UNE CARTE "PROACTIVE SIM" OUVERTE

(article publié dans Electronique Pratique N° 291)

Voici venu le moment d'implanter, dans une BasicCard, des applications qui "tourneront" sur un téléphone portable faisant office de lecteur de cartes à puce muni d'un clavier et d'un écran.
Cela en exploitant hardiment le mécanisme "Proactive SIM" prévu par la spécification GSM 11.14 et destiné, à l'origine, aux seuls opérateurs de téléphonie mobile.

QUELQUES LIGNES DE BASIC...

Nous avons déjà vu comment moins de 200 lignes de code source ZCBasic suffisent pour développer une carte SIM "minimum", capable d'être reconnue par quasiment n'importe quel téléphone portable, et "bricolée" à volonté.
Même s'il n'en faut pas davantage pour démarrer une "session GSM" et manipuler avec les menus, visibles ou cachés, du téléphone, il serait bien plus motivant de venir y greffer des applications personnelles.
Aussi étonnant que cela puisse paraître, quelques lignes supplémentaires de code source permettent de parvenir à un tel résultat, pourvu que le téléphone soit compatible "SIM Toolkit", autrement dit âgé de quatre à cinq ans, et guère plus.
Comme son nom l'indique, la technologie "Proactive SIM" permet à une carte SIM convenablement programmée, de donner des ordres au téléphone dans lequel elle est insérée, et non plus seulement d'en recevoir.
Elle peut ainsi, notamment, "récupérer" son clavier et son écran pour dialoguer avec l'utilisateur, et bien plus encore lorsqu'elle est émise par un opérateur de téléphonie mobile, et en cours de validité.
Il faut pourtant bien admettre que ceux-ci n'ont plus le monopole du développement de cartes de type SIM : à condition de ne pas chercher à s'inscrire indûment sur leurs réseaux, chacun est libre d'exploiter les spécifications GSM pour faire tout autre chose que téléphoner.
Le meilleur exemple de cette (r)évolution est la "Lifecarte", dans laquelle la société Célavie a imaginé d'enregistrer tous les détails médicaux et personnels de son porteur, qui sont ainsi lisibles en cas d'urgence, en insérant tout simplement la carte dans un téléphone portable GSM.
Normalement, le développement de cartes de ce genre nécessite l'usage de kits logiciels très spécialisés et fort coûteux, mais nous allons découvrir, il suffit d'oser, que le kit BasicCard (gratuit...) n'est pas en reste !

UN MÉCANISME ASTUCIEUX

Rendons à César ce qui est à César : il faut bien reconnaître que les concepts "SIM Application Toolkit" et "Proactive SIM" sont de véritables traits de génie de la communauté GSM.
Selon les normes ISO 7816, en effet, une carte à puce est fondamentalement "esclave" du terminal dans lequel elle est insérée : celui-ci lui envoie des commandes, qu'elle exécute et auxquelles elle répond, un point c'est tout.
Dans le contexte particulier des applications GSM, il a été imaginé qu'une carte SIM puisse retourner un compte-rendu **91 Le** au lieu de **90 00** après l'exécution correcte d'une commande.
Grâce à cette adaptation mineure du protocole T=0, la carte peut prévenir le terminal (en l'occurence un téléphone portable) qu'elle souhaite lui donner un ordre, sous la forme d'un bloc de **Le** octets.
C'est alors au téléphone qu'il appartient d'en prendre connaissance (mais seulement si tel est son bon plaisir !) en envoyant une commande "FETCH" (**A0 12 00 00 Le**).
Celui-ci rendra alors généralement compte à la carte de son exécution (c'est

bel et bien "le monde à l'envers" !), en lui envoyant une commande "ENVELOPE" (**A0 C2 00 00 Lc**), contenant un bloc de **Lc** octets.
Mais dans ce bloc d'octets, il est bien entendu que le téléphone peut à son tour donner des ordres à la carte, selon une syntaxe complètement indépendante du jeu de commandes, forcément restrictif, de la spécification GSM 11.11.
Ce sera notamment le cas si l'utilisateur sélectionne une option dans un menu qui lui a été présenté par la carte.
Car c'est cela, la grande force du concept "Proactive SIM" : la carte peut insérer ses propres menus dans celui du téléphone, et prendre le contrôle de l'afficheur de ce dernier.
Cela permet aux applications qu'elle héberge de dialoguer directement avec l'utilisateur, en toute indépendance vis-à-vis des réseaux des opérateurs, même si elles ont la possibilité d'établir des communications ou d'envoyer des SMS, parfois de façon fort discrète, pour ne pas dire furtive...
Bien entendu, tous ces mécanismes ont été conçus de façon à ne dérouter en rien les téléphones qui, en raison de leur âge, ne sont pas compatibles "SIM Toolkit".
De même, une carte ne supportant aucune fonctionnalité STK ou Proactive SIM doit, en principe, fonctionner convenablement dans un téléphone de la toute dernière génération.
Différentes étapes permettent au téléphone et à la carte de s'informer mutuellement de ce qu'ils savent faire, afin d'éviter tout malentendu susceptible de déclencher des comportements erratiques.
En tout premier lieu, le téléphone vient lire l'octet "Phase" de la carte SIM (dans son fichier 7F20:6FAE ou 7F21:6FAE).
Si cet octet est supérieur à 02h (il sera alors en général à 03h), la carte appartient à la "Phase 2+" et supporte donc au moins un sous-ensemble des fonctionnalités STK.
S'il est lui-même compatible "Phase 2+", le téléphone envoie spontanément à la carte une commande "TERMINAL PROFILE" (**A0 10 00 00 Lc**) contenant un bloc de **Lc** octets décrivant ce qu'il sait faire (un bit y étant affecté à chaque fonctionnalité supportée ou non).
Parallèlement, il ira lire le fichier "SIM Service Table" (7F20:6F38 ou 7F21:6F38) de la carte, où il trouvera le détail des "services" que sait gérer cette dernière (à raison de deux bits par fonctionnalité supportée ou non, et activée ou non).
Notons qu'en présence d'une carte de Phase 2 ou 1 (octet Phase respectivement égal à 02h ou absent), le téléphone doit s'abstenir d'envoyer la commande "TERMINAL PROFILE", qui serait de toute façon rejetée avec un message d'erreur **6D 00** (instruction inconnue).

PASSONS A LA PRATIQUE
Il est naturellement logique de prendre pour point de départ le code source MINISIM.BAS qui nous a déjà servi à développer une carte SIM "minimum".
Rappelons qu'il nécessite, pour sa compilation, une version 4.52 ou supérieure du kit logiciel BasicCard (téléchargeable gratuitement sur www.basiccard.com), et qu'il est destiné aux versions "Professional" de la BasicCard (ZC 5.4 ou ZC 5.5).
Pas énormément plus long (moins de 300 lignes au total, y compris plusieurs petites applications de démonstration), le fichier MINISTK.BAS offert sur le site de la revue présente tout de même quelques différences ponctuelles.
La fonction AscW(), notamment, permet de le compiler pour une ZC 5.4 Rev. A, et non plus seulement pour une Rev. C ou supérieure supportant directement la fonction ASC du Basic.
Mais c'est dans la "SIM Service Table" que se situe une astuce assez particulière : un premier octet à 03h au lieu de 00h.
Cela signifie que, bien que le mécanisme de gestion du code PIN ne soit pas

réellement implémenté dans notre carte, celle-ci reconnaîtra néanmoins les commandes d'activation ou de désactivation du code confidentiel.
Tout l'intérêt de la chose est que nous utiliserons la fonction "activation du PIN" du téléphone pour mettre en route, uniquement à notre initiative, notre "moteur" Proactive SIM.
Insérée dans un téléphone compatible STK, la carte sera reconnue d'emblée comme appartenant à la "Phase 2+", et recevra donc une commande "TERMINAL PROFILE" si le mobile est compatible STK. Son contenu sera mémorisé (dans la chaîne TP$) en vue d'une utilisation que nous détaillerons plus loin.
Une "vraie" carte SIM répondrait normalement à cette commande par un compte-rendu **91 XX** au lieu de **90 00**, afin de prier le téléphone de venir chercher un ordre d'insertion d'un nouveau menu.
La nôtre pourrait faire de même si on enlevait le mot REM précédant l'instruction Call CRD(), et cela pourrait effectivement fonctionner avec certains mobiles, notamment les plus anciens supportant le STK.
L'ennui, c'est que les téléphones les plus récents sont tellement surchargés de travail, à ce stade de leur initialisation, qu'ils n'ont souvent pas le temps de répondre par une commande "FETCH".
C'est expressément toléré par la spécification GSM 11.14, qui admet que cette opération soit remise à plus tard, à charge pour la carte de continuer à répondre **91 XX** au lieu de **90 00** jusqu'à ce que le téléphone veuille bien réagir.
Dans la plupart des cas, le mobile se borne à guetter l'apparition d'un compte-rendu **91 XX** en réponse aux commandes "STATUS" (**A0 F2 00 00 Lc**) ou "GET RESPONSE" (**A0 C0 00 00 Lc**) qu'il envoie fréquemment à la carte.
Mais si nous programmions la BasicCard pour qu'elle réponde autre chose que **90 00** à une telle commande, elle s'abstiendrait aussi de transmettre le bloc de données attendu (comme si **Lc** était à 0).
Il s'agit là d'une particularité de son système d'exploitation, que l'on ne peut contourner à coup sûr qu'avec des versions récentes du compilateur ZCBasic et des cartes (ZC 5.5, notamment).
Or, la plupart des téléphones récents considèrent cela comme une anomalie, et mettent fin à la session GSM sur un message d'erreur (carte SIM non valide) !
On pourrait songer à faire émettre le compte-rendu **91 XX** en réponse à une autre commande, n'ayant pas de données à retourner au terminal (par exemple "UPDATE BINARY"). Malheureusement, rien ne dit qu'une telle occasion se présentera à bref délai, tandis que certains téléphones ignorent un compte-rendu **91 XX** reçu à ce stade.
Quasiment tous, par contre, interprètent correctement un compte-rendu **91 XX** reçu en réponse à une commande d'activation ou de désactivation du code PIN, et c'est finalement par ce biais que nous avons pu résoudre le problème.
Notre carte n'exécutera donc aucune fonction "Proactive SIM" tant que l'utilisateur n'aura pas effectué, depuis le menu du téléphone, une tentative d'activation du code PIN.
Il entrera alors quatre à huit chiffres absolument quelconques, mais une modification triviale du code source (ajout d'un simple IF S$ = ...) suffirait pour que seule une valeur bien précise soit reconnue.
Bien que cette demande d'activation soit en réalité ignorée (on se souvient que les fonctions de gestion du code PIN sont simplement simulées), le téléphone signalera sa bonne exécution !
Mais à partir de ce moment, on doit enfin disposer d'une nouvelle entrée (baptisée P. Gueulle) dans le menu principal (ou "Services") du téléphone.
Si on la sélectionne, un sous-menu doit apparaître, proposant les cinq choix suivants :

- *Term profile* (affichage du "Terminal Profile" précédemment mémorisé).
- *Clear LOCI* (réinitialisation des informations de localisation du mobile).

- *LP = FRE* (langue préférentielle : français).
- *LP = ENG* (langue préférentielle : anglais).
- *SIM Reset* (réinitialisation de la carte SIM par le téléphone).

Pour autant qu'ils occupent exactement douze caractères (espaces compris), tous ces libellés peuvent être librement modifiés avant la compilation du code-source, permettant une large personnalisation de ce menu.
Changer leur longueur (dans la limite de ce que peut afficher le téléphone) serait une toute autre affaire, supposant de modifier avec précision plusieurs octets disséminés un peu partout dans le "programme" construit autour de la commande proactive "SET UP MENU".
Cela, du fait de l'utilisation d'un langage dit "TLV" (Tag, Length, Value), dans lequel apparaissent régulièrement des octets indiquant la longueur des données qui suivent, et dont la structure permet l'imbrication, parfois complexe, de plusieurs niveaux (comme on le verra à la figure 1).
C'est dans le dernier des **Lc** octets de données d'une commande "ENVELOPE" (**A0 C2 00 00 Lc D3** etc.) que le mobile indique à la carte quelle option du menu "P. Gueulle" vient d'être sélectionnée par l'utilisateur.
La valeur de la variable STN est alors chargée, s'il y a lieu, avec le numéro (1, 2, ou 3) de l'application "Proactive SIM" à exécuter, et un compte-rendu approprié est élaboré par le sous-programme Sub CRD().
Si STN = 2, par exemple, la carte renverra un compte-rendu **91 1A** en réponse à la commande "ENVELOPE".
Le téléphone saura alors qu'il doit venir chercher vingt-six (1Ah) octets par une nouvelle commande "FETCH" (**A0 12 00 00 1A**).
Lorsqu'elle recevra cette commande, la carte se souviendra que STN = 2 (car STN est une variable "publique"), et placera dans ces 1Ah octets une structure TLV demandant au téléphone d'afficher, en hexadécimal, les quatre premiers octets du "Terminal profile", mémorisés dans TP$: nous y voila !
Dans la foulée, STN sera remis à 0, afin que l'ordre ne soit pas exécuté à plusieurs reprises.

PROGRAMMER EN "TLV"
Analysons donc un peu le contenu de ce "programme" en miniature, qui met en oeuvre la commande proactive "DISPLAY TEXT".
Le langage "TLV" est basé sur l'emploi de "tags", codes tenant sur un octet et fixant la signification des octets de données ("value") qui le suivent.
Entre les deux, un octet "length" précise le nombre d'octets (longueur) du champ "value".
Bien entendu, plusieurs de ces structures "SIMPLE-TLV" peuvent se suivre, comme le montre notre exemple.
Le premier octet est à D0h, valeur de "tag" signalant que tout ce qui suit est une commande "Proactive SIM", écrite en code "BER-TLV" (Basic Encoding Rules).
Le second indique la longueur du bloc d'octets venant après lui, ici vingt-quatre (18h), y compris les douze octets de TP$ (soit quatre groupes de deux caractères hexa, suivis chacun d'un espace).
Les douze octets qui se situent entre deux ont la signification suivante :

- Description de la commande (tag 81h) : 03h (3 octets suivent), 01h (commande N°1), 21h (type = "Display Text"), 81h (priorité haute, effacement manuel par l'utilisateur).
- Identités (tag 82h) : 02h (2 octets suivent), 81h (expéditeur = SIM), 02h (destinataire = afficheur).
- Texte alphanumérique (tag 8Dh) : 0D (13 octets suivent), 04 (codage sur 8 bits).

Bien évidemment, tout cela ne s'invente pas, mais on "s'y met" finalement

assez vite, un peu comme au langage machine d'un microcontrôleur.
La spécification GSM 11.14 (téléchargeable gratuitement sur www.etsi.org) décrit avec précision comment coder chacune des commandes "proactives" que recense le tableau ci-dessous :

Commande	Type	En ligne
Refresh	01h	
More Time	02h	
Poll interval	03h	
Polling Off	04h	
Set Up Call	10h	*
Send SS	11h	*
Send USSD	12h	*
Send SMS	13h	*
Play Tone	20h	
Display Text	21h	
Get Inkey	22h	
Get Input	23h	
Select Item	24h	
Set Up Menu	25h	
Provide local information	26h	*

Si l'on considère que chacune de ces commandes peut admettre de multiples variantes, selon la valeur de l'octet "qualifier" qui la suit (dans notre exemple, pour fixer la priorité et le mode d'effacement du texte affiché), il est clair que les possibilités sont immenses.
Cela, même en se limitant aux commandes utilisables "hors ligne", c'est-à-dire sans inscription sur le réseau d'un opérateur.

ENCORE QUELQUES OCTETS

Imaginons maintenant que l'utilisateur ait sélectionné l'option "LP = ENG" : apprenant que c'est l'entrée N°4 du menu qui a été choisie, la commande "ENVELOPE" met à 01h le premier octet de la chaîne LP$ stockée dans la mémoire EEPROM de la carte (fichier GSM "Language Preference").
Nous sommes cette fois en présence d'un programme strictement interne à la BasicCard (STN reste à 0, et le compte-rendu à **90 00**), qui pourrait bien sûr être infiniment plus complexe (par exemple de type cryptographique).
A partir de maintenant, donc, la langue préférentielle de la carte SIM est l'anglais, mais rien ne transparaît encore au niveau de l'affichage : ce changement ne sera effectif qu'après un arrêt du mobile, ou bien après un "reset" de la carte commandé par la cinquième option du menu.
Cela si, et seulement si, le choix de la langue a été programmé sur "automatique" dans le menu du téléphone (si une langue donnée a été présélectionnée dans le mobile, elle sera en effet prioritaire par rapport à celle spécifiée comme "préférentielle" dans la carte SIM).
Là encore, le "reset" de la SIM est effectué (par le téléphone) à la demande de l'application résidant dans la carte, qui répond à une commande "FETCH" (cas STN = 3), par une structure TLV contenant une commande proactive "REFRESH" (code opératoire 01h), suivie d'un "qualifier" 04h précisant l'ampleur de la réinitialisation demandée.
On aura compris qu'il ne s'agit évidemment là que de quelques exemples, volontairement très simples : l'essentiel de la puissance d'une telle carte tient au fait qu'il reste, dans une BasicCard ZC 5.4 ou ZC 5.5, énormément de place pour les propres applications du développeur.
En fait, celui-ci peut fort bien ne se servir des fonctionnalités "Proactive SIM" que pour communiquer, comme nous venons de le faire, avec le clavier et

l'afficheur du téléphone, qui servira alors de lecteur de cartes à puce évolué.
Et si l'on s'en tient là, ce n'est vraiment pas aussi compliqué qu'on pourrait le croire...