko et d'un petit connecteur qui permet l'implantation en mémoire du programme. Sans avoir branché le fer à souder, on dispose déjà d'une minicarte tout à fait fonctionnelle. Comparativement au PicBasic 3B, le 2H dispose de 2 fois plus de mémoire programme, de 9 entrées logiques et 3 entrées analogiques supplémentaires. Toutefois le nombre d'instructions traitées par seconde est beaucoup plus faible, 56 000 pour le 3B seulement 1 000 pour le 2S, mais cette différence n'est pas du tout pénalisante dans le cadre de notre application.

Schéma électrique

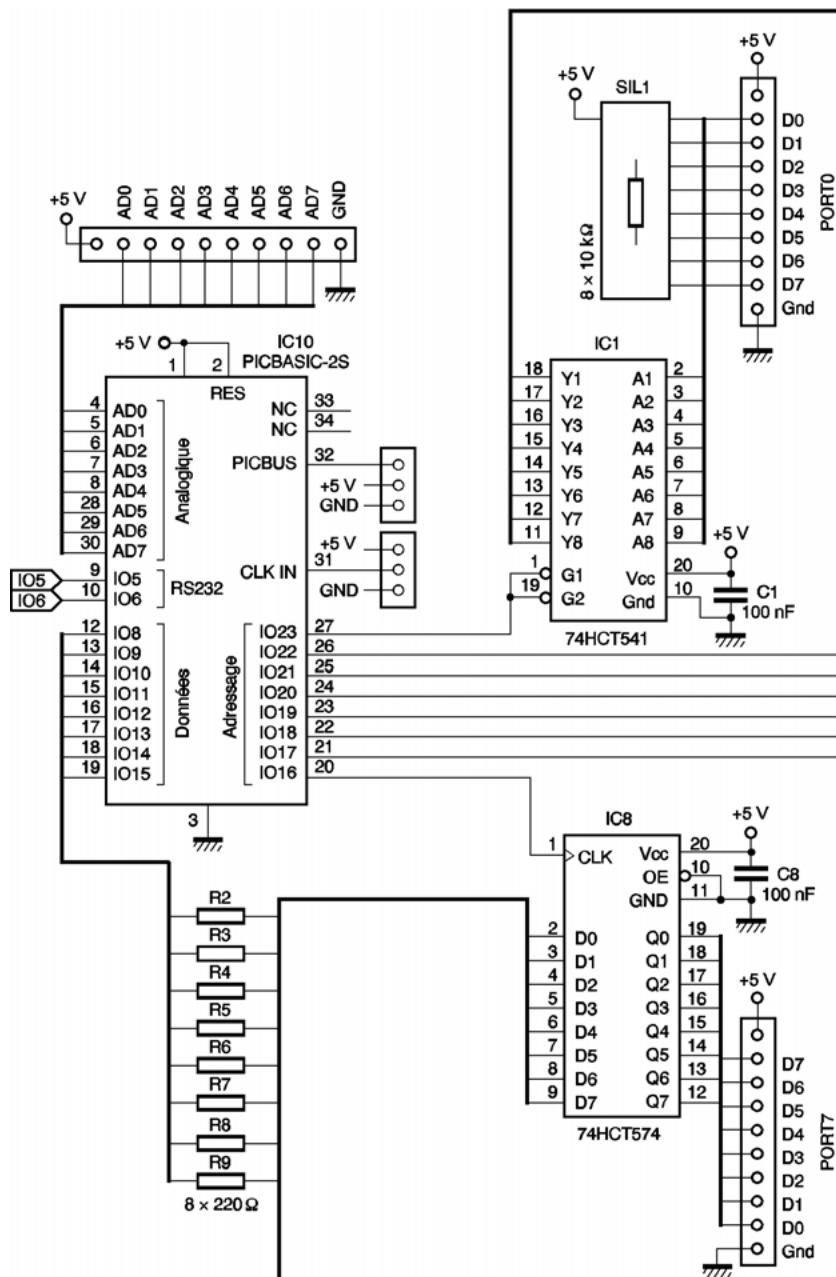
Le schéma électrique (Figure 5.33) peut sembler complexe, au premier coup d'œil. Nous allons voir qu'il n'en est rien.

Port série

Le PicBasic dispose d'instructions qui permettent d'utiliser deux de ses lignes I/O pour simuler une liaison RS232. Le format des données transférées est 8 bits de données avec 1 bit de start, 1 bit de stop et sans bit de parité.

L'instruction **SERIN Port, Param1, Mode, Param2, Adress, [Var1]** permet d'attendre la réception de données sous forme série selon le protocole RS232. La broche **Port** attend la ou les données **Var1** à une vitesse définie par **Param1** (voir tableau 5.26). Durant cette phase le PicBasic ne peut pas effectuer d'autres tâches et attend la réception des données pendant une durée définie par **Param2**.

INTERFACES GSM



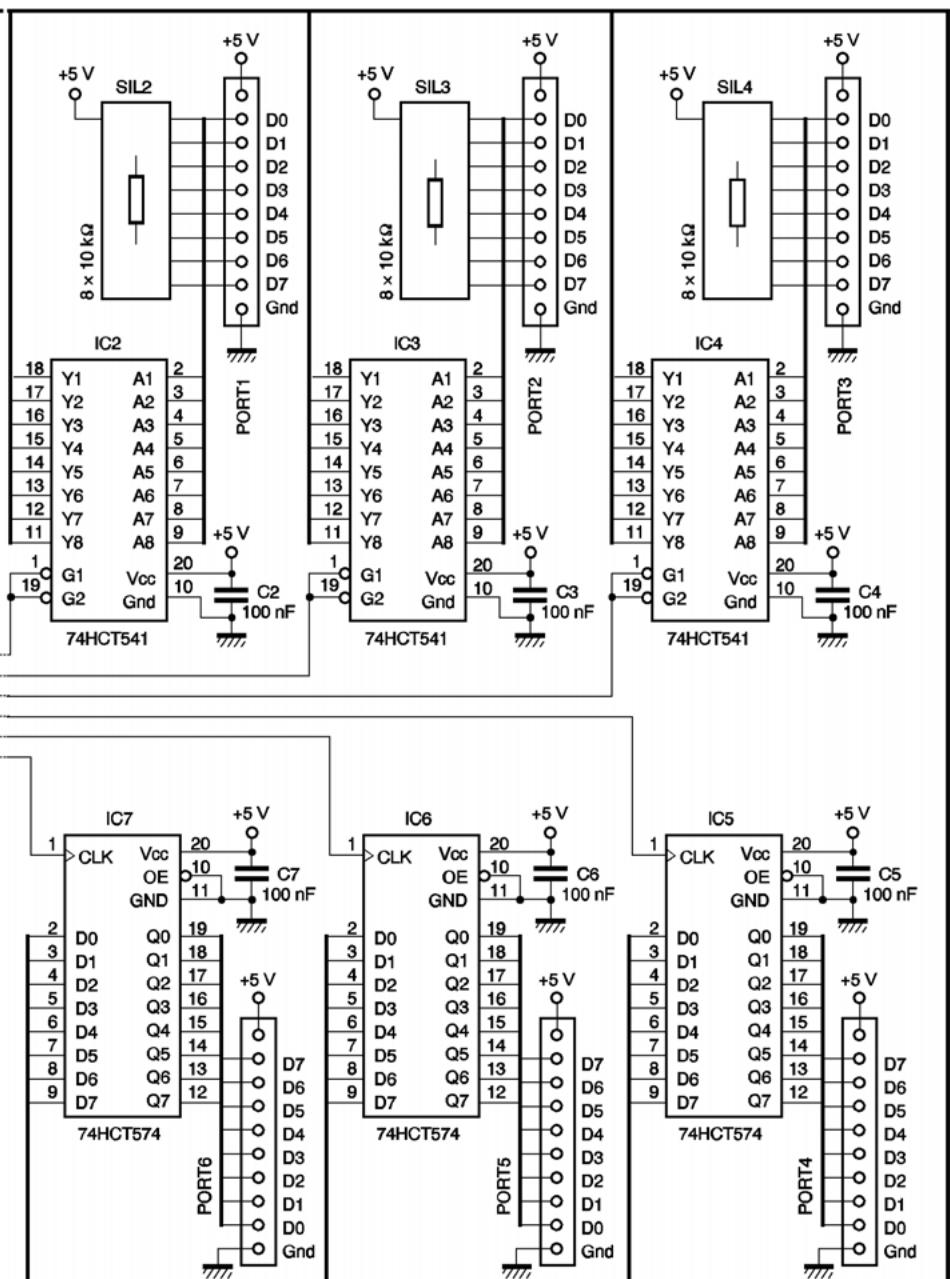


Figure 5.33.
 Schéma de la carte E/S
 pilotée par GSM.

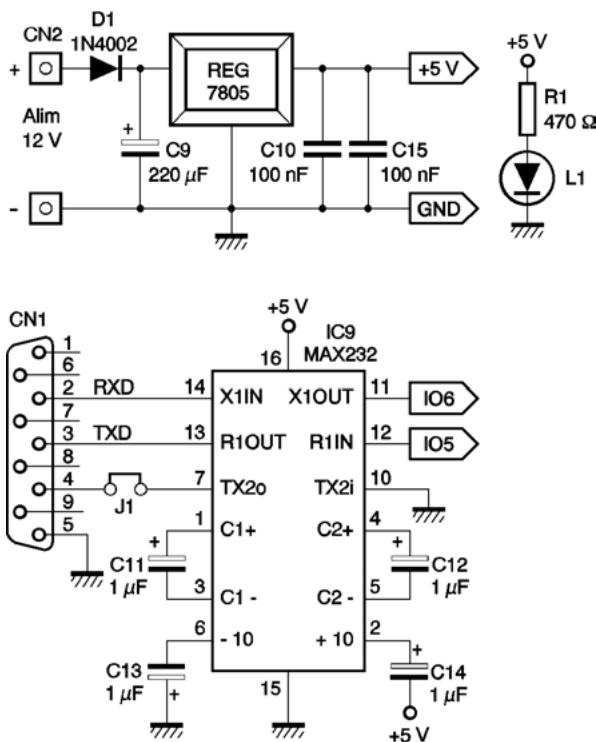


Figure 5.33 (suite).
Sections alimentation
et interface série.

Si la durée d'attente est dépassée, sans qu'aucune donnée ne soit reçue, le programme passera directement à l'adresse définie par **Adress**. Le paramètre **Mode** n'est pas utilisé et doit être positionné à 0.

L'instruction **SEROUT Port, Param1, Mode, Param2, Adress, [Var1]** permet de transmettre des données également sous forme série et au format RS232. La broche **Port** transmet la ou les données **Var1** à une vitesse définie par **Param1**. Le paramètre **Mode** permet d'instaurer une temporisation entre chaque caractère émis dont la durée en millisecondes est fonction de **Param2**.

Les données reçues et envoyées par ces deux instructions doivent être de type byte, c'est-à-dire comprises entre 0 et 255. Si une donnée de type integer, comprise entre 0 et 65 535, est envoyée, seuls les 8 bits de poids faible seront transmis.

La broche I/O5 (broche n° 9) sera utilisée pour recevoir les données série, et la broche I/O6 (broche n° 10) pour effectuer des transmissions. La vitesse de transmission a été fixée à 9 600 bauds, la valeur attribuée au paramètre **Param1** est donc 30. Comme notre carte doit pouvoir dialoguer avec le téléphone via le port série, il faut utiliser un circuit adaptateur de signaux

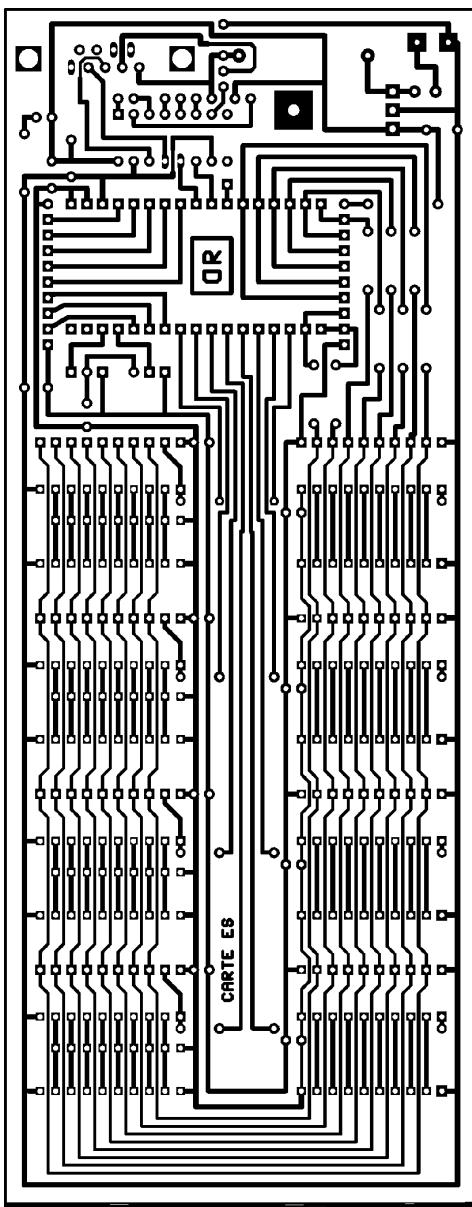


Figure 5.34.
Circuit imprimé
carte principale,
réduit de 10 %.

INTERFACES GSM

Figure 5.35.

**Implantation des composants
carte principale, réduite de 10 %.**

Liste des composants carte principale

R1 : 470 Ω

R2 à R9 : 220 Ω

C1 à C8 : 100 nF / céramique multicouche
(pas de 2,54)

C9 : 220 µF / 25 V / électrolytique

C10, C15 : 100 nF / LCC jaune

C11 à C14 : 1 µF / tantalé

D1 : 1N4002

L1 : Led

SIL1 à SIL4 :

réseau de résistance 10 kΩ (8R+commun)

IC1 à IC4 : 74HCT541

+ support DIL 20 broches

IC5 à IC8 : 74HCT574

+ support DIL 20 broches

IC9 : MAX232

+ support DIL 16 broches

IC10 : PICBASIC-2S

REG : régulateur 7805

+ dissipateur thermique

CN1 : DB9 mâle pour CI coudé à 90°

CN2 : bornier à vis 2 plots

8 barrettes femelles HE14 10 broches

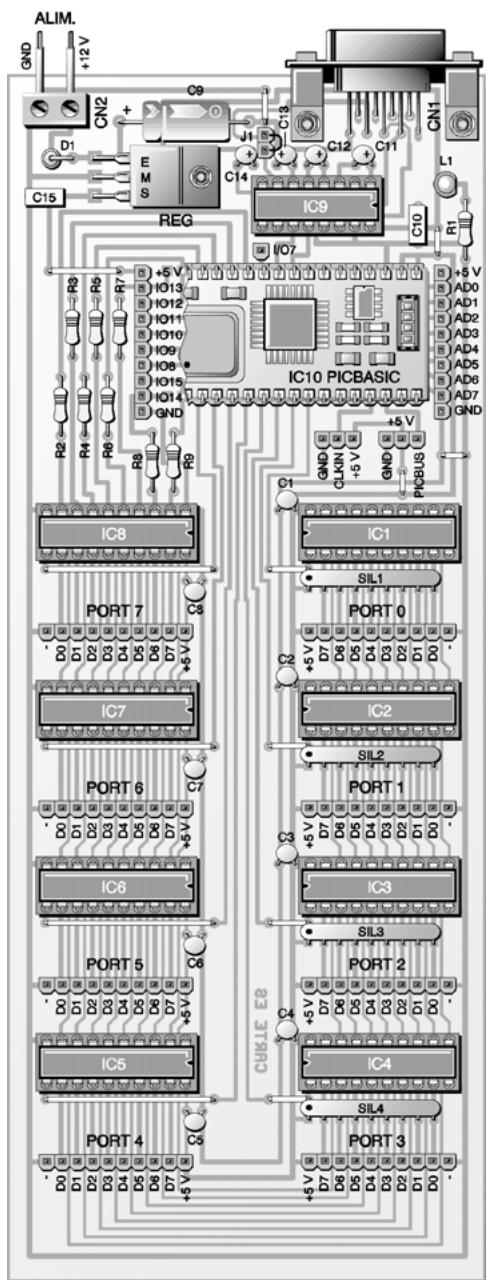
2 barrettes femelles HE14 17 broches

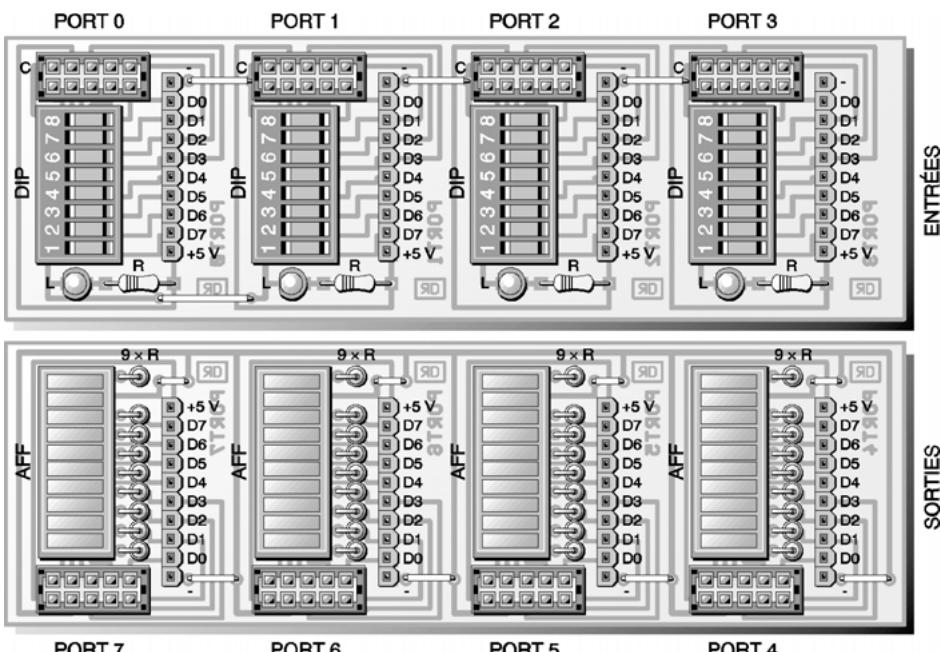
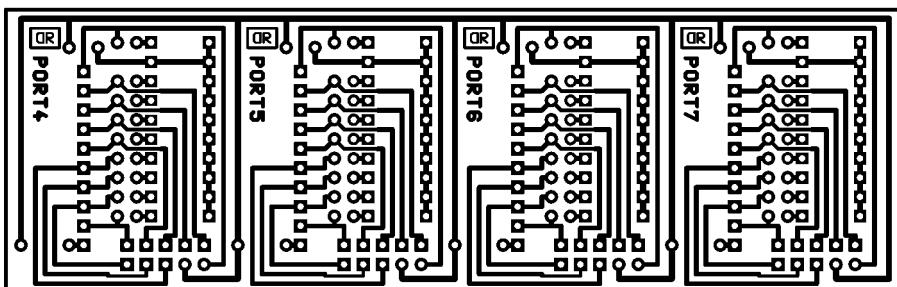
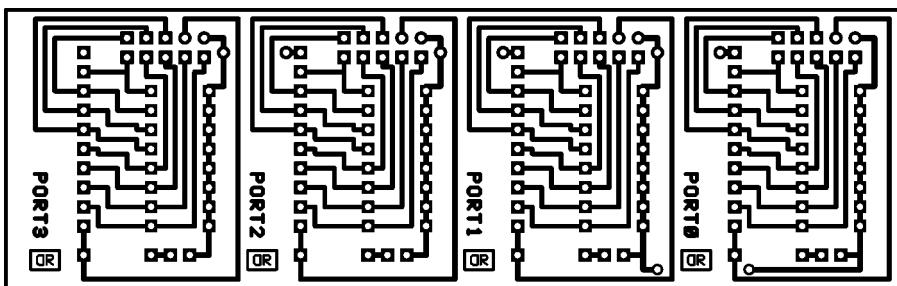
2 barrettes mâles HE14 10 broches

1 barrette mâle HE14 3 broches (facultatif)

1 connecteur pour écran LCD série

(facultatif)





Liste des composants plaques d'essais

AFF : 4 bargraphs 10 Led + support DIL 20 broches

DIP : 4 dips switches 8 interrupteurs + support DIL 16 broches

L : 4 Led rouges rectangulaires

R : 40 résistances 470 Ω

C : 8 connecteurs HE10 mâles 10 broches

8 barrettes mâles HE14 10 broches

Figure 5.36. (en haut)

**Circuit imprimé
plaques d'essais.**

Figure 5.37. (en bas)
**Implantation des composants
plaques d'essais.**

du type MAX232 (IC9 sur le schéma) câblé avec 4 condensateurs de 1 µF, afin de transformer les signaux TTL issus du PicBasic en signaux de + 10 V / - 10 V.

Tableau 5.26.
Différentes vitesses de transmission.

Vitesse (bauds)	Param1 (instructions SERIN et SEROUT)
2 400	138
4 800	66
9 600	30
19 200	11

Protocole de communication

Les commandes envoyées sous forme de SMS à travers le réseau GSM sont réceptionnées par le téléphone puis transmises à la carte via le port série. *Les commandes débutent toujours par les caractères « !! » suivis d'une lettre qui identifie la commande, par exemple « L » pour lecture, « E » pour écriture... puis vient le numéro du port concerné par la commande et finalement la donnée (dans le cas d'une opération d'écriture). Chaque paramètre est séparé par une virgule (tableau 5.27).*

Tableau 5.27.

Commande	Action
!!L, port	Lecture des 8 entrées du port concerné, avec $0 \leq \text{port} \leq 7$
!!E, port, Data	Écriture de la donnée data sur le port concerné, avec $4 \leq \text{port} \leq 7$
!!S, port, NumBit	Mise à 1 de la sortie NumBit du port concerné, avec $4 \leq \text{port} \leq 7$ et $0 \leq \text{NumBit} \leq 7$
!!R, port, NumBit	Mise à 0 de la sortie NumBit du port concerné, avec $4 \leq \text{port} \leq 7$ et $0 \leq \text{NumBit} \leq 7$
!!C, port, NumBit	Complémentie la sortie NumBit du port concerné, avec $4 \leq \text{port} \leq 7$ et $0 \leq \text{NumBit} \leq 7$
!!T	Effectue la lecture de toutes les entrées logiques, le résultat est envoyé sous forme de 4 octets correspondants respectivement aux ports 0, 1, 2 et 3
!!V	Effectue la lecture de toutes les entrées analogiques, le résultat est envoyé sous forme de 8 tensions lues sur les entrées AD0 à AD7

Ports E/S

La carte possède pas moins de 8 ports parallèles comportant chacun 8 lignes, 4 sont utilisables en entrée et sont numérotés de 0 à 3, 4 sont utilisables en sortie et sont numérotés de 4 à 7. Chacun des 8 ports est relié à 8 lignes du PicBasic. Il existe deux instructions spécifiques qui permettent de travailler simultanément avec des blocs de 8 lignes. L'instruction **BYTEIN(Param1)**

permet de récupérer la valeur de 8 entrées dans un mot binaire 8 bits dont chaque bit est l'image de chacune des entrées. Il est possible avec le PICBASIC-2S d'accéder à 3 blocs différents. L'instruction **BYTEOUT Port, Val** permet de sortir la valeur binaire 8 bits d'une donnée **Val** sur 8 sorties du PicBasic. Le paramètre **port** qui peut prendre les valeurs 1, 2 ou 3 permet d'accéder aux 3 blocs. Le bloc n° 1 que nous utilisons comme un bus de données bidirectionnel à l'aide des instructions **BYTEIN** et **BYTEOUT** est constitué des lignes I/O8 (broche n° 12) à I/O15 (broche n° 19). Dans le cas d'une opération d'écriture, la donnée est d'abord lue sur le port série puis recopiée sur le bus de données qui est alors configuré en sortie (instruction **BYTEOUT**). Dans le cas d'une opération de lecture, la donnée est d'abord lue par le bus de données qui est alors configuré en entrée (instruction **BYTEIN**), puis envoyée au téléphone via le port série. Il est évident qu'un seul port est utilisé à la fois, cette sélection s'effectuant par l'intermédiaire du bloc n° 2 constitué des lignes I/O16 (broche n° 20) à I/O23 (broche n° 27) qui est en quelque sorte utilisé comme bus d'adressage. Notez que la configuration des lignes du PicBasic en entrée ou en sortie est automatiquement réalisée. Par exemple si vous utilisez une instruction d'écriture comme **BYTEOUT**, les lignes concernées sont configurées en sortie. Avec l'instruction de lecture **BYTEIN** les lignes concernées sont configurées en entrée.

Sur chacune des 8 lignes du bus de données on trouve une résistance de $220\ \Omega$ chargée de protéger la ligne du PicBasic contre d'éventuelles mauvaises manipulations. Imaginons que vous effectuez une opération d'écriture sur le bus de données alors que le bus d'adressage rend actif, par exemple, le port n° 0. Si par malheur une sortie du bus de données qui est à l'état haut est reliée à une entrée du port qui est à l'état bas on obtient un court circuit qui à pour conséquence de détruire le PicBasic ! Mais, comme nous avons pris le soin d'insérer une résistance, la tension débitée par la sortie du PicBasic ne dépassera pas les 20 mA préconisés par le fabricant et le circuit sera sauvé, ouf !

Entrées

Chaque port d'entrée utilise un circuit 74HCT541. La sélection des ports 0, 1, 2 et 3 s'effectue par les bits G1 et G2 qui sont reliés respectivement à I/O23 (broche 27), I/O22 (broche 26), I/O21 (broche 25) et I/O20 (broche 24). Si une de ces lignes est à l'état bas l'octet présent sur l'entrée du 74HCT541 correspondant est recopié sur sa sortie et envoyé de ce fait sur le bus de données qui est alors configuré en entrée. Si les bits G1 et G2 sont à l'état haut, les sorties du 74HCT541 sont à l'état de haute impédance, ce qui

revient à dire que le circuit est déconnecté du bus de données. Les entrées du 74HCT541 sont reliées à des résistances de rappel. Si l'entrée n'est pas utilisée, elle est mise à l'état haut par une résistance de $10\text{ k}\Omega$.

Sorties

Chaque port de sortie utilise un circuit 74HCT574. La sélection des ports 4, 5, 6 et 7 s'effectue par les broches CLK (*clock*) qui sont reliées respectivement à I/O19 (broche 23), I/O18 (broche 22), I/O17 (broche 21) et I/O16 (broche 20). Chacune des 8 entrées de chaque 74HCT574 est reliée au bus de données qui est alors configuré en sortie. Si une des entrées CLK est soumise à un front montant, la donnée présente en entrée du 74HCT574 correspondant est recopiée sur sa sortie. L'état de la sortie étant mémorisé dans un tampon jusqu'à la prochaine écriture.

Port analogique

Le PICBASIC-2S dispose d'origine de 8 entrées analogiques AD0 à AD7 possédant une résolution de 8 bits, soit une précision de 20 mV sur la mesure. L'instruction très simple comme **ADIN(port)** permet de connaître la valeur de la tension présente sur une broche précise. La valeur à lire doit être impérativement comprise entre 0 et + 5 V sous peine de détériorer le circuit. Pour la lecture de niveaux supérieurs il faudra avoir recours par exemple à des ponts diviseurs de tensions ou à des amplis OP câblés en diviseurs de tension. Le paramètre **port** correspond à la broche du module qui reçoit la valeur à mesurer : broches 0 à 4 et 24 à 26. La valeur de la lecture est de type byte (comprise entre 0 et 255), l'équation qui permet de calculer la tension est : $Tension = (valeur \times 5)/255$. Par exemple pour une valeur de 125 on obtient une tension égale à 2,45 V.

Réalisation

Il faudra un minimum de vigilance lors de la réalisation, la carte comporte de nombreuses pistes proches les une des autres et relativement fines. La majorité des pastilles seront à percer à l'aide d'un foret de 0,8 mm de diamètre. Seules les pastilles du bornier et du régulateur seront à percer à 1,5 mm. Concernant le montage des composants, on commencera par la mise en place des 13 straps. On soudera ensuite les résistances puis les condensateurs dont il faudra impérativement respecter la polarité, particulièrement pour ceux destinés au MAX232. On terminera par les supports des CI, le 7805 et le connecteur DB9. Le support du PicBasic est constitué de deux barrettes HE14 femelles de 17 contacts. Avant la mise en place des CI sur leurs supports

respectifs, il est conseillé de mettre le montage sous tension et de vérifier la tension d'alimentation à l'aide d'un voltmètre qui doit être égale à + 5 V à $\pm 5\%$ près.

Programme PicBasic : « ces.bas »

Étudions maintenant en détail le programme destiné au PicBasic.

'DECLARATION DES CONSTANTES

'-----

Il ne faut surtout pas se priver de l'utilisation des constantes qui facilitent la compréhension et la maintenance d'un programme. De plus les constantes ne sont utilisées que par le compilateur, elles ne prennent donc pas de place dans la mémoire du PicBasic.

```
'Constantes dédiées à la gestion du port série  
'ooooooooooooooooooooooooooooooo
```

Déclaration des constantes utilisées par les instructions SERIN et SEROUT. TXD correspond à la ligne I/O6 du PicBasic, RXD correspond à ligne I/O5. BDS (pour bauds) définit la vitesse de transmission ici fixée à 9 600 car BDS = 30.

```
CONST BDS = 30  
CONST RXD = 5  
CONST TXD = 6
```

```
'Constantes utilisées pour l'adressage de l'eeprom  
'ooooooooooooooooooooooooooooooo
```

De nombreux paramètres utiles à la carte (code PIN, Mem, Index, Numéro...) sont sauvegardés dans la mémoire eeprom du PicBasic. Les plages d'adresses sont identifiées par des constantes, D_x définit l'adresse de début, F_x définit l'adresse de fin.

```
CONST D_FlagPort = &H1FE5  
CONST F_FlagPort = &H1FE8  
CONST D_PORT = &H1FE9  
CONST F_PORT = &H1FEC  
CONST D_PIN = &H1FED  
CONST F_PIN = &H1FF0  
CONST D_Mem = &H1FF1  
CONST F_Mem = &H1FF2  
CONST D_Index = &H1FF3  
CONST F_Index = &H1FF5  
CONST D_Numero = &H1FF6  
CONST F_Numero = &H1FFF
```

INTERFACES GSM

Tableau 5.28.
Cartographie
de la mémoire
eeprom
du PicBasic.

Adresse HEX	Constante	Nature	Espace
0000	_	Premier octet mémoire programme	1 857 octets
...	_	...	
...	_	...	
0740	_	Dernier octet mémoire programme	
0741			6 308 octets non utilisés
1FE4			
1FE5	D_FlagPort	1 ^{er} octet (Flag Port n° 0)	4 octets
1FE6	_	2 ^e octet (Flag Port n° 1)	
1FE7	_	3 ^e octet (Flag Port n° 2)	
1FE8	F_FlagPort	4 ^e octet (Flag Port n° 3)	
1FE9	D_PORT	1 ^{er} octet (Port n° 0)	4 octets
1FEA	_	2 ^e octet (Port n° 1)	
1FEB	_	3 ^e octet (Port n° 2)	
1FEC	F_PORT	4 ^e octet (Port n° 3)	
1FED	D_PIN	1 ^{er} chiffre	4 octets
1FEE	_	2 ^e chiffre	
1FEF	_	3 ^e chiffre	
1FF0	F_PIN	4 ^e chiffre	
1FF1	D_Mem	1 ^{er} caractère	2 octets
1FF2	_	2 ^e caractère	

1FF3	D_Index	1 ^{er} chiffre	3 octets
1FF4	_	2 ^e chiffre	
1FF5	F_Index	3 ^e chiffre	
1FF6	D_Numero	1 ^{er} chiffre	10 octets
1FF7	_	2 ^e chiffre	
1FF8	_	3 ^e chiffre	
1FF9	_	4 ^e chiffre	
1FFA	_	5 ^e chiffre	
1FFB	_	6 ^e chiffre	
1FFC	_	7 ^e chiffre	
1FFD	_	8 ^e chiffre	
1FFE	_	9 ^e chiffre	
1FFF	F_Numero	10 ^e chiffre	

'Constantes dédiées à la gestion des E/S de la carte

'oooooooooooooooooooooooooooo

La constante « adresse » représente le bloc de 8 bits numéro 2 (I/O16 à I/O23), la constante « donnée » représente le bloc numéro 1 (I/O8 à I/O15). Ces deux constantes seront utilisées dans le corps du programme avec les instructions BYTEIN et BYTEOUT.

```
CONST adresse = 2
CONST donnee = 1
```

La constante MaskPort est une constante de type tableau qui contient 8 valeurs représentées ici en binaire. Chaque valeur est accessible par un indice, par exemple MaskPort(2)= 11010000_{bin}. Ces valeurs seront utilisées par le bus d'adresse pour sélectionner un des 8 ports de la carte, par exemple MaskPort(2) sélectionne le port numéro 2, car le bit 5 correspondant à la sortie I/O21 est à zéro. Rappelons que les ports 0 à 3 sont actifs si les lignes correspondantes du bus d'adressage (I/O23 à I/O20) sont à l'état bas. Alors que les ports 4 à 7 sont actifs si les lignes correspondantes du bus d'adressage (I/O19 à I/O16) passent de l'état bas à l'état haut (front montant).

```
CONST BYTE MaskPort= (&b01110000,&b10110000,&b11010000,
&b11100000,&b11111000,&b11110100,
&b11110010,&b11110001)
```

```
'DECLARATION DES VARIABLES
'-----
'Variable dédiée au stockage SMS
'oooooooooooooooooooooooooooo
DIM SMS(30) AS BYTE

'Variables dédiées au pilotage des E/S de la carte
'oooooooooooooooooooooooooooooooooooo
DIM i AS BYTE
DIM n AS BYTE
DIM k AS BYTE
DIM DATA AS BYTE
DIM ValPort(4) AS BYTE
DIM bin(8) AS BYTE
DIM FlagPort(4) AS BYTE
DIM j AS INTEGER
DIM h AS INTEGER
DIM j1 AS INTEGER
DIM j2 AS INTEGER
DIM V AS INTEGER
DIM V1 AS INTEGER
DIM V2 AS INTEGER
DIM V3 AS INTEGER

'INITIALISATION DU BUS D'ADRESSE ET DE DONNEES
'-----
```

Toutes les sorties de la carte sont positionnées à l'état logique bas (Ports 4 à 7).

```
BYTEOUT adresse,&b11110000
BYTEOUT donnee, &b00000000
BYTEOUT adresse,&b11111111
```

```
'INITIALISATION DE LA VARIABLE TABLEAU ValPort
'-----
FOR i=0 TO 3
ValPort(i)=0
NEXT i
```

```
'TEST LIAISON SERIE
'-----
```

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit : AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte. Les caractères « AT » suivis du caractère <CR>=13_{dec} sont envoyés par la commande SEROUT. L'instruction SERIN permet d'attendre l'éventuelle réponse « OK » pendant 2 000 ms

(soit 2 s). Si les caractères OK sont réceptionnés dans le temps donné, le caractère suivant soit <CR> est placé dans la variable i. Dans le cas contraire le programme saute à la ligne repérée par l'étiquette TEST car i est vide. Il suffit de tester le contenu de i pour savoir si la liaison est établie. En phase de paramétrage, le montage est connecté à un PC, si à la commande « AT » le PC répond par « OK » suivi de la lettre P le montage se place en phase de programmation, si le PC répond par la lettre L, c'est la phase de lecture qui est activée.

```
i=0
TEST: SEROUT TXD,BDS,0,1,[ "AT",13]
SERIN RXD,BDS,0,2000,TEST,[WAIT("OK"),i]
IF i=0 THEN GOTO TEST
IF i=="P" THEN GOTO PROG_EE
IF i=="L" THEN GOTO LECT_EE

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500
```

```
'CODE PIN
'-----
```

En principe le code PIN qui autorise l'accès aux fonctions du téléphone doit être composé à chaque mise sous tension. Avec un téléphone classique vous pouvez le saisir à partir du clavier. Ce qui n'est plus possible si vous utilisez un terminal GSM intégré, pour la simple et bonne raison qu'il ne dispose pas de clavier ! L'instruction « AT+CPIN » suivie de votre code PIN est dans ce cas incontournable. Le code PIN est stocké dans la mémoire eeprom du PicBasic entre les adresses D_PIN et F_PIN.

```
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34]
j1=D_PIN:j2=F_PIN
GOSUB READ_EE
SEROUT TXD,BDS,0,1,[34,13]
DELAY 500

'MEMOIRE UTILISEE POUR LE STOCKAGE DES SMS
'-----
```

La mémoire utilisée pour le stockage des SMS réceptionnés (<mem1>) est également contenue dans la mémoire eeprom du PicBasic entre les adresses D_MEM et F_MEM.

```
SEROUT TXD,BDS,0,1,[ "AT+CPMS=",34]
j1=D_MEM:j2=F_MEM
```

```
GOSUB READ_EE  
SEROUT TXD,BDS,0,1,[34,13]  
DELAY 500
```

```
'INITIALISATION DU ME
```

```
'-----
```

■ Le ME est configuré en mode TEXT.

```
SEROUT TXD,BDS,0,1,[ "AT+CMGF=1",13]  
DELAY 500
```

```
'INITIALISATION DE LA VARIABLE TABLEAU FLAGPORT
```

```
'-----
```

■ Les 8 variables FlagPort sont chargées avec les données lues entre les adresses D_FlagPort et F_FlagPort de l'eprom. (voir la partie de programme SCRUT pour comprendre leur utilité).

```
i=0  
FOR j=D_FlagPort TO F_FlagPort  
FlagPort(i)=EEREAD(j)  
i=i+1  
NEXT j
```

```
'INITIALISATION DES VARIABLES
```

```
'-----
```

```
DEBUT:
```

```
'Initialisation du bus de données  
oooooooooooooooooooooooooooo
```

■ Le bus d'adresse est positionné de manière à ce qu'aucun port ne soit sélectionné. Pour des raisons de sécurité, on effectue une lecture « bidon » du bus de données pour configurer les broches correspondantes du PicBasic en entrées, cette précaution évite les courts-circuits si par mégarde un des ports 0 à 3 est actif alors que le bus de données est configuré en sortie.

```
BYTEOUT adresse,&b11110000  
DATA=BYTEIN(donnee)
```

```
'Initialisation de la variable de stockage SMS
```

```
oooooooooooooooooooooooooooooooooooo
```

```
FOR i=0 TO 29  
SMS(i)=0  
NEXT i
```

```
'ATTENTE SMS
```

```
'-----
```

La lecture du SMS est provoquée par la commande « AT+CMGR=<index> », le paramètre Index est récupéré dans la mémoire eeprom du PicBasic. Dès la réception des caractères « !! » les 12 caractères suivants sont placés dans la variable SMS. Si les deux points d'exclamation ne sont pas réceptionnés dans les 5 s, le programme bascule sur l'étiquette SCRUT.

```
SEROUT TXD,BDS,0,1,["AT+CMGR="]
j1=D_Index:j2=F_Index
GOSUB READ_EE_Index
SERIN RXD,BDS,0,5000,SUITE,[WAIT("!!"),SMS(0)~12]
SUITE: IF SMS(0)=0 THEN GOTO SCRUT
```

En l'état actuel du programme, si l'on considère que le SMS envoyé était de la forme « !!E,1,154 », la variable tableau SMS doit contenir ce qui est indiqué **tableau 5.29** (l'état des variables SMS(7) à SMS(29) nous est dans ce cas indifférent).

SMS(0)	SMS(1)	SMS(2)	SMS(3)	SMS(4)	SMS(5)	SMS(6)
E	,	4	,	1	5	4

Tableau 5.29.

- SMS(0) contient l'identifiant de la commande, ici « E » pour Écriture.
- SMS(2) contient le numéro du port sollicité.
- SMS(4), SMS(5) et SMS(6) contiennent la donnée à écrire sur le port.

'CONVERSION ASCII -> DECIMAL

'-----

Les valeurs contenues dans la variable SMS sont codées sous forme de caractères ASCII, pour calculer la valeur numérique exprimée en décimal il suffit de retrancher la valeur 48_{dec} qui est le code ASCII du chiffre zéro. Par exemple si $SMS(2) = 4_{ascii} = 52_{dec}$, d'où $SMS(2) - 48_{dec} = 52_{dec} - 48_{dec} = 4_{dec}$. Attention cette conversion ne doit pas être faite concernant les commandes $!N,06xxxxxxxx$ et $!F,xxxx$.

```
IF SMS(0)<>"N" AND SMS(0)<>"F" THEN
    SMS(2)=SMS(2)-48
    SMS(4)=SMS(4)-48
    SMS(5)=SMS(5)-48
    SMS(6)=SMS(6)-48
END IF
```

'AIGUILLAGE DU PROGRAMME EN FONCTION DU CONTENU DE SMS(0)

'-----

Le test du contenu de la variable SMS(0) nous indique vers quelle partie le programme doit s'orienter. Dans le cas où la commande reçue n'est pas valide, le programme bascule sur l'étiquette RAZ ce qui provoque la suppression du SMS.

```
IF SMS(0)="L" THEN GOTO LECTURE  
IF SMS(0)="E" THEN GOTO ECRITURE  
IF SMS(0)="S" THEN GOTO SETBIT  
IF SMS(0)="R" THEN GOTO RESETBIT  
IF SMS(0)="C" THEN GOTO COMPLBIT  
IF SMS(0)="T" THEN GOTO LECTURES  
IF SMS(0)="V" THEN GOTO ANALOG  
IF SMS(0)="N" THEN GOTO MAJNUM  
IF SMS(0)="F" THEN GOTO FLAG  
GOTO RAZ
```

'LECTURE ENTREES DU PORT POINTE PAR SMS(2)

Envoi d'un SMS contenant l'état logique du port spécifié. Le sous-programme NUM compose le numéro de téléphone du destinataire, LEC1PORT effectue la lecture de la valeur décimale et binaire présente sur le port sollicité. Finalement le code ASCII 26_{dec}=EOF est envoyée sur la sortie TxD, ce qui déclenche l'envoi du SMS sur le réseau GSM.

LECTURE:

```
GOSUB NUM  
GOSUB LEC1PORT  
SEROUT TXD,BDS,0,1,[26]  
DELAY 5000  
GOTO RAZ
```

'LECTURE DE TOUTES LES ENTREES

Effectue la lecture de tous les ports de la carte. Le sous-programme NUM compose le numéro de téléphone du destinataire du SMS. Pour la lecture de tous les ports on fait appel 8 fois au sous-programme LEC1PORT en incrémentant d'une unité la variable SMS(2) à chaque appel. Finalement le code ASCII 26_{dec}=EOF est envoyé sur la sortie TxD, ce qui déclenche l'envoi du SMS sur le réseau GSM.

LECTURES:

```
GOSUB NUM  
FOR i=0 TO 7  
SMS(2)=i  
IF i<>0 THEN SEROUT TXD,BDS,0,1,[ " / " ]  
GOSUB LEC1PORT  
NEXT i
```

```
SEROUT TXD,BDS,0,1,[26]  
DELAY 5000  
GOTO RAZ
```

'ÉCRITURE SUR PORT CONCERNÉ

'-----
ÉCRITURE:

```
'Conversion de la valeur de consigne  
'ooooooooooooooooooooooo
```

■ SMS(4) contient le chiffre des centaines de la valeur de consigne, SMS(5) le chiffre des dizaines et SMS(6) le chiffre des unités. Les 3 lignes ci-dessous permettent de reconstituer facilement la valeur de consigne qui est ensuite enregistrée dans la variable DATA.

```
SMS(4)=SMS(4)*100  
SMS(5)=SMS(5)*10  
DATA=SMS(4)+SMS(5)+SMS(6)
```

```
'Mise à jour du port concerné  
'ooooooooooooooooooooooo
```

■ Lors d'une opération d'écriture, on commence par envoyer la valeur de consigne contenue dans DATA sur le bus de données. Le bus d'adresse est ensuite positionné de manière à ce que la donnée en question soit transférée sur le port pointé par SMS(2). Notez que l'on mémorise la valeur appliquée sur le port dans la variable tableau ValPort.

MAJBIT:

```
IF SMS(2)>=4 THEN  
BYTEOUT donnee,DATA  
BYTEOUT adresse,MaskPort(SMS(2))  
n=SMS(2)-4  
ValPort(n)=DATA  
END IF  
GOTO RAZ
```

'MISE A 1 D'UN BIT SUR PORT CONCERNÉ

'-----

■ Grâce au sous-programme UnBIT, la variable j contient le bit qu'il faut mettre à 1. Pour ne mettre à 1 que la sortie concernée, il faut faire un OU logique entre j et la valeur en cours sur le port.

SETBIT:

```
GOSUB UnBIT  
DATA=j OR ValPort(n)  
GOTO MAJBIT
```

Exemple : dans cet exemple, seul le bit n° 4 passe à 1, tous les autres bits conservent leur état.

OU	j	0	0	0	1	0	0	0	0
	ValPort(n)	1	1	0	0	1	1	0	0
	DATA	1	1	0	1	1	1	0	0

'MISE A 0 D'UN BIT SUR PORT CONCERNÉ

'-----

Grâce au sous-programme UnBIT, la variable j contient le bit qu'il faut mettre à 0. Pour ne mettre à 0 que la sortie concernée, il faut faire un ET logique entre le complément de j (obtenu par 255-j) et la valeur en cours sur le port.

RESETBIT:

```
GOSUB UnBIT
DATA=(255-j) AND ValPort(n)
GOTO MAJBIT
```

Exemple : dans cet exemple, seul le bit n° 4 passe à 0, tous les autres bits conservent leur état.

ET	j	0	0	0	1	0	0	0	0
	(255-j)	1	1	1	0	1	1	1	1
	ValPort(n)	1	1	0	1	1	1	0	0
	DATA	1	1	0	0	1	1	0	0

'INVERSE L'ETAT D'UN BIT SUR PORT CONCERNÉ

'-----

Grâce au sous-programme UnBIT, la variable j contient le bit qu'il faut mettre à 0. Pour ne mettre à 0 que la sortie concernée, il faut faire un OU EXCLUSIF (XOR) entre j et la valeur en cours sur le port.

COMPLBIT:

```
GOSUB UnBIT
DATA=j XOR ValPort(n)
GOTO MAJBIT
```

Exemple 1 : dans cet exemple, seul le bit n° 4 change d'état et passe à 1, tous les autres bits conservent leur état.

XOR	j	0	0	0	1	0	0	0
	ValPort(n)	1	1	0	0	1	1	0
	DATA	1	1	0	1	1	1	0

Exemple 2 : dans cet exemple, seul le bit n° 4 change d'état et passe à 0, tous les autres bits conservent leur état.

XOR	j	0	0	0	1	0	0	0
	ValPort(n)	1	1	0	1	1	1	0
	DATA	1	1	0	0	1	1	0

'LECTURE DES 8 ENTREES ANALOGIQUES

'-----

Voici la partie de programme chargée de lire les 8 entrées analogiques de la carte et d'envoyer le résultat sous forme de SMS. L'acquisition s'effectue en deux parties car les entrées analogiques ne se suivent pas (broches I/O0 à I/O4 puis I/O24 à I/O26). Le sous-programme CONVTENS permet de convertir la valeur lue en décimale codée sur 8 bits en une valeur codée en ASCII.

ANALOG:

```
SEROUT TXD,BDS,0,1,[ "ETAT DES 8 ENTREES ANALOGIQUES : " ]
FOR i=0 to 7
    k=i
    IF i>4 THEN k=i+19
    DATA=ADIN(k)
    GOSUB CONVTENS
    n=i+48
    SEROUT TXD,BDS,0,1,[ "E",n," = +",V1,",",V2,V3,"v", " " ]
NEXT i
SEROUT TXD,BDS,0,1,[26]
GOTO RAZ
```

'MISE A JOUR DU NUMERO UTILISE POUR L'ENVOI DES SMS

'-----

Non seulement il est possible de modifier le numéro grâce au logiciel « ConfigES_GSM.exe » (montage connecté à un PC) mais aussi par SMS (montage connecté à un téléphone GSM). Le numéro contenu dans la commande !IN,06xxxxxxxx est sauvegardé dans la mémoire eeprom du PicBasic aux adresses D_Numero à F_Numero. C'est ce numéro qui sera utilisé pour l'expédition des SMS.

```
MAJNUM:  
    i=2  
    FOR j = D_Numer0 TO F_Numer0  
        EEWRITE j,SMS(i)  
        i=i+1  
    NEXT j
```

```
'POSITIONNEMENT DES FLAGS
```

```
'-----
```

Les variables FlagPort peuvent être positionnées par l'envoi d'un SMS de la forme : !!F,f0f1f2f3. Si fx = 1 le port n° x ne peut pas déclencher l'envoi d'un SMS. Si fx = 0 le port n° x peut déclencher l'envoi d'un SMS. Avec 0 ≤ x ≤ 3. Si fx = ? la valeur du flag n'est pas modifiée (voir la partie de programme SCRUT pour comprendre l'utilité des flags).

```
FLAG:
```

```
n=2  
FOR i=0 TO 4  
    If SMS(n)<>"?" THEN FlagPort(i)=SMS(n)  
    n=n+1  
NEXT i  
GOTO RAZ
```

```
'EFFACE LE SMS EN MEMOIRE
```

```
'-----
```

Le SMS est systématiquement effacé, à l'aide de la commande AT+CMGD suivie de l'index, pour éviter une saturation de la mémoire utilisée, et une incrémentation de l'index. Du fait chaque SMS reçu aura le même index.

```
RAZ:
```

```
SEROUT TXD,BDS,0,1,[ "AT+CMGD=" ]  
j1=D_Index:j2=F_Index  
GOSUB READ_EE_Index  
DELAY 500  
GOTO DEBUT
```

```
'SCRUTATION DES ENTREES
```

```
'-----
```

Après avoir vérifié si un nouveau SMS est présent dans la mémoire du téléphone et, le cas échéant, traité celui-ci, le PicBasic effectue une lecture des ports n° 0 à 3 et compare les 4 valeurs lues avec celles situées entre les adresses D_PORT et F_PORT. En cas d'égalité un SMS contenant l'état des entrées du port est envoyé, un message entre parenthèses indique qu'il s'agit d'un message d'alerte. L'envoi ne peut se faire que si la variable

FlagPort correspondante est à zéro. Lors du premier envoi la variable FlagPort est mise systématiquement à 1 pour éviter que le port correspondant envoie d'autres SMS. Le logiciel « ConfigES_GSM.exe » permet à l'utilisateur de définir les valeurs qui déclenchent l'envoi d'un SMS, il est également possible d'inhiber cette fonction en positionnant les variables FlagPort à 1.

SCRUT:

```

h=D_PORT
FOR k=0 TO 3
  IF FlagPort(k)="0" THEN
    BYTEOUT adresse,MaskPort(k)
    DATA=BYTEIN(donnee)
    IF EEREAD(h)=DATA THEN
      FlagPort(k)="1"
      GOSUB NUM
      SMS(2)=k
      GOSUB MESS1PORT
      SEROUT TXD,BDS,0,1,[" (Ceci est un message d'alerte
      !)",26]
    END IF
  END IF
  h=h+1
NEXT k
GOTO RAZ

```

```
' *****
' PROGRAMMATION DE LA CARTE AVEC UN PC
'******
```

Tous les paramètres utilisés par le montage peuvent être consultés et modifiés dans l'eeprom du PicBasic grâce au logiciel « ConfigES_GSM.exe », la carte est dans ce cas connectée au port série d'un PC. Deux parties de code très simples permettent la lecture et l'écriture dans l'eeprom. Notez que les données sont préservées lorsque le montage est hors tension.

' LECTURE DES PARAMETRES SITUÉS EN EEPROM

Toutes les données situées entre les adresses D_FlagPort (1FE5_{hex}) et F_Numero (1FFF_{hex}) sont envoyées au PC. Le logiciel « ConfigES_GSM.exe » se charge d'afficher clairement à l'écran les différents paramètres (voir copie d'écran).

LEC_EE:

```

j1=D_FlagPort:j2=F_Numero
GOSUB READ_EE
i=0
GOTO TEST

```

' ECRITURE DES DONNEES DE PARAMETRAGE DANS L'EEPROM

'-----

En phase d'écriture le logiciel « ConfigES_GSM.exe » envoie tous les paramètres au PicBasic qui les mémorise dans son eeprom, écrasant ainsi les anciennes valeurs.

PROG_EE:

```
SERIN RXD,BDS,0,5000,TEST,[WAIT("!!"),SMS(0)~30]
i=0
FOR j=D_FlagPort TO F_Numero
EEWRITE j,SMS(i)
i=i+1
NEXT j
i=0
GOTO TEST
```

'*****

' SOUS - PROGRAMMES

'*****

'LECTURE ENTREES DU PORT POINTE PAR SMS(2)

'-----

Le sous-programme LEC1PORT effectue la lecture d'un seul port. Le numéro du port concerné est contenu dans la variable SMS(2). Le texte du SMS est composé par le sous-programme MESS1PORT. Notez que pour les ports 4 à 7 la lecture s'effectue dans la variable tableau ValPort.

LEC1PORT:

```
IF SMS(2)<4 THEN
BYTEOUT adresse,MaskPort(SMS(2))
DATA=BYTEIN(donnee)
ELSE
n=SMS(2)-4
DATA=ValPort(n)
END IF
GOSUB MESS1PORT
RETURN
```

'COMPOSITION DU TEXTE DU SMS

'-----

Composition du SMS, on rappelle le numéro du port contenu par la donnée SMS(2), la donnée lue sur le port contenu par DATA est affichée en décimal et en binaire (état logique pour chaque entrée, voir sous-programme CONVBIN). On affiche également la valeur du Flag.

```

MESS1PORT:
    SEROUT TXD,BDS,0,1,["PORT "]
    n=SMS(2)+48
    SEROUT TXD,BDS,0,1,[n," = ",DEC(DATA,3,1)," "]
    GOSUB CONVBIN
    IF SMS(0)<>"T" THEN
        SEROUT TXD,BDS,0,1,[ "> "]
        FOR i=0 TO 7
            n=i+48
            SEROUT TXD,BDS,0,1,[ "S",n,"=",bin(i)]
            IF i<>7 THEN SEROUT TXD,BDS,0,1,[", "]
        NEXT i
    END IF
    IF SMS(2)<4 THEN SEROUT TXD,BDS,0,1,[ " (Flag=",FlagPort
                                         (SMS(2)),") "]
    RETURN

```

'PREPARE L'ENVOI DU SMS (composition du numéro de téléphone)

'-----

Le numéro de téléphone utilisé est récupéré dans la mémoire eeprom du PicBasic à partir de l'adresse D_Numero jusqu'à F_Numero.

NUM:

```

    SEROUT TXD,BDS,0,1,[ "AT+CMGS=",34]
    j1=D_Numero:j2=F_Numero
    GOSUB READ_EE
    SEROUT TXD,BDS,0,1,[34,13]
    DELAY 1000
    RETURN

```

'CONVERSION LOGIQUE DEC -> BIN -> ASCII

'-----

Sous-programme qui convertit la donnée contenue dans la variable DATA en valeurs binaires : b(0) à b(7). Notez la présence de l'instruction j<<1 qui effectue une rotation à gauche de la donnée j, ce qui est équivalent à une multiplication par 2 de j.

CONVBIN:

```

    j=1
    FOR k=0 TO 7
        n=DATA AND j
        IF n=j THEN bin(k)="1" ELSE bin(k)="0"
        j=(j<<1)
    NEXT k
    RETURN

```

'CONVERSION ANALOGIQUE DEC -> BIN -> ASCII

'-----

Ce sous-programme permet de convertir la valeur décimale fournie par l'instruction DATA=ADIN(i) en trois caractères ASCII. La formule qui permet de calculer la tension correspondante est de la forme $V = (D \times 5)/2^8$. Comme le PicBasic ne peut pas travailler avec des nombres à virgule, nous allons multiplier le résultat par 100, en simplifiant, la relation devient $V = (100 \times D)/51$. On obtient alors un nombre entier compris entre 0 et 500. Pour extraire le chiffre des unités nommé V1 on divise le résultat par 100, d'où la relation $V1 = V/100$. La première décimale nommée V2 est obtenue par la formule $V2 = (V - 100 \times V1)/10$. Enfin la deuxième décimale est obtenue par la formule $V3 = V - (100 \times V1 + 10 \times V2)$, notez qu'il est nécessaire de découper cette formule en 3 sous formules pour que le PicBasic puisse effectuer le calcul de V3. Finalement on ajoute à chacun des chiffres le nombre 48_{dec} pour obtenir le caractère ASCII correspondant.

CONVTENS:

```
V=(100*DATA)/51  
V1=V/100  
V3=100*V1  
V2=(V-V3)/10  
V3=V3+10*V2  
V3=V-V3  
V1=V1+48  
V2=V2+48  
V3=V3+48  
RETURN
```

'MASQUE UTILISE POUR MODIFIER 1 BIT

Le masque utilisé pour les opérations sur 1 seul bit est réalisé par un simple décalage à gauche d'une variable j initialisée à 1. Le nombre de décalage est défini par SMS(4) qui contient le numéro de la sortie à modifier. La boucle produit un décalage à gauche de trop, d'où la présence de la formule $j=j/2$.

UnBIT:

```
j=&b00000001  
FOR i=0 TO SMS(4)  
    j=(j<<1)  
NEXT i  
j=j/2  
n=SMS(2)-4  
RETURN
```

'LECTURE DE L'EEPROM ENTRE LES ADRESSES j1 et j2

```
READ_EE:  
FOR j=j1 TO j2  
n=EEREAD(j)  
SEROUT TXD,BDS,0,1,[n]  
NEXT  
RETURN  
  
'LECTURE DE LA DONNÉE INDEX ENTRE LES ADRESSES j1 et j2 DE  
L'EEPROM  
'-----
```

À la différence du sous-programme READ_EE, celui-ci évite d'envoyer au téléphone des caractères indésirables lorsque la donnée Index est codée sur 1 ou 2 chiffres.

```
READ_EE_Index:  
FOR j=j1 TO j2  
n=EEREAD(j)  
IF (n>=48) AND (n<=57) THEN SEROUT TXD,BDS,0,1,[n]  
NEXT  
RETURN
```

Programmation et configuration

Le programme « CES.bas » destiné au PicBasic, une fois compilé, ne fait que 1 857 octets + 27 octets de paramétrage. Sachant que l'eprom du PICBASIC-2S est de 8 Ko, le programme est à l'aise puisqu'il occupe moins de $\frac{1}{4}$ de l'espace. Cela peut sembler du luxe mais vu la simplicité du langage de programmation basic il est plus que probable que vous allez ajouter de nouvelles fonctionnalités à la carte. Voyons tout d'abord comment transférer notre programme au PicBasic. La programmation ne se fait pas via le port série, mais par le port parallèle à l'aide d'un cordon spécifique fourni par Lextronic. Si celui-ci est déjà monopolisé par l'imprimante vous pouvez utiliser le port LPT2. Si votre ordinateur dispose que d'un seul port parallèle pas de problème, il suffit de déconnecter temporairement l'imprimante. L'autre extrémité du cordon prend place dans le petit connecteur présent sur le module PicBasic, un détrompeur vous évite toute erreur de branchement. Attention, vous devez alimenter la carte une fois que le PC est allumé et que le logiciel PICBASIC-LAB est actif. Consultez la documentation fournie par Lextronic pour l'installation et l'étalonnage du logiciel PICBASIC-LAB. Copiez ensuite le programme « CES.bas » sur votre disque dur et ouvrez ce fichier à partir du logiciel PICBASIC-LAB. Cliquez sur l'icône RUN ce qui a pour effet de compiler le programme, de le transférer dans l'eprom et de l'exécuter. Attention, vous devez impérativement couper l'alimentation de la carte avant de déconnecter le PicBasic du PC.

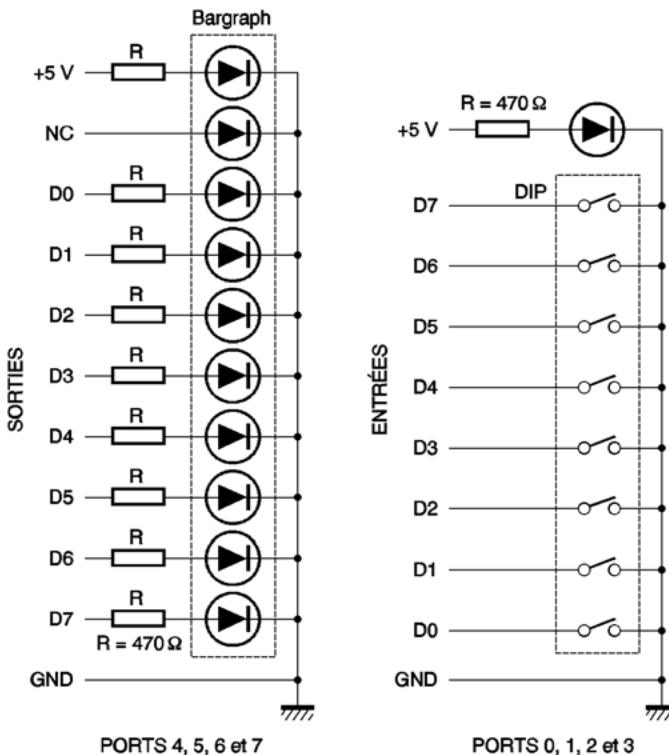
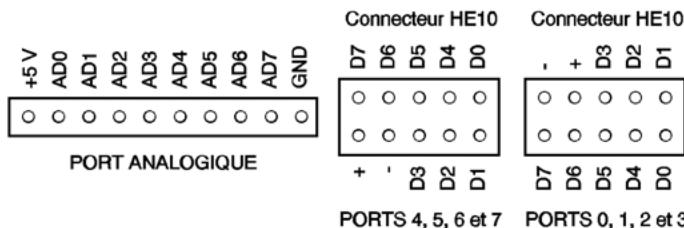
Maintenant que le PicBasic est programmé vous pouvez relier la carte au port série de votre PC pour procéder à la configuration. Les différents paramètres utiles au montage seront envoyés par l'intermédiaire du logiciel « ConfigES_GSM.exe ». Sélectionnez le port com sur lequel le montage est connecté (com2 par défaut), cliquez alors sur le bouton « Ouvrir ». Un voyant signalant que la communication est établie doit s'allumer en rouge, en fait les caractères « AT » envoyés par la carte sont bien réceptionnés par le PC. Entrez le code PIN de votre téléphone, la mémoire utilisée pour stocker les SMS réceptionnés, l'index du prochain SMS et le numéro de téléphone utilisé pour envoyer les SMS. Pour chacun des ports n° 0 à 3, il est possible de définir une valeur qui va déclencher l'envoi de SMS, à condition d'avoir coché la case « Envoi SMS ». Chaque case à cocher n° 0 à 7 correspond à une entrée. Par exemple, avec la copie d'écran ci-après, si une fois le montage connecté à un téléphone, toutes les entrées du port n° 0 sont à 1, un SMS est envoyé. Une fois tous ces paramètres définis, cliquez sur le bouton « ECRITURE » pour les transférer dans la mémoire eeprom du PicBasic. Le bouton « LECTURE » permet de s'assurer que les paramètres sont correctement mémorisés.



Figure 5.38.
Configuration
de la carte.

Essais

Pour faciliter le contrôle de bon fonctionnement des différents ports de la carte, deux montages très simples vous sont proposés. Le premier est une platine de visualisation qui comporte 8 bar-graphs constitués de 10 Led rouges rectangulaires associées à des résistances de 470 Ω qui limitent le courant. Cette platine s'enfiche directement dans les 4 ports utilisés en sortie. En effet le 74HCT541 peut débiter un courant d'environ 50 mA sur chacune de ses

Schémas platines d'essais**Brochage des ports de la carte**Figure 5.39.
Schéma
des platines d'essais.

sorties ce qui est amplement suffisant. Notez qu'une Led du bargraph est reliée au + 5 V pour signaler la présence de la tension d'alimentation, il reste une Led qui est non connectée.

Le deuxième montage permet par l'intermédiaire d'un dip switch de mettre indépendamment à la masse chacune des lignes d'un port utilisé en entrée. Rappelons qu'à l'état de repos chacun des 8 bits d'un port est mis à l'état haut par un réseau de résistance. Une Led rectangulaire associée à chacun des dip switch signale

la présence de la tension d'alimentation. À l'aide de ces deux petits montages vous pourrez valider le bon fonctionnement de chaque port de la carte, celle-ci ne comportant aucun réglage.

Pour le contrôle du port analogique il est possible de câbler un potentiomètre de $10\text{ k}\Omega$ qui viendra se connecter sur les plots de l'alimentation et sur les lignes de lecture analogique.

Interface de puissance

En bonus, nous vous proposons une interface de puissance à base de triacs qui permet à notre carte de piloter 8 charges indépendantes, alimentées sous une tension de 220 V. Cette interface se connecte sur l'un des 4 ports de sortie de la carte (ports n° 4, 5, 6 ou 7). En réalisant 4 exemplaires de cette interface vous aurez la possibilité de commander 32 charges !

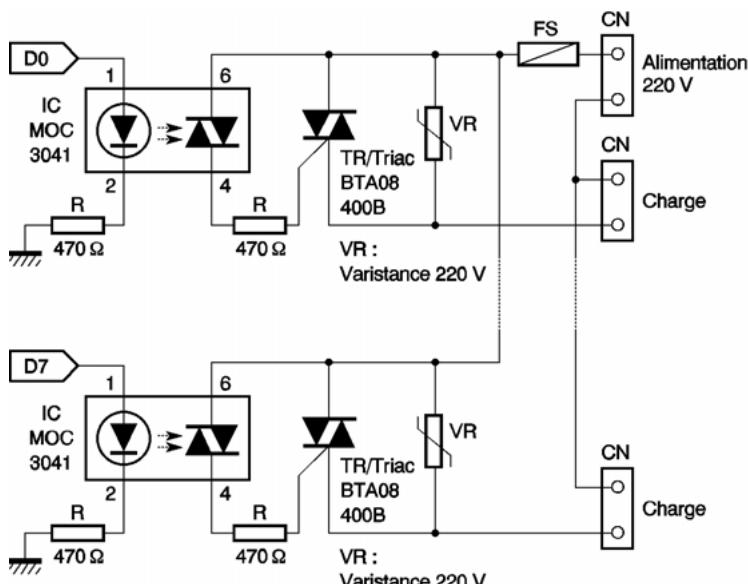
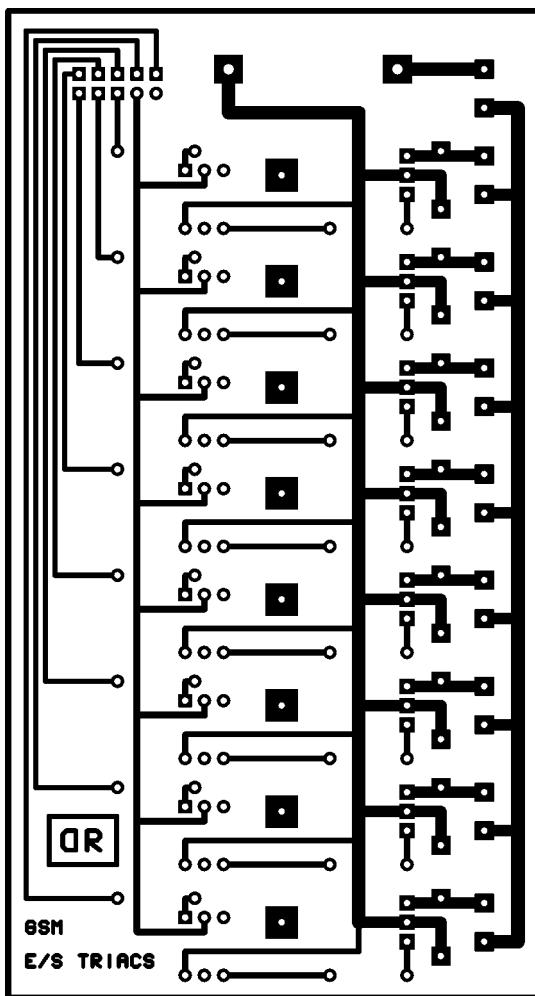


Figure 5.40.
Schéma
de l'interface
de puissance.

L'utilisation de la tension du secteur nous conduit à réaliser une isolation galvanique afin de protéger correctement l'électronique placée en amont. Cette isolation est réalisée à l'aide d'un optocoupleur (du type MOC3041), un tel circuit se compose de deux parties distinctes (isolation galvanique de 7 500 V) : la première est constituée d'une diode infrarouge qui va venir mettre en conduction le triac contenu dans la deuxième partie. Il dispose également d'un dispositif qui détecte le passage à zéro de la tension du secteur afin d'éviter de générer des parasites lors de l'alimentation de la charge. Le courant de l'ordre de 10 mA,

Figure 5.41.
Circuit imprimé.

nécessaire à l'activation de la diode infrarouge, est généré par la sortie du port, la limitation de l'intensité est assurée par une résistance de $470\ \Omega$. Outre la sécurité offerte par les optocoupleurs, leur utilisation nous permet de se passer d'une alimentation en courant continu. En effet la diode de commutation est alimentée par la sortie du port de la carte E/S et le triac par la tension secteur. Donc nul besoin de transformateur et autre régulateur de tension. La faible puissance du triac interne à l'optocoupleur ($I_{max} = 100\ mA$) ne permet pas une alimentation directe d'une charge importante. Un deuxième triac mis en cascade permet de disposer d'une puissance beaucoup plus importante. Toutefois, compte tenu de la largeur des pistes de la carte, il est conseillé de ne pas dépasser 200 W par sortie. Le composant référencé VR est

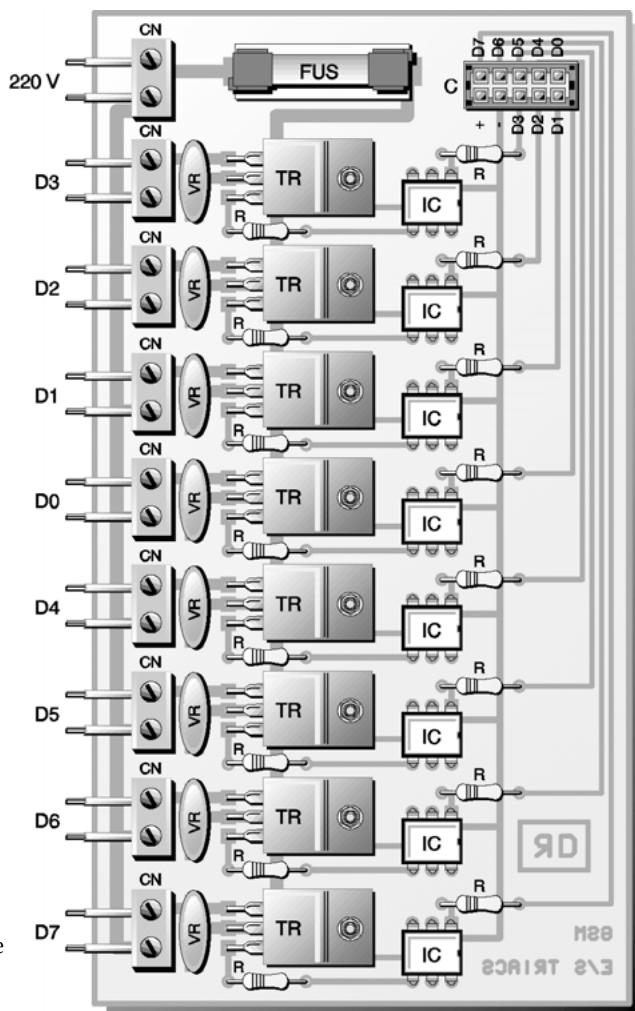


Figure 5.42.

**Implantation
des composants.**

Liste des composants

R : 16 résistances 470 Ω

IC : 8 MOC3041

ou TLP3041

TR : 8 triacs BTA08-400B

VR : 8 varistances 220 V

CN : 9 connecteurs à vis

2 bornes

FUS : porte fusible + fusible

C : connecteur HE10

mâle 10 broches

une varistance qui permet de protéger le montage lors du pilotage d'une charge inductive, les phénomènes d'auto-induction lors de l'établissement et la coupure du courant peuvent détériorer le triac. Chaque triac possède donc une varistance montée en parallèle. Ce composant voit son impédance chuter très fortement en présence d'une surtension (tension > tension nominale de 250 V), protégeant ainsi le circuit placé en aval, en l'occurrence le triac. Pour terminer, un fusible vient compléter la protection de notre montage, le calibre sera déterminé en fonction du courant maximum absorbé par les 8 charges.

Résumé des points importants

Tableau 5.30.

CARTE E/S PILOTÉE PAR GSM	
Configuration	
Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce	
Paramétrage de la carte (montage relié au port série d'un PC)	
Tous les paramètres ci-dessous peuvent être modifiés grâce au logiciel « ConfigES_GSM.exe ».	
<ul style="list-style-type: none"> • Code PIN • Mémoire utilisée pour la lecture des SMS • Index du prochain SMS • Numéro de téléphone pour l'envoi des SMS • Valeurs sur les ports 0 à 3 qui déclenchent l'envoi de SMS 	
Utilisation de la carte (montage relié à un téléphone ou terminal GSM)	
Commande SMS reçue	Action du montage
!!L, port	Lecture des 8 entrées du port concerné, avec $0 \leq \text{port} \leq 7$
!!E, port, Data	Écriture de la donnée data sur le port concerné, avec $4 \leq \text{port} \leq 7$. <i>Data doit toujours être codé sur 3 chiffres</i> , complétez par des zéros si nécessaire, ex : E,4,001
!!S, port, NumBit	Mise à 1 de la sortie NumBit du port concerné, avec $4 \leq \text{port} \leq 7$ et $0 \leq \text{NumBit} \leq 7$
!!R, port, NumBit	Mise à 0 de la sortie NumBit du port concerné, avec $4 \leq \text{port} \leq 7$ et $0 \leq \text{NumBit} \leq 7$
!!C, port, NumBit	Complément de la sortie NumBit du port concerné, avec $4 \leq \text{port} \leq 7$ et $0 \leq \text{NumBit} \leq 7$
!!T	Effectue la lecture de toutes les lignes logiques, le résultat est envoyé sous forme de 8 octets correspondants respectivement aux 8 ports de la carte
!!V	Effectue la lecture de toutes les entrées analogiques, le résultat est envoyé sous forme de 8 valeurs lues sur les entrées AD0 à AD7
!!N,06xxxxxxxx	Enregistre le numéro de téléphone indiqué dans l'eeprom du PicBasic. C'est ce numéro qui sera utilisé pour l'envoi des SMS
!!F,f0f1f2f3	Positionnement des flags. Si $f_x = 1$ le port $n^{\circ}x$ ne peut pas déclencher l'envoi d'un SMS. Si $f_x = 0$ le port $n^{\circ}x$ peut déclencher l'envoi d'un SMS. Si $f_x = ?$ le flag n'est pas modifié. Avec $0 \leq x \leq 3$

5.5 GÉOLOCALISATION PAR GSM

La plupart des terminaux GSM, tel que le TM2 de Teltonika, possèdent une commande spécifique qui permet de connaître l'identifiant de la cellule en cours d'utilisation, le fameux code Cell-ID exprimé sur 4 chiffres et codé en hexadécimal. En théorie tous les téléphones portables sont pourvus de cette fonctionnalité, car elle est prévue dans la norme GSM07.07, il s'agit de la commande « AT+CREG » détaillée dans le chapitre « Commandes AT ». Cette instruction indique si le mobile est connecté au réseau et sous certaines conditions le paramètre « Ci » équivalent au code Cell-ID.

Cell Monitor

Il est important de noter que les opérateurs gardent secret les numéros d'identification de leurs cellules. Dans un premier temps nous allons reprogrammer le montage vu précédemment dans la partie « Récepteur/émetteur SMS » afin d'afficher sur l'écran LCD les paramètres LAC et Cell-ID, ainsi il vous sera possible d'identifier et de situer géographiquement les cellules de votre environnement plus ou moins proche afin de vous constituer une petite base de données. Nous en profiterons pour afficher également la puissance du signal reçu.

Programme du PicBasic : « monitor.bas »

'DECLARATION DES CONSTANTES

'-----

Pour faciliter la maintenance du programme nous avons déclaré trois constantes : TXD qui correspond à la ligne I/O17 (broche n° 22) du PicBasic, RXD qui correspond à ligne I/O16 (broche n° 21) et BDS pour bauds qui définit la vitesse de transmission ici fixée à 9 600 car BDS = 103.

```
Const BDS = 103  
Const RXD = 17  
Const TXD = 16
```

'DECLARATION DES VARIABLES

'-----

La variable tableau Tampon qui peut contenir jusqu'à 16 octets est utilisée à diverses reprises dans le programme, notamment pour stocker les données LAC et Cell-ID en attendant leur transfert vers l'écran LCD. La variable Rssi stocke la valeur de la puissance du signal reçu par le téléphone.

```

DIM Tampon(16) AS BYTE
DIM Rssi      AS BYTE
DIM i         AS BYTE
DIM n         AS BYTE

'INITIALISATION DE L'ECRAN LCD
'-----

```

Il convient d'initialiser l'écran LCD connecté au PicBasic grâce aux instructions spécifiques à ce type d'afficheur. L'instruction SET PICBUS HIGH ou LOW permet de paramétriser la vitesse de communication du bus spécialisé « PICBUS ». Par défaut, ce type d'afficheur est configuré pour travailler à une vitesse de 19 200 bauds donc l'instruction SET PICBUS sera suivie de l'instruction HIGH (LOW pour une vitesse de 4 800 bauds). L'instruction LCDINIT initialise l'écran LCD.

```

SET PICBUS HIGH
LCDINIT

```

```

'TEST LIAISON SERIE
'-----

```

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit : AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte. Les caractères « AT » suivis du caractère <CR>=13_{dec} sont envoyés par la commande SEROUT. L'instruction SERIN permet d'attendre l'éventuelle réponse « OK » pendant 2 000 ms (soit 2 s). Si les caractères OK sont réceptionnés dans le temps donné, le caractère suivant soit <CR> est placé dans la variable i. Dans le cas contraire le programme saute à la ligne repérée par l'étiquette TEST0, i est alors vide. Il suffit de tester le contenu de i pour savoir si la liaison est établie.

```

i=0
TEST0: SEROUT TXD,BDS,0,1,["AT",13]
        SERIN RXD,BDS,0,2000,TEST1,[WAIT("OK"),i]
TEST1: IF i<>0 THEN
        LOCATE 0,0
        PRINT "Liaison OK"
        LOCATE 0,1
        PRINT "Test mode ..."
ELSE
        LOCATE 0,0
        PRINT "PB liaison !"
        DELAY 5000
END IF

```

Tant que la liaison n'est pas établie le programme boucle sur l'étiquette TEST0. L'écran LCD affiche le message « PB liaison ! ». Une fois la liaison établie le programme suit son cours normal.

```
IF i=0 THEN GOTO TEST0

'SELECTION DE L'ALPHABET GSM
'-----
SEROUT TXD,BDS,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500

'CODE PIN
'-----
```

En principe le code PIN qui autorise l'utilisation du téléphone doit être composé à chaque mise sous tension.

```
SEROUT TXD,BDS,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500
```

```
'MESSAGE LCD D'ATTENTE
'-----
```

À l'initialisation du montage un message d'accueil s'affiche sur l'écran LCD. Les paramètres LAC et Cell-ID seront affichés lors du changement de cellule.

```
CLS
LOCATE 0,0
PRINT "Acquisition"
LOCATE 0,1
PRINT "en cours..."
```

```
'INITIALISATION DES VARIABLES
'-----
```

Initialisation de la variable Tampon grâce à une boucle FOR/NEXT.

```
DEBUT: FOR i=0 TO 15
    Tampon(i)=0
NEXT i
```

```
'PROGRAMME PRINCIPAL
'-----
```

Voici la partie du code qui permet de connaître les champs LAC et Cell-ID. L'instruction « AT+CREG=2 » lance la procédure d'acquisition, à chaque changement de cellule le téléphone envoie au PicBasic les nouvelles données <lac> et <ci> qui identifient la BTS en cours d'utilisation. Chaque envoi est précédé par les

caractères « +CREG », le PicBasic se contente de détecter la paire de caractères « EG » et de placer les 16 caractères suivants dans la variable Tampon.

```
SEROUT TXD,BDS,0,1,[ "AT+CREG=2" ]
DELAY 500
SEROUT TXD,BDS,0,1,[13]
ATT1: SERIN RXD,BDS,0,10000,ATT1,[WAIT("EG"),Tampon(0)~16]
SEROUT TXD,BDS,0,1,[ "AT+CREG=0" ,13]
```

Un exemple de ce que peut contenir la variable Tampon est indiqué **tableau 5.31**.

Tableau 5.31.

Tampon(0)	Tampon(1)	Tampon(2)	Tampon(3)	Tampon(4)	Tampon(5)	Tampon(6)	Tampon(7)	Tampon(8)
:		1	,	"	0	B	D	C

Tampon(2) contient la donnée <stat> qui représente l'état d'enregistrement du téléphone sur le réseau :

<stat> :

- 0 téléphone non enregistré, pas de recherche d'opérateur en cours
- 1 téléphone enregistré sur le réseau
- 2 téléphone non enregistré, recherche d'opérateur en cours
- 6 enregistrement interdit
- 7 inconnu
- 8 enregistré, *roaming*

Tampon(5) à Tampon(8) contiennent les 4 octets codifiant la donnée <lac>.

Tableau 5.32.

Tampon(9)	Tampon(10)	Tampon(11)	Tampon(12)	Tampon(13)	Tampon(14)	Tampon(15)
"	,	"	F	A	0	9

Tampon(12) à Tampon(15) contiennent les 4 octets codifiant la donnée <ci>.

Il ne reste qu'à afficher sur l'écran LCD les données Stat, LAC et Cell-ID. Le buzzer est activé pour signaler le changement de cellule. La donnée Stat est affichée entre crochets sur la première ligne, les deux autres paramètres apparaissent sur la deuxième ligne.

```
BEEP 9
BEEP 9
CLS
LOCATE 0,0
PRINT "[",Tampon(2),"]"
LOCATE 0,1
PRINT "LAC:",Tampon(5) ,Tampon(6) ,Tampon(7) ,Tampon(8)
PRINT " CI:",Tampon(12),Tampon(13),Tampon(14),Tampon(15)
```

Maintenant le PicBasic interroge le téléphone pour connaître la qualité du signal à l'aide de la commande « AT+CSQ ». Le téléphone répond au montage en débutant par les caractères « +CSQ » suivis du paramètre Rssi qui indique la puissance du signal. Dès la réception du couple de caractères « SQ » le PicBasic mémorise les 4 caractères suivants dans la variable Tampon.

```
ATT2: SEROUT TXD,BDS,0,1,[ "AT+CSQ"]
DELAY 500
SEROUT TXD,BDS,0,1,[13]
SERIN RXD,BDS,0,1000,ATT2,[WAIT("SQ"),Tampon(0)~4]
```

Un exemple de ce que peut contenir la variable Tampon est indiqué **tableau 5.33.**

Tableau 5.33.

Tampon(0)	Tampon(1)	Tampon(2)	Tampon(3)
:		2	0

Attention la valeur Rssi est comprise entre 0 et 31, le programme doit avant de convertir cette valeur en décimal, vérifier si elle est codée sur un ou deux chiffres. Si la valeur Rssi est codée sur 2 chiffres, Tampon(2) contient le chiffre des dizaines et Tampon(3) le chiffre des unités. Dans le cas contraire, Tampon(2) contient le chiffre des unités, Tampon(3) contient alors une virgule qui est ignorée par le programme. Si la valeur Rssi est supérieure à 31 cela signifie que la puissance du signal reçu n'est pas quantifiable un "?" est dans ce cas mémorisé dans la variable Rssi. Dans le cas contraire le programme calcule la puissance exprimée en dBm. Notez que les membres de l'équation sont inversés afin d'obtenir un résultat positif. Le signe moins sera ajouté au moment de l'affichage de la valeur sur l'écran LCD.

```
IF Tampon(3)>=48 AND Tampon(3)<=57 THEN
    Tampon(2)=Tampon(2)-48
    Tampon(2)=Tampon(2)*10
    Tampon(3)=Tampon(3)-48
    Rssi=Tampon(2)+Tampon(3)
ELSE
```

```
Tampon(2)=Tampon(2)-48
Rssi=Tampon(2)
END IF
IF Rssi<=31 THEN
    Rssi=113-(2*Rssi)
ELSE
    Rssi="?"
END IF
```

Affichage du résultat sur la première ligne de l'écran LCD, à droite de la donnée Stat.

```
LOCATE 5,0
PRINT "P= -",DEC(Rssi,3,1),"dBm "
GOTO DEBUT
```

Résumé des points importants

Tableau 5.34.

CELL MONITOR	
Configuration	Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce
Éléments du programme PicBasic à modifier	
• Code PIN (7208 par défaut)	

Tracker GPS

Nous vous proposons la réalisation d'un montage capable d'envoyer par SMS sa propre position géographique. La grande précision est due à la mise en œuvre d'un récepteur GPS miniature. Une fois les coordonnées longitude et latitude rentrées dans les applications telles que Google Map ou Google Earth, vous localiserez précisément la position de votre montage sur une carte et/ou une photo satellite.

Positionnement géographique

Déterminer une position géographique sur notre bonne vieille planète revient à déterminer les coordonnées d'un point situé à la surface d'une sphère. Qui dit coordonnées dit repère cartésien constitué de deux plans perpendiculaires. Le premier passe par les pôles Nord et Sud et par l'observatoire de Greenwich près de Londres : on l'appelle le méridien d'origine. Le deuxième est à équidistance des pôles Nord et Sud : il s'agit de l'Equateur. Positionner un point sur la Terre revient à déterminer la distance qui le sépare du méridien d'origine : c'est la longitude, et la distance

qui le sépare de l'Équateur : c'est la latitude. Ces distances se mesurent en degrés d'angle avec une étendue de -180° (Ouest et Nord) à $+180^{\circ}$ (Est et Sud).

Opérationnel depuis 1995, le GPS (*Global Positioning System*) permet à un récepteur utilisant cette technologie de déterminer précisément sa position géographique. Un tel récepteur est susceptible d'utiliser les quelque 24 satellites en orbite autour de la Terre. Chaque satellite émet en permanence un signal radio contenant diverses informations, dont un signal horaire, l'heure d'émission du signal et la position du satellite. Afin de se situer le récepteur GPS mesure le temps que met ce signal à lui parvenir. En effectuant cette opération avec au moins trois satellites simultanément il est capable de calculer les données longitude et latitude.

Récepteur GPS

Le récepteur mis en œuvre dans cette réalisation est un GPS OEM subminiature référencé EM-406 ; il est distribué en France par la société Lextronic (<http://www.lextronic.fr>) pour un prix abordable même pour l'électronicien amateur (au sens noble du terme).

Ce GPS est basé sur le chipset SiRF StarIII™, un des plus apprécié du marché, ce qui lui confère une stabilité exceptionnelle. Il est capable d'utiliser jusqu'à 20 satellites, ce qui lui permet d'avoir un bon temps de réponse et une précision correcte.

Un des avantages de ce modèle est qu'il dispose d'une antenne de réception intégrée GPS USGlobalSat, ce qui permet la réalisation d'un montage compacte et discret.

Principales caractéristiques du récepteur EM-406

Dimensions 30 X 30 X 10,5 mm (antenne incluse)
Alimentation +4,5 à +6,5 Vcc
Consommation 70 mA
Canaux 20
Position 10 m, 2D RMS
Vélocité 515 m/s
Altitude maxi. 18 000 m
Accélération < 4 g
Temps de ré acquisition 0,1 s
Hot Start 1 s
Warm Start 38 s
Cold Start 42 s

Le récepteur délivre une fois par seconde des trames conformes à la norme NMEA0183 (National Marine Electronics Associa-

tion). Les trames sont composées de caractères ASCII qui transittent sous forme série sur la sortie nommée TX. Le protocole utilisé est conforme à la norme RS232 avec une vitesse prédéfinie de 4 800 bits/s.

Chaque trame débute par les caractères \$GP. Trois caractères supplémentaires identifient le type de trame envoyée. L'EM-406 est capable de générer 6 sortes de trames :

Nom	Exemple	Unité	Description
Message ID	\$GPGGA		GGA protocol header
UTC Time	161229.487		hhmmss.sss
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Position Fix Indicator	I		0 : fix not available or invalid 1 : GPS SPS Mode, fix valid 2 : Diff. GPS, SPS Mode , fix valid 3 : GPS PPS Mode, fix valid
Satellites Used	07		Range 0 to 12
HDOP	1.0		Horizontal Dilution of Precision
MSL Altitude1	9.0	mètre	
Units	M	mètre	
Geoid Separation1		mètre	
Units	M	mètre	
Age of Diff. Corr.		seconde	Null fields when DGPS is not used
Diff. Ref. Station ID	0000		
Checksum	*18		
<CR><LF>			End of message termination

Tableau 5.35.
GGA : GPS fix et date

INTERFACES GSM

Tableau 5.36.
GLL : position géographique longitude-latitude.

Nom	Exemple	Unité	Description
Message ID	\$GPGLL		GLL protocol header
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
UTC Time	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Checksum	*2C		
<CR><LF>			End of message termination

Tableau 5.37.
GSA : satellites actifs.

Nom	Exemple	Unité	Description
Message ID	\$GPGSA		GSA protocol header
Mode 1	A		M : Manual-forced to operate in 2D or 3D mode A : 2Dautomatic-allowed to automatically switch 2D/3D
Mode 2	3		1 : Fix not Available 2 : 2D 3 : 3D
Satellite used	07		Sv on Channel 1
Satellite used	02		Sv on Channel 2
...			
Satellite used			Sv on Channel 12
PDOP	1.8		Position dilution of Precision
HDOP	1.0		Horizontal dilution of Precision
VDOP	1.5		Vertical dilution of Precision
Checksum	*33		
<CR><LF>			End of message termination

Dans le cadre de notre application nous utiliserons exclusivement la trame RMC qui nous donne les informations nécessaires et suffisantes compte tenu de notre application : la longitude et la latitude.

Nom	Exemple	Unité	Description
Message ID	\$GP GSV		GSA protocol header
Number of Messages	2		Range 1 to 3
Message Number1	1		Range 1 to 3
Satellites in View	07		
Satellite ID	07		Channel 1 (Range 1 to 32)
Elevation	79	degré	Channel 1 (Maximum 90)
Azimuth	048	degré	Channel 1 (True. Range 0 to 359)
SNR(C/No)	42	dBHz	Range 0 to 99. Null when not tracking
...			
Satellite ID	27	degré	Channel 4
Elevation	27	degré	Channel 4
Azimuth	138	degré	Channel 4
SNR(C/No)	42	dBHz	
Checksum	*33		
<CR><LF>			End of message termination

Tableau 5.38.
GSV : satellites visibles.

■ Exemple de trame RMC :

\$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598,E,*10

Les champs sont séparés par des virgules. Un checksum optionnel peut être présent à la fin de la trame (non utilisé ici), il est précédé du caractère *. Chaque trame se termine par les caractères <CR><LF> (retour chariot, retour à la ligne).

Schéma électrique

La broche I/O11 est dédiée à la récupération des trames émises sur la sortie TX du GPS. Toutefois les niveaux de tension entre ces deux broches ne sont malheureusement pas compatibles. Le PicBasic attend des niveaux de 0v ou +5v alors que le GPS délivre des niveaux compris entre 0v et 3,3v. Plutôt que de mettre en oeuvre un circuit intégré pour adapter les niveaux de tension, nous avons choisi la simplicité en utilisant deux transistors BC547 travaillant en bloqué ou saturé. Lorsque la base de T1 est soumise à une tension de 3,3v on relève sur le collecteur de T2 une tension de +5v.

INTERFACES GSM

Tableau 5.38.
RMC : données minimales exploitables specifications spécifiques.

Nom	Exemple	Unité	Description
Message ID	\$GPRMC		RMC protocol header
UTC Time	161229.487		hhmmss.sss
Status	A		A=data valid or V=data not valid
Latitude	3723.2475		ddmm.mmmm
N/S Indicator	N		N=north or S=south
Longitude	12158.3416		dddmm.mmmm
E/W Indicator	W		E=east or W=west
Speed Over Ground	0.13	noeud	
Course Over Ground	309.62	degré	True
Date	120598		ddmmyy
Magnetic Variation	E	degré	E=east or W=west
Checksum	*10		
<CR><LF>			End of message termination

Tableau 5.39.
VTG : direction et vitesse de déplacement.

Nom	Exemple	Unité	Description
Message ID	\$GPVTG		VTG protocol header
Course	309.62	degré	Measured heading
Reference	T		True
Course		degré	Measured heading
Reference	M		Magnetic
Speed	0.13	noeud	Measured horizontal speed
Units	N		Knots
Speed	0.2	km/h	Measured horizontal speed
Units	K		Kilometers per hour
Checksum	*6E		
<CR><LF>			End of message termination

Particularité concernant la réalisation

Le récepteur GPS est livré avec un petit câble muni de deux connecteurs. Sectionnez le câble afin d'éliminer un connecteur et remplacez-le par un morceau de barrette sécable 6 points. Ainsi il est facile de le relier au connecteur CN5 du montage. Attention à l'orientation du connecteur car celui-ci n'a plus de détrompeur – se repérer par rapport au conducteur de couleur grise.

Programme du picbasic « trackerGPS.bas »

```
'DECLARATION DES CONSTANTES
```

Constantes utilisées pour la liaison série avec le téléphone GSM :

```
Const BDS_GSM = 103  
Const RXD_GSM = 17
```

Constantes utilisées pour la liaison série avec le GPS. La vitesse de communication est de 4800bds. Pas de constante concernant une sortie TXD puisque le PicBasic se contente de recevoir des données en provenance du GPS.

```
Const BDS_GPS = 207  
Const RXD_GPS = 11  
  
Const BUZZER = 10
```

```
'DECLARATION DES CONSTANTES
```

Nous verrons l'utilité des variables ci-dessous au fur et à mesure du détail du programme.

```
DIM Tampon(43) AS BYTE  
DIM Index(3) AS BYTE  
Const TXD_GSM = 16  
DIM j AS INTEGER  
M num AS BYTE
```

```
'TEST LIAISON SERIE
```

Pour s'assurer que la liaison entre le montage et le téléphone est valide, nous allons envoyer la commande la plus simple qui soit, AT<CR>, le ME doit répondre par <CR><LF>OK<CR><LF> si la liaison est correcte.

```
DEBUT2:
```

```
i=0
```

```
TEST:
```

```
SEROUT TXD_GSM,BDS_GSM,0,1,["AT",13]
```

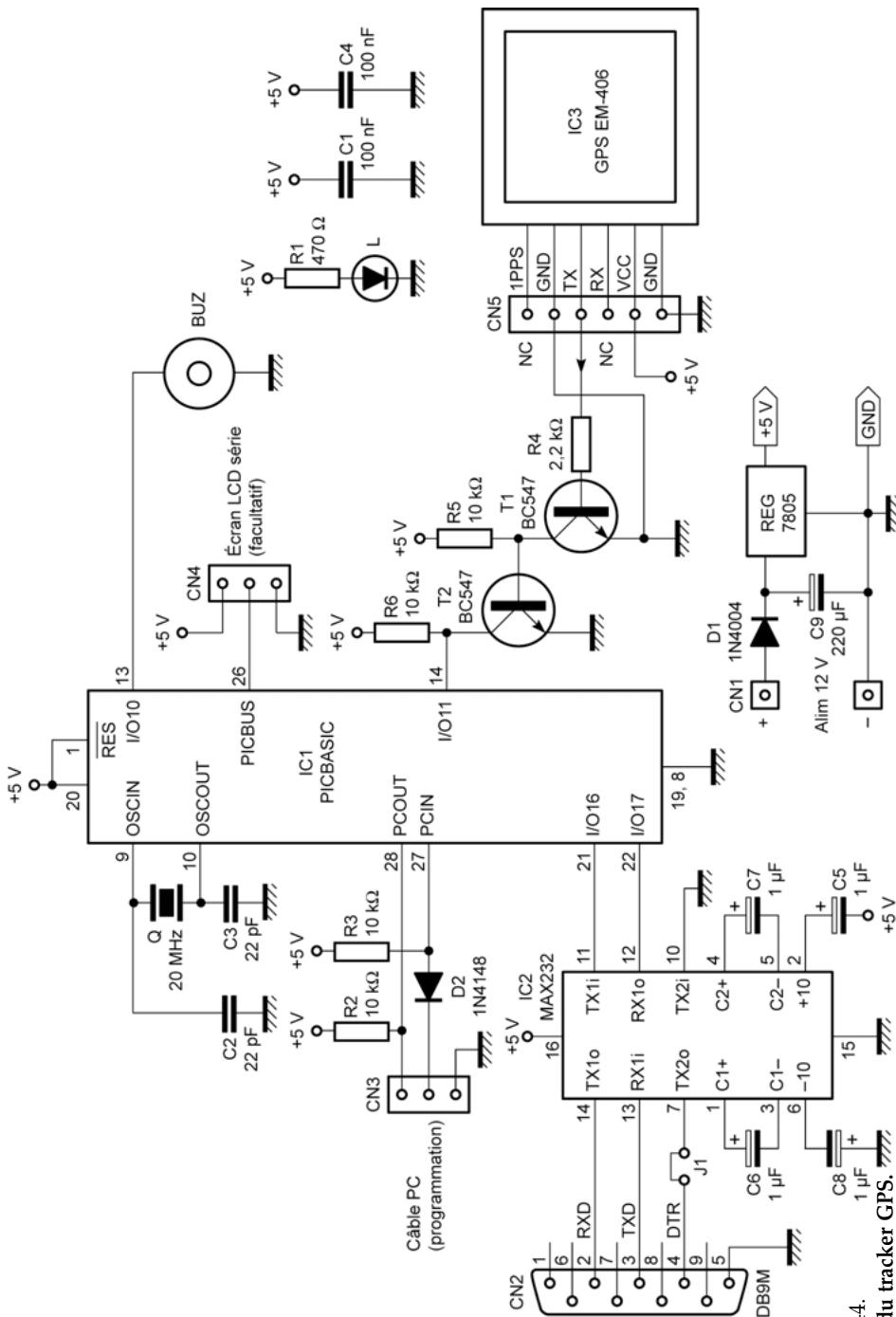


Figure 5.44.
Schéma du tracker GPS.

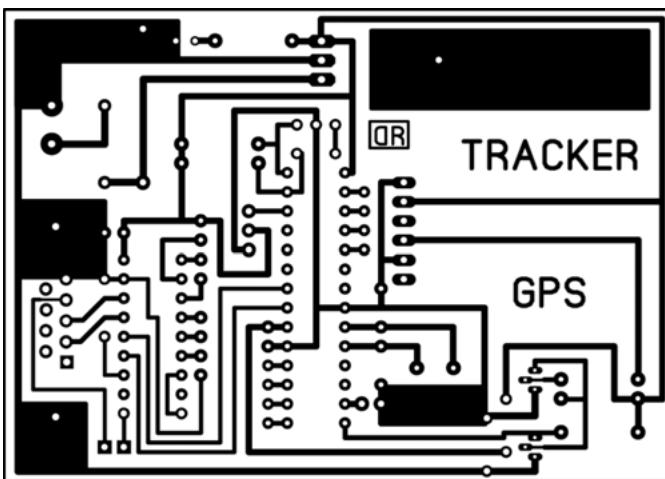


Figure 5.45.
Circuit imprimé.

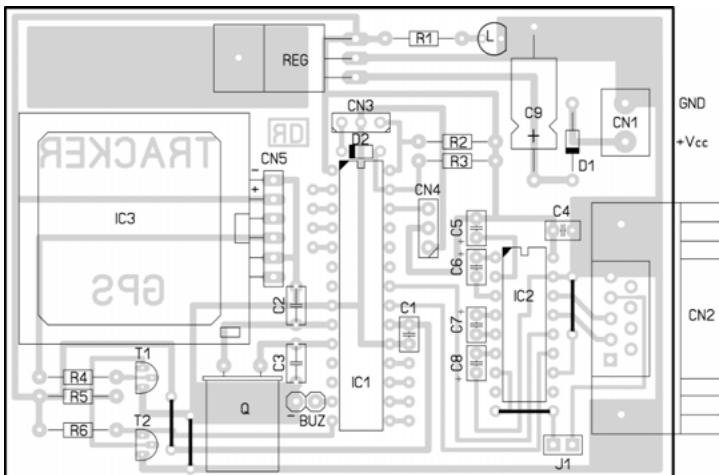


Figure 5.46.
Implantation
des composants.

Liste des composants

R1 : 470 Ω

R2, R3, R5, R6 : 10k Ω

R4 : 2.2k Ω

C1-C4 : 100nF (pas de 2.54mm)

C2 C3 : 22nF / céramique

C5 C6 C7 C8 : 1uE / tantale / 15v

C₆, C₈, C₇, C₉: fer / tantale / Ti
C₉: 220μF / électrolytique / 15V

D1 : diode 1N4004

D1 : diode IN4004
D2 : diode 1N4148

D₂ diode IN4148
L : Lead standard

L : Led standard
Q : quartz 20ML

Q:quartz 20MHz
BEC:réglateur 7805

BUZ : buzzer piezzo (sans électronique intégrée)

J1 : barrette HE10 2 contacts + cavalier

CN1 : bornier à vis 2 plots

CN2 : connecteur DB9 mâle pour CI / coudée à 90°

CN3 : connected

CNS : connecteur pour cable de programmation (LEXTRONIC)

CN4 : connecteur pour écran LCD (L)

CN5 : barrette sécable tu

T1, T2 : transistor BC547

IC1 : PicBasic PB 3B (LEXTRONIC) + support 28

ICP. The
breaks

IC2 : MAX232 + support DIL 16 broches

IC2 : MAX32 + support DIL 16 broches

INTERFACES GSM

```
SERIN RXD_GSM,BDS_GSM,0,2000,TEST1,[WAIT("OK"),i]
TEST1:
    IF i=0 THEN GOTO TEST
    BEEP BUZZER
    DELAY 1000
    BEEP BUZZER

'SELECTION DE L'ALPHABET GSM
'-----
```

```
SEROUT TXD_GSM,BDS_GSM,0,1,[ "AT+CSCS=",34,"GSM",34,13]
DELAY 500
```

```
'CODE PIN
'-----
```

■ N'oubliez pas de modifier le code pin par défaut égale à 7208 :

```
SEROUT TXD_GSM,BDS_GSM,0,1,[ "AT+CPIN=",34,"7208",34,13]
DELAY 500
```

```
'INITIALISATION DU ME
'-----
```

■ Le ME est configuré en mode TEXT par la commande AT+CMGF=1. La commande AT+CNMI=1,1 indique au ME que chaque nouveau SMS reçu doit être signalé au TE. Ainsi l'arrivée d'un SMS sera signalée par l'envoi au ME de la commande +CMTI: <mem1>,<index> :

```
SEROUT TXD_GSM,BDS_GSM,0,1,[ "AT+CMGF=1",13]
DELAY 500
SEROUT TXD_GSM,BDS_GSM,0,1,[ "AT+CNMI=1,1",13]
DELAY 500
```

```
'INITIALISATION N° TELEPHONE UTILISE
PAR DEFAUT POUR L'ENVOI DES SMS
'-----
```

■ Le numéro de téléphone utilisé par défaut pour envoyer des SMS est initialisé dans la mémoire EEPROM du PicBasic. Pour ne pas interférer avec la partie programme, le stockage se fait dans les 10 derniers emplacements de la mémoire de FF6hex à FFFhex. Cette mémorisation ne se réalise qu'une seule fois car le programme teste avant si l'adresse FF6hex est vide (notez qu'un emplacement vide contient la donnée FFhex).

```
IF EEREAD(&HFF6)=&HFF THEN
    EEWRITE &HFF6,"0"
    EEWRITE &HFF7,"6"
    EEWRITE &HFF8,"x"
    EEWRITE &HFF9,"x"
    EEWRITE &HFFA,"x"
    EEWRITE &HFFB,"x"
```

```

EEWRITE &HFFD,"x"
EEWRITE &HFFE,"x"
EEWRITE &HFFF,"x"
END IF

'INITIALISATION DES VARIABLES
'-----

```

Initialisation des différentes variables utilisées par le programme :

```

DEBUT:
FOR i=0 TO 42
    Tampon[i]=0
NEXT i

```

```
'ATTENTE RECEPTION SMS
'-----
```

Désormais le uC scrute l'entrée RXD dans l'attente des caractères « TI ». Dès leur réception les 10 caractères suivants sont placés dans la variable Tampon. Une série de 11 bips signale l'arrivée du SMS :

```

ATT_SMS:
SERIN RXD_GSM,BDS_GSM,0,10000,ATT_SMS, [WAIT("TI"),Tampon(0)~10]
FOR i=0 TO 10
    BEEP BUZZER
    DELAY 500
NEXT i

```

Comme l'index s'incrémente à chaque nouveau message reçu, il est nécessaire d'extraire cette donnée pour savoir où aller lire le message en mémoire. Dans l'état actuel des choses admettons que la donnée Tampon contient ceci :

On considère dans le programme que la valeur <index> sera codée au maximum sur 3 chiffres. Il est possible, comme le montre cet exemple, que le stockage des SMS se fasse dans la mémoire ME à partir de l'index 900 :

```
'LECTURE DU SMS RECU
'-----
```

Le TE configure le ME pour que la lecture soit faite dans la mémoire définie par Tampon(3) et Tampon(4) :

Tableau 5.40.

Tampon (0)	Tampon (1)	Tampon (2)	Tampon (3)	Tampon (4)	Tampon (5)	Tampon (6)	Tampon (7)	Tampon (8)	Tampon (9)
:		"	M	E	"	,	9	0	0

INTERFACES GSM

```
SEROUT TXD_GSM,BDS_GSM,0,1,  
["AT+CPMS=",34,Tampon(3),Tampon(4),34,13]  
DELAY 500
```

La lecture du SMS est provoquée par la commande « AT+CMGR=<index> ». Au préalable nous stockons la donnée index dans la variable tableau du même nom. Si la donnée <index> est codée sur un ou deux chiffres, on récupère des données indésirables (<CR><LF>). Pour les éliminer lors de la reconstitution de l'index du message on s'assure que les données Index(0) à Index(1) contiennent un caractère compris entre $0_{\text{ASCII}}=48_{\text{dec}}$ et $9_{\text{ASCII}}=57_{\text{dec}}$:

```
Index(0)=Tampon(7)  
Index(1)=Tampon(8)  
Index(2)=Tampon(9)  
  
SEROUT TXD_GSM,BDS_GSM,0,1,["AT+CMGR="]  
FOR i=0 TO 2  
IF Index(i)>=48 AND Index(i)<=57 THEN SEROUT  
TXD_GSM,BDS_GSM,0,1,[Index(i)]  
NEXT i  
SEROUT TXD_GSM,BDS_GSM,0,1,[13]
```

Dès la réception des caractères !!, les 7 caractères suivants sont placés dans la variable Tampon. Dans le cas où les caractères !! ne sont pas détectés dans les 5 secondes, le programme passe au label SUITE, comme la variable Tampon(0) est vide, le SMS est effacé. Si le SMS contient moins de 7 caractères, le programme passe également au label SUITE mais comme Tampon(0) est dans ce cas différent de 0, le programme suit son cours :

```
SERIN RXD_GSM,BDS_GSM,0,5000,SUITE,[WAIT("!!"),Tampon(0)~13]  
SUITE:  
IF Tampon(0)=0 THEN GOTO RAZ
```

En l'état actuel du programme, si l'on considère que le SMS envoyé était de la forme !!G?, la variable tableau SMS doit contenir au minimum ceci :

Tableau 5.41.

Tampon (0)	Tampon (1)
G	?

```
IF Tampon(0)<>"G" OR Tampon(1)<>"?" THEN GOTO DEBUT  
'PREPARATION ENVOI SMS : N° TEL DESTINATAIRE  
'-----
```

Le numéro utilisé est celui situé dans la mémoire EEPROM du PicBasic, aux adresses FF6hex à FFFhex ou le cas échéant celui précisé dans le corps du SMS reçu :

Tableau 5.42.

Tampon	2	3	4	5	6	7	8	9	10	11	12
Donnée	,	0	6	x	x	x	x	x	x	x	x

```
SEROUT TXD_GSM,BDS_GSM,0,1,[ "AT+CMGS=",34]
IF Tampon(2)=".," THEN
  FOR i=3 TO 12
    SEROUT TXD_GSM,BDS_GSM,0,1,[Tampon(i)]
  NEXT i
ELSE
  FOR j=&HFF6 TO &HFFF
    num=EEREAD(j)
    SEROUT TXD_GSM,BDS_GSM,0,1,[num]
  NEXT j
END IF
SEROUT TXD_GSM,BDS_GSM,0,1,[34,13]
```

'ACQUISITION TRAME NMEA

'-----

Attente de l'acquisition d'une trame de type \$GPRMC dans la variable tableau Tampon :

```
ATT_GPS:
  SERIN RXD_GPS,BDS_GPS,0,1000, ATT_GPS,
  [WAIT("$G"), Tampon(0)~42]
```

Comme l'instruction WAIT n'admet que 2 caractères en argument, il nous faut ruser et contrôler les caractères suivants afin d'exclure les trames inutilisées :

```
IF Tampon(0)<>"P" THEN GOTO ATT_GPS
IF Tampon(1)<>"R" THEN GOTO ATT_GPS
IF Tampon(2)<>"M" THEN GOTO ATT_GPS
IF Tampon(3)<>"C" THEN GOTO ATT_GPS
```

Ainsi, lorsque les 4 premiers caractères mémorisés dans la variable Tampon sont égaux à "PRMC", le programme peut alors continuer afin d'extraire les fameuses données latitude et longitude.

Exemple de contenu de la variable tableau Tampon lorsque la trame réceptionnée est de type "RMC" :

Seuls les 42 premiers caractères de la trame sont mémorisés car ils contiennent les informations utiles à notre montage. Les différentes informations sont extraites en accédant simplement aux données

INTERFACES GSM

Tampon	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Donnée	P	R	M	C	,	1	6	1	2	2	9	.	4	8	7	,
Type	Message ID				UTC Time											

15	16	17	18	19	20	21	22	23	24	25	26	27	28	29		
,	A	,	3	7	2	3	.	2	4	7	5	,	N	,		
Status		Latitude														N/S

30	31	32	33	34	35	36	37	38	39	40	41	42		
1	2	1	5	8	.	3	4	1	6	,	W	,		
Longitude													E/W	

Tableau 5.43.

contenues dans la variable tableau Tampon, chaque caractère est accessible via son index de 0 à 41. La première information extraite à l'index 16 est le status car il signale si la trame est valide par la lettre "A". Une trame non valide est identifiée par le status "V", cela se produit lorsque le récepteur n'a pas assez de satellites en vue (on appelle cela le fix) pour déterminer ses coordonnées. Dans ce cas de figure, la trame est ignorée et le montage attend la prochaine trame. Le programme est dirigé vers le label ATT_GPS :

```
IF Tampon(16)<>"A" THEN GOTO ATT_GPS
```

```
'ENVOI SMS
'-----
```

Le message expédié sur le réseau GSM contient les données heure, latitude et longitude fournies par le GPS :

```
SEROUT TXD_GSM,BDS_GSM,0,1,[ "RESULTAT GEOLOCALISATION",13]
```

Une boucle FOR parcourt les index 5 à 10 qui contiennent l'heure UTC :

```
SEROUT TXD_GSM,BDS_GSM,0,1,[ "Heure UTC : "]
FOR i=5 TO 10
    SEROUT TXD_GSM,BDS_GSM,0,1,[ Tampon(i)]
    IF i=6 OR i=8 THEN SEROUT TXD_GSM,BDS_GSM,0,1,[ ":"]
NEXT i
```

```
SEROUT TXD_GSM,BDS_GSM,0,1,[13,"Latitude : "]
```

L'index 28 contient l'indicateur Nord/Sud qui est transformé en signe positif ou négatif avant d'être envoyé :

```
IF Tampon(28)="N" THEN
    SEROUT TXD_GSM,BDS_GSM,0,1,[ "+"]
ELSE
    SEROUT TXD_GSM,BDS_GSM,0,1,[ "-"]
END IF
```

Une boucle FOR parcourt les index 18 à 26 qui contiennent la donnée Latitude, ainsi que le caractère de séparation (virgule) en position 27 :

```
FOR i=18 TO 26
    SEROUT TXD_GSM,BDS_GSM,0,1,[ Tampon(i)]
NEXT i

SEROUT TXD_GSM,BDS_GSM,0,1,[13,"Longitude : "]
```

L'index 41 est l'indicateur Est/Ouest (East/West) transformé en signe positif ou négatif :

```
IF Tampon(41)="E" THEN
    SEROUT TXD_GSM,BDS_GSM,0,1,[ "+"]
ELSE
    SEROUT TXD_GSM,BDS_GSM,0,1,[ "-"]
END IF
```

Une boucle FOR parcourt les index 30 à 39 qui contiennent la donnée Longitude :

```
FOR i=30 TO 39
    SEROUT TXD_GSM,BDS_GSM,0,1,[ Tampon(i)]
NEXT i

SEROUT TXD_GSM,BDS_GSM,0,1,[26]

'EFFECTUE LE SMS EN MEMOIRE
-----
```

Cette partie du programme permet de systématiquement effacer le SMS en mémoire, ainsi le prochain SMS réceptionné aura toujours le même index. Ceci évite de saturer la mémoire utilisée. Notez que cette partie de programme est appelée même si aucun SMS n'est à effacer ; dans ce cas le ME répond par un message d'erreur qui est ignoré par le programme :

```
RAZ:
SEROUT TXD_GSM,BDS_GSM,0,1,[ "AT+CMGD=" ]
FOR i=0 TO 2
    IF Index(i)>=48 AND Index(i)<=57 THEN SEROUT
        TXD_GSM,BDS_GSM,0,1,[Index(i)]
NEXT i
```

```
SEROUT TXD_GSM,BDS_GSM,0,1,[13]  
DELAY 5000  
  
GOTO DEBUT
```

Interprétation du résultat

Les coordonnées sont fournies en représentation sexagésimale. Il est nécessaire de les convertir en décimal afin de pouvoir les utiliser dans Google Map et Google Earth :

Latitude : ddmm.mmmm = dd + mm.mmmm/60
Longitude : dddmm.mmmm = ddd + mm.mmmm/60

Exemple (il s'agit des coordonnées extraites de la datasheet du EM-406) :

Latitude : = +3723,2475 = + (37+23,2475/60) = +37,387458
Longitude = -12158,3416 = -(121+58,3416/60) = -121,97236

Ensuite, à l'aide de votre navigateur préféré, il suffit de vous rendre à l'adresse <http://maps.google.fr/maps> pour saisir les coordonnées Latitude et Longitude, séparées par une virgule, dans la zone de recherche et enfin cliquer sur le bouton Recherche Google Maps, pour localiser votre montage.

Nous avons réalisé un petit logiciel maison intitulé VisuTracker-GPS.exe qui convertit automatiquement les coordonnées et affiche le résultat directement dans Google Map ou Google Earth.

Nota : vous devez au préalable télécharger gratuitement Google Earth à l'adresse <http://earth.google.fr/>. Petite astuce : à l'ouverture décocher l'option Atmosphère dans le menu Affichage sinon on ne voit rien !

La figure 5.47 est une copie d'écran du résultat obtenu dans Google Map.

Résumé des points importants.

Dernière minute

Parmi les fichiers téléchargeables via le site www.dunod.com, vous trouverez une version 2 du programme (trackerGPS V2.bas). Dans cette version l'envoi du SMS est déclenché lorsque le montage reçoit un appel, à condition que le numéro de l'appelant soit celui indiqué dans le programme. Ainsi il n'y a pas de frais de communication à prévoir car il n'y a pas de prise de ligne, c'est la « sonnerie » qui est le vecteur de la commande. Attention, il faut tout de même que la carte SIM utilisée par le tracker inclue la présentation du numéro.

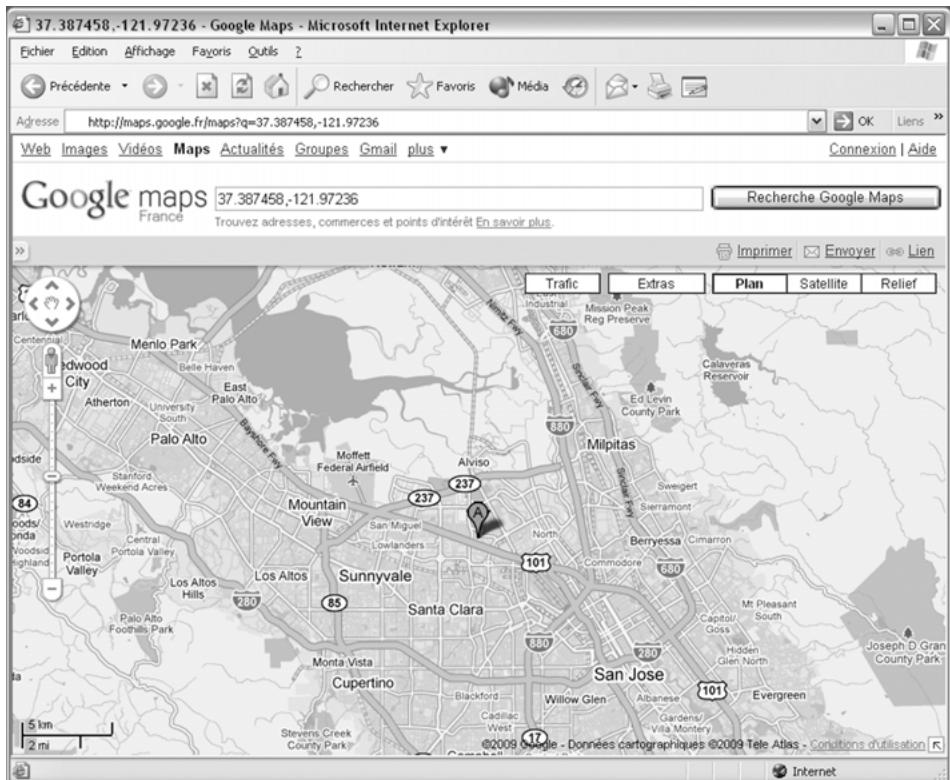


Figure 5.47

TRACKER GPS**Configuration**

Cavalier J1 à mettre en place uniquement si vous utilisez un câble DATA LINK du commerce

Eléments du programme PicBasic à modifier

- ☞ Code PIN (7208 par défaut)
- ☞ Numéro de téléphone utilisé pour l'envoi des SMS (06xxxxxxxx par défaut)

Utilisation

Commande SMS reçue	Action du montage
!!G?	Acquisition des données Latitude et Longitude puis envoi de celles-ci via SMS au numéro programmé.
!!G?,06xxxxxxxx	Acquisition des données Latitude et Longitude puis envoi de celles-ci via SMS au numéro indiqué.

ANNEXES

A.1	Brochages circuits intégrés	252
A.2	Tables des caractères	259

	Glossaire	261
	Bibliographie	264

A.1 BROCHAGES CIRCUITS INTÉGRÉS

PicBasic – 3B

Broche	Désignation	Bloc	Fonction
1	RES	—	Reset
2	I/O 0-AD0	—	E/S ou CAN
3	I/O 1-AD1	—	E/S ou CAN
4	I/O 2-AD2	—	E/S ou CAN
5	I/O 3-AD3	—	E/S ou CAN
6	CLKIN	—	Comptage
7	I/O 4-AD4	—	E/S ou CAN
8	GND	—	Masse
9	OSCIN	—	Quartz
10	OSCOUP	—	Quartz
11	I/O 8	¹ LSB	E/S
12	I/O 9-PWM0	1	E/S ou PWM
13	I/O 10-PWM1	1	E/S ou PWM
14	I/O 11	1	E/S
15	I/O 12	1	E/S
16	I/O 13	1	E/S
17	I/O 14	1	E/S
18	I/O 15	¹ MSB	E/S
19	GND	—	Masse
20	+ 5 V	—	Alimentation
21	I/O 16	—	E/S
22	I/O 17	—	E/S
23	I/O 18	—	E/S
24	I/O 19	—	E/S
25	I/O 20	—	E/S
26	PICBUS	—	Afficheur
27	PCIN	—	Prog. PC
28	PCOUT	—	Prog. PC

Caractéristiques :

- Plage d'alimentation : 4,5 à 5,5 V
- Consommation typique : 15 mA
- Courant maximum par sortie : 25 mA
- Température de stockage : - 40 à + 80 °C
- Température d'utilisation : 0 à 75 °C

PicBasic – 2S

Broche	Désignation	Bloc	Fonction	
1	+ 5 V	—	Alimentation	
2	RES	—	Reset	
3	GND	—	Masse	
4	I/O 0-AD0	0 _{LSB}	E/S ou CAN	
5	I/O 1-AD1	0	E/S ou CAN	
6	I/O 2-AD2	0	E/S ou CAN	
7	I/O 3-AD3	0	E/S ou CAN	
8	I/O 4-AD4	0	E/S ou CAN	
9	I/O 5	0	E/S	
10	I/O 6	0	E/S	
11	I/O 7	0 _{MSB}	E/S	
12	I/O 8	1 _{LSB}	E/S	
13	I/O 9-PWM0	1	E/S ou PWM	
14	I/O 10-PWM1	1	E/S ou PWM	
15	I/O 11	1	E/S	
16	I/O 12	1	E/S	
17	I/O 13	1	E/S	
18	I/O 14	1	E/S	
19	I/O 15	1 _{MSB}	E/S	
20	I/O 16	2 _{LSB}	E/S	
21	I/O 17	2	E/S	
22	I/O 18	2	E/S	
23	I/O 19	2	E/S	
24	I/O 20	2	E/S	
25	I/O 21	2	E/S	
26	I/O 22	2	E/S	
27	I/O 23	2 _{MSB}	E/S	
28	I/O 24-AD5	3 _{LSB}	E/S ou CAN	
29	I/O 25-AD6	3	E/S ou CAN	
30	I/O 26-AD7	3 _{MSB}	E/S ou CAN	
31	CLKIN	—	Comptage	
32	PICBUS	—	Afficheur	
33	NC	—	Non connectée	
34	NC	—	Non connectée	

PICBASIC-2S

+5 V	1	34	NC
RES	2	33	NC
GND	3	32	PICBUS
I/O0-AD0	4	31	CLKIN
I/O1-AD1	5	30	I/O26-AD7
I/O2-AD2	6	29	I/O25-AD6
I/O3-AD3	7	28	I/O24-AD5
I/O4-AD4	8	27	I/O23
I/O5	9	26	I/O22
I/O6	10	25	I/O21
I/O7	11	24	I/O20
I/O8	12	23	I/O19
I/O9-PWM0	13	22	I/O18
I/O10-PWM1	14	21	I/O17
I/O11	15	20	I/O16
I/O12	16	19	I/O15
I/O13	17	18	I/O14

Caractéristiques :

- Plage d'alimentation : 4,5 à 5,5 V
- Consommation typique : 15 mA

- Courant maximum par sortie : 25 mA
- Température de stockage : - 40 à + 80 °C
- Température d'utilisation : 0 à 75 °C

Adaptateur TTL/RS232 MAX232

Broche	Désignation	Fonction	
1	C1+		
2	V+		
3	C1-	Condensateurs 1 µF pour l'activation de la pompe de charge pour passage du niveau de tension TTL à un niveau RS232	
4	C2+		
5	C2-		
6	V-		
7	TX2o	Sortie RS232 numéro 2	
8	RX2i	Entrée RS232 numéro 2	
9	RX2o	Sortie TTL numéro 2	
10	TX2i	Entrée TTL numéro 2	
11	TX1i	Entrée TTL numéro 1	
12	RX1o	Sortie TTL numéro 1	
13	RX1i	Entrée RS232 numéro 1	
14	TX1o	Sortie RS232 numéro 1	
15	GND	Masse	
16	Vcc	Alimentation + 5 V	

MAX232

C1+	1	16	Vcc
V+	2	15	GND
C1-	3	14	TX1o
C2+	4	13	RX1i
C2-	5	12	RX1o
V-	6	11	TX1i
TX2o	7	10	TX2i
RX2i	8	9	RX2o

Amplificateur ULN2803A

Broche	Désignation	Fonction	ULN2803A
1	IN1	Entrées	IN1 □ 1 18 □ OUT1
2	IN2		IN2 □ 2 17 □ OUT2
3	IN3		IN3 □ 3 16 □ OUT3
4	IN4		IN4 □ 4 15 □ OUT4
5	IN5		IN5 □ 5 14 □ OUT5
6	IN6		IN6 □ 6 13 □ OUT6
7	IN7		IN7 □ 7 12 □ OUT7
8	IN8		IN8 □ 8 11 □ OUT8
9	GND	Masse	GND □ 9 10 □ Vcc
10	Vcc	Alimentation + 5 V	
11	OUT8	Sorties (Courant maximum par sortie : 500 mA)	
12	OUT7		
13	OUT6		
14	OUT5		
15	OUT4		
16	OUT3		
17	OUT2		
18	OUT1		

Optocoupleur MOC3041

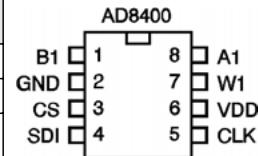
Broche	Désignation	Fonction	MOC3041
1	ANODE	Anode diode émission IR	ANODE □ 1 6 □ CHARGE
2	CATHODE	Cathode diode émission IR	CATHODE □ 2 5 □ NC
3	NC	Non connecté	NC □ 3 4 □ CHARGE
4	CHARGE	Charge max. 400 V/1 A *	
5	NC	Non connecté	
6	CHARGE	Charge max. 400 V/1 A *	

(*)Détection de passage au 0

INTERFACES GSM

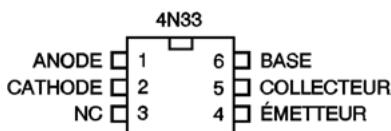
Potentiomètre numérique AD8400

Broche	Désignation	Fonction	
1	B1	« Butée » B du potentiomètre numérique	
2	GND	Masse	
3	CS	Entrée de sélection du circuit (Chip Select) active à l'état bas	
4	SDI	Entrée de donnée série (Serial Data Input)	
5	CLK	Entrée d'horloge, active sur front montant	
6	Vdd	Alimentation comprise entre + 3 V et + 5 V	
7	W1	Curseur du potentiomètre numérique	
8	A1	« Butée » A du potentiomètre numérique	



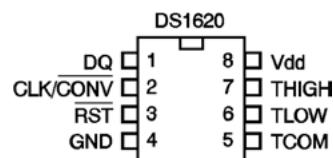
Optocoupleur 4N33

Broche	Désignation	Fonction	
1	ANODE	Anode diode IR	
2	CATHODE	Cathode diode IR	
3	NC	Non connecté	
4	EMETTEUR	Émetteur transistor	
5	COLLECTEUR	Collecteur transistor	
6	BASE	Base transistor (non connecté)	

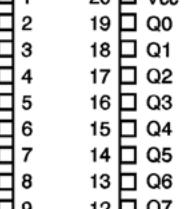


Capteur de température DS1620

Broche	Désignation	Fonction	
1	DQ	Entrée/Sortie de données	
2	CLK/CONV	Horloge/Conversion	
3	RST	Reset (entrées active à l'état bas)	
4	GND	Masse	
5	T _{HIGH}	Indicateur dépassement seuil T° haute	
6	T _{LOW}	Indicateur dépassement seuil T° basse	
7	T _{COM}	Passe à l'état haut quand la T° dépasse le seuil TH, passe à l'état bas quand la T° passe en dessous du seuil TL	
8	Vdd	Alimentation + 5 V	



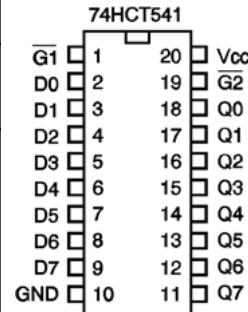
74HCT574

Broche	Désignation	Fonction	74HCT574
1	CLK	Horloge Sur un front montant appliqu� sur CLK, la valeur pr�sente sur les entr�es Dx et recopi�e sur les sorties Qx	
2	D0		
3	D1		
4	D2		
5	D3		
6	D4	Entr�es logiques	
7	D5		
8	D6		
9	D7		
10	OE	S�lection du circuit, actif � l�tat bas	
11	GND	Masse	
12	Q7		
13	Q6		
14	Q5		
15	Q4		
16	Q3		
17	Q2	Sorties logiques	
18	Q1		
19	Q0		
20	Vcc	Alimentation + 5 V	

INTERFACES GSM

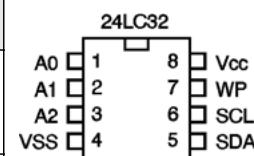
74HCT541

Broche	Désignation	Fonction	
1	G1	Si l'entrée G1 (et G2) est à l'état bas la valeur présente sur les entrées Dx est recopiée sur les sorties Qx	
2	D0		
3	D1		
4	D2		
5	D3		
6	D4		
7	D5		
8	D6		
9	D7		
10	GND	Masse	
11	Q7		
12	Q6		
13	Q5		
14	Q4		
15	Q3		
16	Q2		
17	Q1		
18	Q0		
19	G2	Si l'entrée G2 (et G1) est à l'état bas la valeur présente sur les entrées Dx est recopiée sur les sorties Qx	
20	Vcc	Alimentation + 5 V	



24LC32

Broche	Désignation	Fonction	
1	A0		
2	A1	Entrées dont l'état logique définit l'adresse du circuit	
3	A2		
4	Vss	Masse	
5	SDA	Ligne de données	
6	SCL	Ligne d'horloge	
7	WP	Verrouille la mémoire en écriture lorsque WP = 1	
8	Vcc	Alimentation + 5 V	



A.2 TABLES DES CARACTÈRES

ASCII

		b7	0	0	0	0	1	1	1	1
		b6	0	0	1	1	0	0	1	1
		b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5
0	0	0	0	0	(nul)	(dle)	(sp)	0	@	P
0	0	0	1	1	(soh)	(dc1)	!	1	A	Q
0	0	1	0	2	(stx)	(dc2)	"	2	B	R
0	0	1	1	3	(etx)	(dc3)	#	3	C	S
0	1	0	0	4	(eot)	(dc4)	\$	4	D	T
0	1	0	1	5	(enq)	(nak)	%	5	E	U
0	1	1	0	6	(ack)	(syn)	&	6	F	V
0	1	1	1	7	(bel)	(etb)	'	7	G	W
1	0	0	0	8	(bs)	(can)	(8	H	X
1	0	0	1	9	(tab)	(em))	9	I	Y
1	0	1	0	10	(lf)	(eof)	*	:	J	Z
1	0	1	1	11	(vt)	(esc)	+	;	K	[
1	1	0	0	12	(np)	(fs)	,	<	L	\
1	1	0	1	13	(cr)	(gs)	-	=	M]
1	1	1	0	14	(so)	(rs)	.	>	N	^
1	1	1	1	15	(si)	(us)	/	?	O	_

Exemples de conversion :

$$\text{CR}_{\text{ASCII}} = 0001101_{\text{bin}} = 13_{\text{dec}} = \text{D}_{\text{hex}}$$

$$\text{l}_{\text{ASCII}} = 0110001_{\text{bin}} = 49_{\text{dec}} = 31_{\text{hex}}$$

GSM

		b7	0	0	0	0	1	1	1	1
		b6	0	0	1	1	0	0	1	1
		b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1		0	1	2	3	4	5
0	0	0	0	0	@	Δ	SP	0	-	P
0	0	0	1	1	£	-	!	1	A	Q
0	0	1	0	2	\$	Φ	"	2	B	R
0	0	1	1	3	¥	Γ	#	3	C	S
0	1	0	0	4	è	Λ	¤	4	D	T
0	1	0	1	5	é	Ω	%	5	E	U
0	1	1	0	6	ù	Π	&	6	F	V
0	1	1	1	7	ì	Ψ	'	7	G	W
1	0	0	0	8	ò	Σ	(8	H	X
1	0	0	1	9	ç	Θ)	9	I	Y
1	0	1	0	10	(lf)	Ξ	*	:	J	Z
1	0	1	1	11	ø	Ξ	+	;	K	Ä
1	1	0	0	12	ø	Æ	,	<	L	Ö
1	1	0	1	13	(cr)	æ	-	=	M	Ñ
1	1	1	0	14	Å	ß	.	>	N	Ü
1	1	1	1	15	å	É	/	?	O	§
									o	à

GLOSSAIRE

ASCII : *American Standard Code for Information Interchange.* Table de code à 7 éléments permettant de représenter les lettres et les chiffres sous forme de caractères et de codes (ISO 7).

BCD : *Binary Coded Decimal.* Système de numération où chaque groupe de 4 bits d'un nombre représente un chiffre d'un chiffre.

Bi-bande : Terminal capable de fonctionner indifféremment sur le réseau GSM 1800 et GSM 900. Il s'agit de la même technologie (GSM) utilisée dans des bandes de fréquences différentes : 900 MHz ou 1 800 MHz. Alors que les deux types de réseaux imposaient l'utilisation de terminaux différents, on voit désormais sur le marché des terminaux bi-bandes GSM 900/1800, capables de fonctionner, soit sur un réseau GSM 900, soit sur un réseau GSM 1800, soit sur un réseau bi-bandes GSM 900/1800.

BPS : Vitesse de transmission des données sur un réseau. La norme GSM limite la vitesse de transmission de données à 9 600 bps sur le réseau.

BSIC : Code regroupant le BCC et le NCC, il sert à différencier 2 BTS utilisant le même canal FCN.

BTS : *Base Transceiver Station.* Équipement comprenant l'antenne et les émetteurs/récepteurs radio.

CAN : *Convertisseur Analogique Numérique.* Élément électronique permettant de convertir un signal analogique (ex : tension) en une valeur numérique (ex : octet).

Cell-ID : Numéro codé sur deux octets qui identifie une cellule (ou BTS).

DATA FAX : C'est la fonction des téléphones qui permet le transfert à distance de données ou de fax, d'ordinateur portable à ordinateur. La vitesse de transfert des informations s'exprime en bauds.

dBm : Unité de mesure exprimant un niveau référencé par rapport à une puissance de 1 mW.

E-GSM : *Extended GSM*. Extension du système GSM à d'autres fréquences que la zone de fréquences standards.

ETS : *European Telecommunication Standard*. Nom de la norme créée par l'ETSI, la norme provisoire est nommée I-ETS. Par exemple les normes GSM 07.07 et GSM 07.05 sont des ETS.

ETSI : *European Telecommunications Standard Institute*. Organisme créé par la Commission européenne et chargé de la normalisation des télécommunications.

FCN : *Frequency Chanel Number*. Numéro désignant une porteuse de façon unique dans le système GSM.

GPRS : *General Packet Radio Services*. Système de commutation de données par paquets selon le protocole TCP/IP permettant d'améliorer les débits fournis par les réseaux GSM, on peut espérer un débit de 115 kbits/s. Technologie standardisée à l'ETSI (Institut Européen des Normes de Télécommunication). On trouve désormais sur le marché des téléphones utilisant le GPRS, c'est le cas du MY-X5 de Sagem. Le constructeur SIEMENS propose le terminal MC35 qui dispose de cette fonction.

GSM : *Global System for Mobile communications*. Norme de téléphonie cellulaire numérique européenne développée par l'ETSI. Le GSM utilise une fréquence de 900 MHz et atteint un taux de transfert de 9 600 bits/s. Il existe aussi des versions dérivées du GSM atteignant des fréquences de 1 800 ou 1 900 MHz. Cette norme de téléphone mobile est apparue en 1992 avec un premier appel effectué en Finlande. Le GSM (Itinéris, SFR) est la principale norme utilisée en Europe avec le DCS (Bouygues Télécom).

GSM Phase 1. : Première phase de spécification du système GSM.

GSM Phase 2. : Deuxième phase de spécification du système GSM.

GSM Phase 2+ : Nouvelle phase de spécification du GSM, après un codage plus efficace des données le débit atteint 14,4 kbits/s.

IMEI : *International Mobile Equipment Identity*. Terme qui désigne le numéro d'identification d'un mobile, et qui figure dans le corps de l'appareil ainsi que dans sa mémoire.

IMSI : *International Mobile Subscriber Identity*. Identité Internationale de l'abonné Mobile.

ITU-T : International Telecommunication Union, Telecommunication sector.

LAC : Location Area Code. Code attribué à l'ensemble des cellules d'une même zone.

ME : Mobile Equipment. Équipement mobile qui permet l'envoi et la réception de données sur le réseau GSM.

MODEM : Modulateur-Demodulateur. Il s'agit d'un dispositif, ou d'un périphérique de conversion des données qui transittent via une ligne téléphonique RTC ou GSM. (conversion en émission analogique vers numérique).

OPÉRATEUR : Compagnie offrant des services de télécommunications.

PDU : Protocol Data Unit. Protocole qui définit la constitution numérique de la trame d'un SMS.

PIN : Personal Identification Number. Numéro d'identification personnel. À l'inverse des cartes bancaires, le code PIN du mobile peut être modifié par l'utilisateur. Un code PIN2 permet également de limiter des droits d'accès lorsque l'on prête son mobile à une autre personne.

PLMN : Public Land Mobile Network. Réseau GSM géré par un opérateur.

PUK : Lorsque 3 codes PIN erronés ont été rentrés, la carte SIM est bloquée, et il faut le code PUK à 8 chiffres pour la débloquer.

ROAMING : Mécanisme permettant d'offrir les mêmes services de télécommunications mobiles à des clients (*roamers*) abonnés à d'autres réseaux ou dans d'autres pays.

SIM : Subscriber Identification Module. Module d'identité d'abonné. La carte SIM est une carte à puce contenant les informations sur les droits d'accès. La carte SIM permet d'activer le mobile, de recevoir son numéro de téléphone ainsi que le droit d'accès au réseau. Les appels d'urgence peuvent être émis avec n'importe quel mobile, même sans carte SIM. Cette technologie est standardisée à l'ETSI.

SMS : Short Message Service. Service de messages courts permettant de transmettre et de recevoir de brefs messages de 160 caractères maximum.

SMS-DELIVER : Représente le protocole qui permet le transfert d'un SMS à partir du SMSC à destination d'un téléphone portable.

SMS-SUBMIT : Représente le protocole qui permet le transfert d'un SMS à partir d'un téléphone portable à destination d'un SMSC.

SMSC : *Short Message Service Center.* Centre de service de messages courts. Tous les messages courts sont tout d'abord transmis dans le SMSC. Le message est ensuite transmis au destinataire depuis ce centre. Le SMSC stocke temporairement les messages lorsque le destinataire n'est pas disponible. Dès que le destinataire est à nouveau disponible sur le réseau (par exemple en allumant son appareil), les messages en attente lui sont transmis.

TA : *Terminal Adaptator.* Assure la liaison entre le ME et le TE.

TE : *Terminal Equipment.* Représente un ordinateur ou un microcontrôleur disposant d'un port série permettant de piloter le ME à travers le TA.

TEXT : Permet de constituer/lire un SMS en mode texte.

TTL : *Transistor Transistor Logic.* Famille de circuits logiques utilisant des transistors bipolaires. L'état logique haut est fixé entre 2 et 5 V, l'état logique bas entre 0 et 0,8 V. Le courant maximum disponible par sortie est de 20 mA.

BIBLIOGRAPHIE

ETSI - ETS – NORME GSM 07.07

ETSI - ETS – NORME GSM 07.05

SIEMENS - DATA SHEET TC35 TERMINAL

LEXTRONIC. COMFILE Technologie - PICBASIC – Manuel de référence Vol.1.A

D. REY. *Interfaces PC* numéro 08 – *Potentiomètres numériques sur port série*

P. GUEULLE. *Téléphones portables et PC* 3^e édition, Dunod, PARIS, 2006