Programmation BasicCard

Didier DONSEZ
Université Joseph Fourier
Polytech'Grenoble –LIG/ADELE
Didier.Donsez@imag.fr

Sommaire

- Motivation
- Cartes BasicCard
- Cycle de Développement
- Langage ZC-Basic

Motivation

- Développement d'applications carte
  en langage Basic

- Coté Carte
  - interpréteur de « bytecode » P-Code
- Coté Terminal
  - Interpréteur de « bytecode » P-Code
  - Remarque : l'application terminal peut être écrite aussi
    - en C/C++/C# avec un driver PS/SC ou MUSCLE
    - en Java avec OCF

Exemple : Commande GetCustomerCredits d'une carte de pré-paiement (PME,Tickets,…)

- Programme Carte

```
Eeprom CustomerCredits ' Declare a permanent Integer variable
Command &H20 &H01 GetCustomerCredits (Credits) ' Credits is a result
  Credits = CustomerCredits
End Command
```

- Programme coté PC

```
Const swCommandOK = &H9000
Declare Command &H20 &H01 GetCustomerCredits (Credits)
Status = GetCustomerCredits (Credits)
If Status <> swCommandOK Then
  Print "GetCustomerCredits: Status = &H"; Hex$ (SW1SW2)
  GoTo CancelTransaction
End If
Print "current credits:"; Credits
```

Produits (www.zeitcontrol.de)

- *Selon les gammes (Compact, Enhanced†, Professional‡)*
- Carte mono applicative, Carte multi applicative (06/2004)
- Mémoire
  - RAM=256 à 1024 octets
  - EEPROM=1Ko à 29Ko
  - ROM=17,7Ko
- Communications ISO 7816-3
  - ATR
  - T=1 (mode bloc) ‡ (et T=0 mode octet)
- Commandes et réponses APDU ISO 7816-4
- Système de Fichiers (non ISO 7816-4, DOS like)
- Support calcul flottant‡
- Fonctions cryptographiques (natif ou en bibliothèque)
  - SHA1, SG-LFSR
  - DES, 3DES, AES (128b), IDEA, RSA (1024b), Elliptic Curve (161b)
- Générateur matériel de nombres aléatoires‡

Produits (www.zeitcontrol.de)

- *ZC1.x (Compact x=1)*
  - *format module SIM du GSM11.11*
  - *T=1 (ISO/IEC 7816-3 et ISO/IEC 7816-4*
  - *Chiffrage des commandes SG-LFSR (Shrinking Generator – Linear Feedback Shift Register) avec des clés de 64 bits.*
  - *256o de RAM, 1 Ko EEPROM*
- *ZC3.x (Enhanced) 1998*
  - *T=1*
  - *DES, 3DES*
  - *File System, Calcul flottant IEEE*
  - *256o de RAM, 2 à 16 Ko EEPROM*
- *ZC4.x (Professional) 2001*
  - *T=0 et T=1*
  - *1024o RAM, 16 à 30 Ko EEPROM*
  - *DES, 3DES, AES 128 bits, RSA 1024, Elliptic Curve 167 bits, SHA-1*
- *ZC5.x (Professional)*
  - *4 fois plus rapide que les ZC4.x*
- *ZC6.x (x=5 06/2004)*
  - *Carte multi-application*
  - *Chaque application doit être signé pour le card provider*
  - *1,75Ko RAM, 30 Ko EEPROM*
  - *DES, 3DES, AES 128-192-256 bits, Elliptic Curve 167 bits, SHA-1*
- *ZC7.5*
  - *Combi or Dual Interface (Contact and ContactLess)*

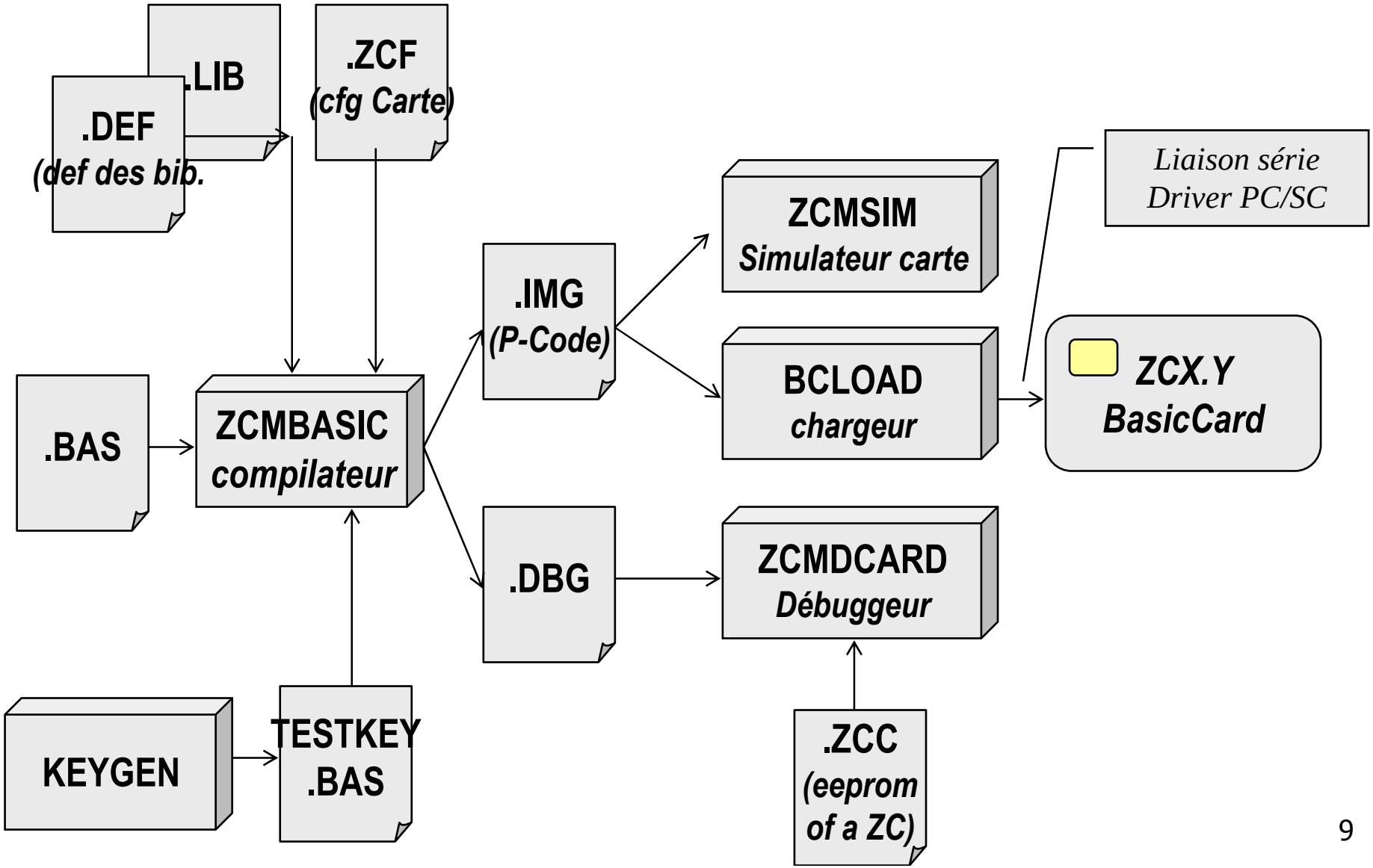
ATR (cf 7.2)

- Par défaut
  - Compact et Enhanced
    - `TS T0 TB1 TC1 TD1 TD2 TA3 TB3 T1-TK`
    - `3B EF 00  FF  81  31  50 or 20 45 or 75 'BasicCard ZCvvv'`
  - Professionnal
    - `TS T0 TA1 TB1 TC1 TD1 TC2 T1-TK`
    - `3B F9 13 00 FF 40 80 'ZC4.1 RSA'`
- Configurable
  - En ZC-Basic (en respectant les contraintes de la carte)
    - Declare ATR= <séquence de Byte et String>

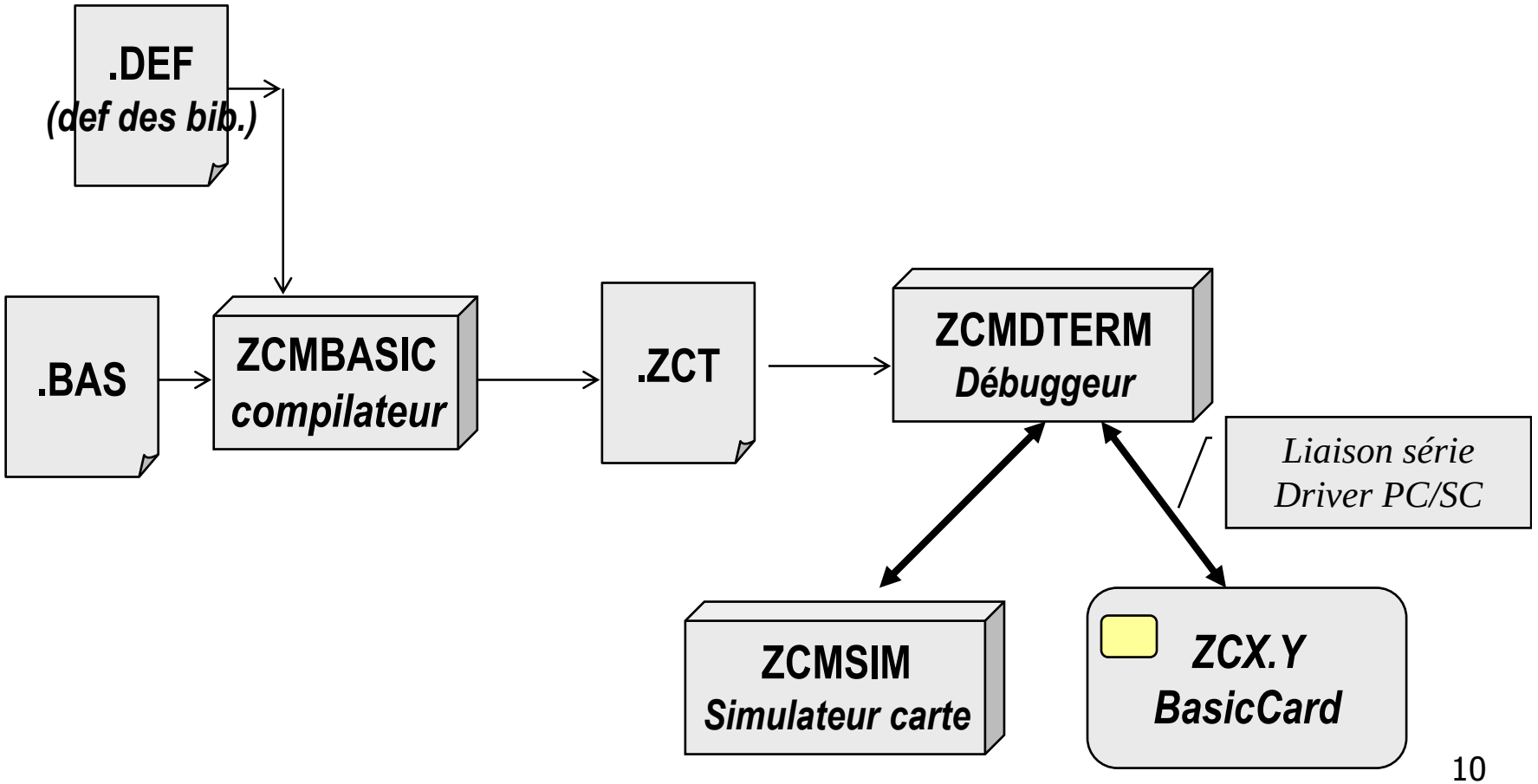
Environnements de développement

- Carte
  - Gestion de projets
  - Débuggeur
  - Compiler ZC-Basic (en P-Code)
  - Chargeur (P-Code)
  - Simulateur
  - Générateurs de clé, …
- Terminal

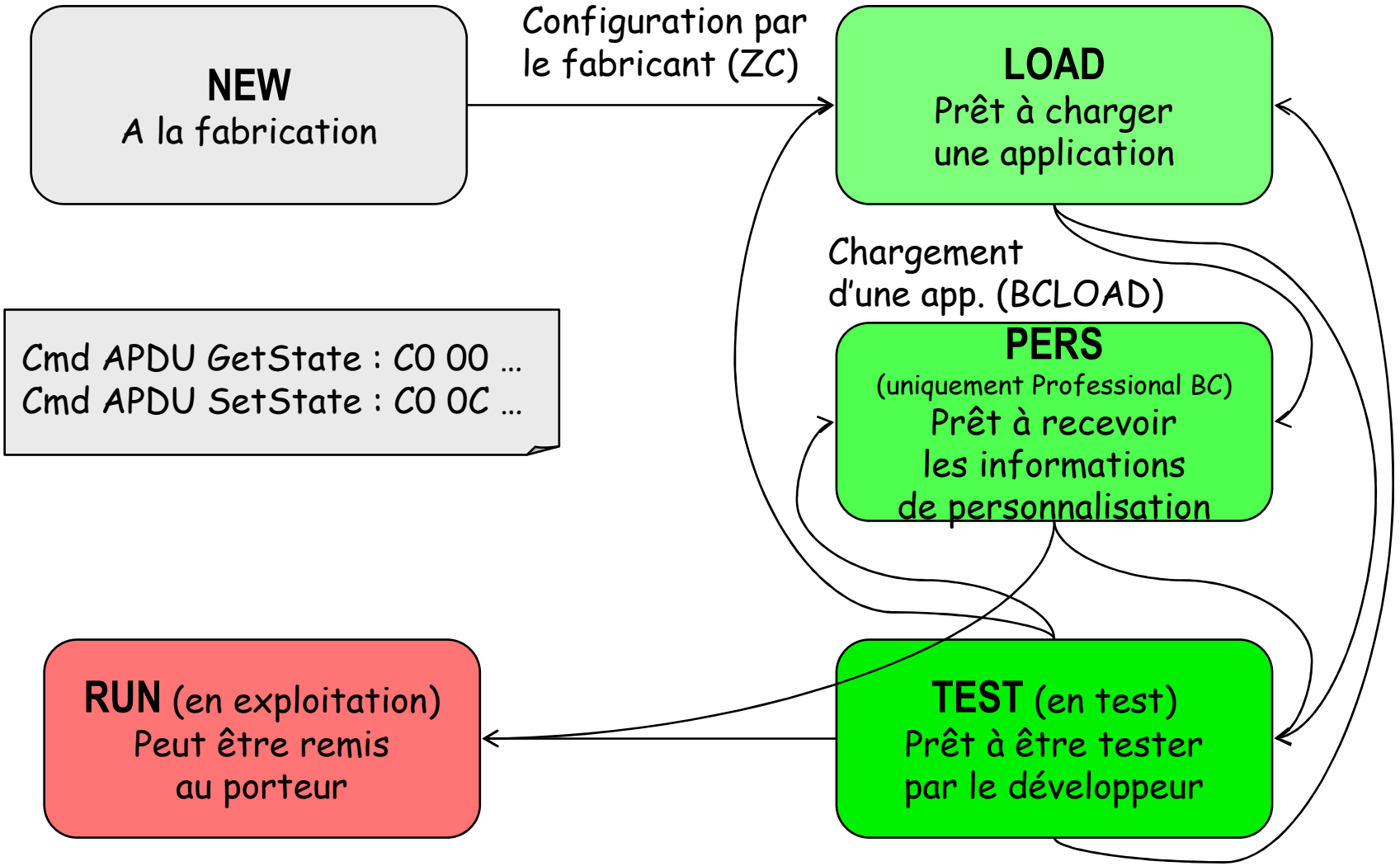
Environnements de développement Coté Carte



Environnements de développement Coté Terminal



Cycle de vie d'une carte (cf 7.7.1)



Le langage ZC-Basic

- Généralités
- Variables
- Typage primitif, tableau, structure
- Structures de contrôle
- Fonctions, Procédures et Commandes

Environnement

- Directive de pré-compilation
  - #Include, #Library, #If, #Else, #Endif, #IfDef, …
- Constantes
  - #Stack, #Files, #Message, #State, #BWT …
- Séparateur d'instruction :
- Commentaires
  - ‘’
  - rem
- Mots réservés

Constantes prédéfinies

Variables

- Déclaration implicite et explicite
  - `option explicit` supprime la déclaration implicite pour éviter les erreurs (de frappe)
- Portée des variables
  - Persistantes (en EEPROM)
    - **Eeprom CustomerName$ = "" ' We don't know customer's name yet**
    - **Eeprom Balance& = 500 ' Free 5-euro bonus for new members**
  - Session (reset au Reset)
    - Public (globale)
    - Static (local à la procédure)
  - Procédure (reset lors de l'appel de la procédure)
    - Private

Portée des variables

Nature	Déclaration	Durée de Vie	Portée	Initialisation
Eeprom	Explicite	Permanente	Globale	A l'installation
Public	Explicite	Temporaire	Globale	Au Reset
Static	Explicite	Temporaire	Locale	Au Reset
Private	Explicite ou implicite	Temporaire	Locale	A l'invocation

Typage (3.5)

Byte	@	entier 8 bits non signé
Integer	%	entier 16 bits signé
Long	&	entier 32 bits signés
Single	!	Flottant IEEE 32 bits
String	$	chaîne variable de (<254) car.
String*n		chaîne fixe de n car. n<255

- @,%,&,!,$ sont les suffixes pour le typage des variables/paramètres
- `String` utilise un pointeur (16 bits) vers la zone dans laquelle sont stockés les caractères préfixés par la longueur (8 bits)
- **DefByte | DefInt | DefLng | DefSng | DefString** } *range* [**,** *range***, . . .**]
  - Définit le type par défaut des variables dont le préfixe est dans l'intervalle
  - Exemple **DefInt A-Z** signifie que par défaut toute variable est entière

Déclaration de Variables

```
DefInt I-R
Private Iter = 0 ' type Int

Public Radius! = 1 ‘ type Single, affectation avec conversion
Public Pistr$ = "3.1416"
Public Pi As Single = Val!(Pistr$)

Eeprom S1 As String*5 = "ABC" ' complété avec des octets NULL
Public S2 As String*3 = &H81, &H82, &H83
Private S3 As String*7 = 3, 4, "XYZ" ’ = 3, 4, 88, 89, 90, 0, 0
```

Déclaration de (Variable) Tableaux

- Déclaration d'un tableau fixe multi-dimension
  - **Dim** *array* **(** [, , . . .]) [**As** *type*]
- Déclaration d'un tableau dynamique multi-dimension
  - **Dim Dynamic** *array* **(** [, , . . .]) [**As** *type*]
- Rédimensionnement
  - **ReDim** *array* **(***bounds* [, *bounds*, . . .])
  - **ReDim** *array* **(***bounds* [, *bounds*, . . .]) [**As** *type*]
- Destruction
  - **Erase** *array* [, *array*, . . .]
- Définition des bornes
  - **Option Base** [*lower-bound* **To**] *upper-bound*

Déclaration de (Variable) Tableaux

```
DefInt I-T
Private Tab(4) = 1,2,3,4 ' tableau de 4 entiers démarrant à 0

Option Base 1        ' fixe la borne inférieure à 1
                     ' pour les nlles déclarations

Private X$(2,2) = "X$(1,1)", "X$(2,1)", "X$(1,2)", "X$(2,2)"
```

Type structuré

- Déclarations de type structuré

```
Type Point: X!, Y!: End Type ' X et Y sont de type Single
Type Rectangle
  Area As Single
  TopLeft As Point
  BottomRight As Point
End Type
```

- Déclarations de variable

```
Public Square As Rectangle = 0,1,1,2,2 ' TopLeft= (1.0,1.0) BottomRight = (2.0,2.0)
```

- Accès aux champs

```
Sub Area (R As Rectangle) ' Calcule le champs Area
    R.Area = (R.BottomRight.X! – R.TopLeft.X!) *  (R.BottomRight.Y! – R.TopLeft.Y!)
End Sub
Call Area (Square)
```

- Déclaration d'un tableau de type structuré

```
Public RA(10) As Rectangle ' déclare un tableau de Rectangle
For I = 1 To 10 : RA(I) = Square : Next
```

Opérateurs et Expressions

- **X + 5 *' Apply '+' (addition) to terms X and 5***
- **A(I) * Rnd *' Apply '*' (multiplication) to terms A(I) and Rnd***
- **S$ + "0" *' Apply '+' (concatenation) to terms S$ and "0"***

- **X$ = Chr$(34) + "STRING" + Chr$(34) + Chr$(10) *' 10 = new line***

Affectation

- Affection
  - [**Let**] *var* **=** *expression*
- Affection de sous chaînes
  - Une sous partie de la chaîne est affectée
  - [**Let**] **Mid$ (***string*, *start* [, *length*]**) =** *expression*
    - *À partir de la position start sur length caractères*
  - [**Let**] **Left$ (***string*, *length***) =** *expression*
  - [**Let**] **Right$ (***string*, *length***) =** *expression*
  - [**Let**] *string* **(***n***) =** *expression* ' ***Mid$ (****string, n, 1****)****=expression*

Structures de contrôle (i)

Saut

Goto *label*

…

label:

Saut avec retour

GoSub *label*

…

label:

subroutine-code

Return [*return-label*]

Saut conditionnel multiple

On *expression entière* { **GoTo** | **GoSub** } *label1* [, *label2*, **. . .** , *labeln*]

Structures de contrôle (ii)

- Test

If *condition1* **Then**
code block 1
[**ElseIf** *condition2* **Then**
code block 2]
[**Else**
code block n]
End If

- Choix

Select Case *test-expression*
Case *case-test* [, *case-test*, . . .]
[*code block*]
[**Exit Case**]
[*code block*]
Case *case-test* [, *case-test*, . . .]
[*code block*]
[**Exit Case**]
[*code block*]
. . .
[**Case Else**
[*code block*]
[**Exit Case**]
[*code block*]]
End Select

Structures de contrôle (iii)

- Boucle

For *loop-var* **=** *start* **To** *end* [**Step** *increment*]
[*code block*]
[**Exit For**]
[*code block*]
Next [*loop-var*]

While *condition*
[*code block*]
[**Exit While**]
[*code block*]
Wend

Do [{**While** | **Until**} *condition*]
[*code block*]
[**Exit Do**]
[*code block*]
Loop [{**While** | **Until**} *condition*]

Structures de contrôle (iv)

- Sortie de boucle de la boucle courante

  Exit For, Exit While, Exit Do
- Sortie d'un choix courante

  Exit Case
- Retour de procédure et de fonction

  Exit Sub , **Exit Function**
- Retour à l'appelant dans le programme terminal

  Exit Command
- Sortie du programme terminal

  Exit

Fonction, Procédure et Commande (i) Déclaration

Declare Sub *proc-name* **(**[*param-def*, *param-def*, **. . .**]**)**

Declare Function *proc-name* **(**[*param-def*, *param-def*, **. . .**]**)** [**As** *type*]

Declare Command [*CLA*] [*INS*] *proc-name* **(**[*PreSpec*,] [*param-def*, *param-def*, **. . .**] [, *PostSpec*]**)**

PreSpec

CLA=*constant* alternatif à la spécification de **CLA**

INS=*constant* alternatif à la spécification de **INS**

Lc=0

PostSpec (valable pour T=0) **Disable Le | Input Le**

(ii) Définition

[**Static**] **Function** *proc-name* **(**[*param-def*, *param-def*, **. . .**]**)** [**As** *type*]
[*procedure code*]
[*proc-name* **=** *expression*]
End Function

[**Static**] **Sub** *procedurename* **(**[*param-def*, *param-def*, **. . .**]**)**
[*procedure code*]
End Sub

[**Static**] **Command** [*CLA*] [*INS*] *proc-name* **(**[*PreSpec*,] [*param-def*, *param-def*, **. . .**] [, *PostSpec*]**)**
[*procdedure code*]
End Command

(iii) Passage des paramètres

param-def [{**ByVal** | **ByRef**}] *param-name* [**As** *type*]

passage par référence/variable (défaut) ou par valeur

```
Declare Sub S (X, ByVal Y, ByRef Z) ' 'ByRef' redundant here
Private A, B, C
Call S (A, B, C) ' A and C can change
Call S (ByVal A, ByRef B, C) ' B and C can change
Call S (A+1, B, (C)) ' Nothing can change – 'A+1' and '(C)'
                     ' are not assignable expressions
```

- Cas des String (variables)

  il ne peut y avoir d'une seule chaîne variable dans une commande et en dernier paramètre

  <=*len* contrainte de longueur des chaînes

  Command &H20 &H00 SetUserName(UserID, Name$<=25)

(iv)
Passage des paramètres

- Cas des Tableaux

```
Declare Sub S (A() As Integer) ' les parenthèses sont obligatoire

Dim X (10) As Integer
Dim Y (20) As Long
Dim Z (5, 5, 5)

Call S (X) ' OK
Call S (X()) ' OK – les parenthèses sont optionnelles
Call S (Y) ' Erreur de compilation – Y est de type tableau de Long
Call S (Z) ' Erreur à l'exécution de Call S(Z) qui est à 3 dim.

…
Sub S (A() As Integer)
A (2, 2) = 0
End Sub
```

Fonctions prédéfinies (cf 3.16)

- Numériques

  Abs(*X*), **Sqrt**(*X*), **Rnd**()

- Tableau

  LBound(*array* [, *dim*]), **UBound**(*array* [, *dim*])

- Chaînes

  string (*n*), **Asc**(*string*), **Chr$**(*char-code*), **Hex$**(*val*), **Left$**(*string*, *len*), **LCase$**(*string*), **Len**(*string*), **LTrim$**(*string*), **Mid$**(*string*, *start*[, *len*]) **Right$**(*string*, *len*) **RTrim$**(*string*), **Space$**(*len*), **Str$**(*val*), **String$**(*len*, *char*), **Trim$**(*string*), **UCase$**(*string*) **Val&**(*string*[, *len*]), **Val!**(*string*[, *len*]), **ValH**(*string*[, *len*])

- Cryptographique

  Key(*keynum*), **DES**(*type*, *key*, *block$*) **Certificate**(*key*, *data*)

- Type

  Len(*variable*), **Len**(*type*)

Exemple de PME (1/2)

```
Declare Command &H80 &H10 SetCustomerName(Name$<=25)
Declare Command &H80 &H20 GetCustomerName (Name$)
Declare Command &H80 &H30 DebitAccount (Amount As Long)
Declare Command &H80 &H40 setBalance (NewBalance As Long)

Const InsufficientCredit = &H6F00
Eeprom Balance As Long
Eeprom CustomerName As String

Command &H80 &H10 SetCustomerName(Name$<=25)
  CustomerName$=Name$
End Command

Command &H80 &H20 GetCustomerName (Name$)
  Name$ = CustomerName$
End Command
```

Exemple (2/2 a)

```
Command &H80 &H30 DebitAccount (Amount As Long)
  If Balance < Amount Then
    SW1SW2 = InsufficientCredit
  Else
    Balance = Balance - Amount
  End If
End Command

Command &H80 &H40 setBalance (NewBalance As Long)
  Balance = NewBalance
End Command
```

Exemple (2/2 b) avec fiabilisation

- Remarque : l'écriture en EEPROM est lente et peut être inachevée en cas d'arrachement

```
Eeprom ShadowBalance As Long
Eeprom Committed = False
Command &H80 &H20 DebitAccount (Amount As Long)
 If Balance < Amount Then SW1SW2 = InsufficientCredit
 Else
  ShadowBalance = Balance – Amount
  Committed = True : Balance = ShadowBalance : Committed = False
 End If
End Command
Command &H80 &H30 setBalance (NewBalance As Long)
 ShadowBalance = NewBalance
 Committed = True : Balance = ShadowBalance : Committed = False
End Command
Rem Then in the initialisation code:
If Committed Then Balance = ShadowBalance : Committed = False : End If
```

Programmation Terminal

- Affichage
  - **Cls, Print, FgCol**, **BgCol**, **CursorX**, **CursorY**
- Saisie
  - **InKey$, Line Input** *X$,* **Input** *variable-list*
- Communication
  - **CardReader** [(*name$*)], **CardInReader, ResetCard** [(*ATR$*)], *nReaders* **= PcscCount,** *ReaderName* **= PcscReader(***ReaderNum***)**
- Journalisation/Trace (ATR+APDU)
  - **Open Log File** *filename,* **Close Log File**

Cryptographie : gestion des clés

- Déclaration des clés et des polynômes (algo SG-LFSR)
  - **Declare Key *key****num*** [(*length* [**,** *counter*])] [= *b1*, *b2*, *b3*, . . .]
  - **Declare Polynomials =** *PolyA&*, *PolyS&*
  - peuvent être :
  - générées avec KEYGEN
  - incluses à la compilation
  - chargées dans la carte avec BCKEYS
  - chargées à l'exécution du terminal avec : **Read Key File** *filename*
- Désignation d'une clé courante pour le chiffrage
  - **Enable Key *key****num*** [(*counter*)]
    - Counter donne le nombre de tentatives de chiffrage incorrectes avant abandon de la clé
- Abandon de la clé courante
  - **Disable Key** *keynum*

Cryptographie

- Chiffrage / Déchiffrage DES
  - *result$* **= DES(***type*, *keynum*, *block$***)**
    - *type*=+1 (chiffrage), –1(dechiffrage) DES
    - *type*=+3 (chiffrage), –3(dechiffrage) Triple DES
- Signature DES/3DES
  - *S$* **= Certificate(***keynum*, *data***)**
  - **Principe**
    - **Découper P en N blocs de 64 bits**
      - **(le dernier peut être complété avec des 0)**
    - **C=0x0000000000000000**
    - **Pour i de 0 à N-1 faire C = E(C Xor P[i])**
    - **Retourner C**
- Autres algorithmes
  - *fournis comme des bibliothèques à inclure*

Génération de Nombres aléatoires (cf 3.18)

- Fonctions
  - Rnd() ‘retourne un Long
  - Randomize *seed ‘Long ou String*
- Méthodes
  - Terminal
    - basée sur l’horloge système
  - Compact et Enhanced
    - Basée sur un numéro de série unique (en EEPROM)
    - Donne une séquence différente entre toutes les cartes et d’un reset à l’autre
  - Professionnal
    - Disposif matériel (Truly RNG)

Chiffrage des échanges (cf 3.17)

- Chiffrage automatique des commandes et réponses APDU
  1. Générer un fichier de clés avec KEYGEN
  2. Inclure ou charger les clés dans les programmes terminal et cartes
  3. Inclure COMMANDS.DEF dans le programme terminal
  4. Basculer le chiffrage automatique
  5. Appeler des commandes entre les commandes
     - *Compact and Enhanced BasicCards:*
     - **Call StartEncryption (P1=*algorithm*, P2=*keynum*, Rnd)**
     - **Call EndEncryption()**
     - *Professional BasicCard:*
     - **Call ProEncryption (P1=*algorithm*, P2=*keynum*, Rnd, Rnd)**
     - **Call EndEncryption()**

Fichiers et Répertoires (cf 4.)

- Système de fichiers Carte
  - Hiérarchique (Répertoires)
  - 1 répertoire racine \
  - Désignation à la DOS (max 254 car. pour le chemin)
  - Contrôle d'accès (R,W) et de concurrence (verrou)
  - Accès séquentiel / aléatoire structuré ou non
  - Ensemble de commandes prédéfinies (cf. 7.7.14)
    - Cla=C0, Ins=18, P1=opération, …

Fichiers et Répertoires (cf 4.)

- Coté carte
  - Un seul lecteur par défaut

- Coté terminal
  - Le système de fichier carte est représenté par un lecteur spécial “@:” (en plus des lecteurs `A:` à `Z:` de l’hôte)
    - Le fichier “`\Transport\Bus\Credits`” dans la carte est désigné par “`@:\Transport\Bus\Credits`” par le programme terminal

Fichiers et Répertoires (cf 4.11)

- Section de définition du système de fichiers de la carte
  - Définit la structure du système de fichier à l'installation de l'application
  - Définition d'un Répertoire
    - **Dir** *path ' le chemin du répertoire*
    - *Lock Definitions '***Lock** ou **Unlock**
    - *File Definitions ' Définition des fichiers*
    - *Sub-directory Definitions ' Définition (récursive) des sous répértoires*
    - **End Dir**
  - Définition d'un Fichier
    - **File** *filename* [**Len =** *blocklen*]
    - *Lock Definitions '***Lock** ou **Unlock**
    - *expr* [**As** *type*] [**(***repeat-count***)**] [**,** *expr* [**As** *type*] [**(***repeat-count***)**], **. . .**]

Exemple de section de définition

```
Declare ApplicationID = "FILECARD"
#Include GOODKEYS.BAS

Dir "\"
  File "DATA1.TXT"                          ' crée le fichier \DATA1.TXT
  Read Lock Key = 99,100                    ' accès en lecture avec une des 2 clés
  "Contenu du fichier DATA1.TXT"            ' contient la chaîne

  File "DATA2.TXT"                          ' crée le fichier \DATA1.TXT
  Read Write Lock Key = 99,100              ' accès en écriture avec une des 2 clés
  "Contenu du fichier DATA2.TXT"            ' contient la chaîne

  Dir "USERS"                               ' crée le répertoire \USERS

    File "OWNER.DAT"                        ' crée le fichier \USERS\OWNER.DAT
    "DONSEZ","Didier" (2),"UJF",123 As Byte (10)     ' le contenu

  End Dir
End Dir
```

Fichiers et Répertoires (cf 4.)

- Répertoires et Fichiers
  - **MkDir** *path,* **RmDir** *path*
  - **ChDir** *path, S$* **= CurDir** [(*drive*)]
  - **ChDrive** *drive*, *S$* **= CurDrive**
  - **Name** *OldPath* **As** *NewPath*
  - *nFiles* **= Dir (** *filespec*), *file$* **= Dir (** *filespec*, *n*)
  - **SetAttr** *filename*, *attributes*, *attributes* **= GetAttr** *filename*
  - **Kill** *filename*

Fichiers et Répertoires (cf 4.)

- Ouverture d'un fichier
  - **Open** *filename* [**For** *mode*] [**Access** *access*] [*lock*] **As** [**#**] *filenum* [**Len=***recordlen*]
  - *filenum* **= Open** *filename* [**For** *mode*] [**Access** *access*] [*lock*] [**Len=***recordlen*]
    - *mode* = mode d'accès
      - Séquentiel **Input, Output, Append**
      - Alétoire octet **Binary**
      - Aléatoire en enregistrement **Random**
- Fermeture
  - **Close** [[**#**] *filenum* [, [**#**] *filenum* , . . .]]

Fichiers et Répertoires (cf 4.)

- Ecriture
  - **Print #***filenum*, [*field* | *separator*] [*field* | *seperator*] **. . .**
  - **Write** [**#**] *filenum*, *expression-list*
  - **Put** [**#**] *filenum*, [*pos*], *data*
- Lecture
  - **Line Input #***filenum*, *X$*
  - *X$* **= Input** (*len*, [**#**] *filenum*)
  - **Input #***filenum*, *variable-list*
  - **Get** [**#**] *filenum*, [*pos*], *variable* [, *len*]
- Positionnement
  - **Seek** [**#**] *filenum*, *pos*
  - **Seek ([#]** *filenum***)**

Fichiers et Répertoires (cf 4.)

- Divers
  - *filenum* **= FreeFile** ‘nombre de descripteur encore libres
  - **Len (***#filenum***)** ‘longueur du fichier
  - **EOF ([#]** *filenum***)** ‘ True si fin de fichier atteinte.
- Contrôle d’accès
  - **Read Lock** *filename* [**Key =** *k1* [**,** *k2*]]
  - **Read Unlock** *filename*
  - **Write Lock** *filename* [**Key =** *k1* [**,** *k2*]]
  - **Write Unlock** *filename*
  - **Read Write Lock** *filename* [**Key =** *k1* [**,** *k2*]]
  - **Read Write Unlock** *filename*
  - **Get Lock** *filename***,** *LockInfo*

Exemple avec les fichiers (1/3) Coté Carte

```
Declare ApplicationID = "FILECARD"
#Include GOODKEYS.BAS

Command &H20 &H04 EnableKey (Key@, Counter@)
  Enable Key Key@ (Counter@)
End Command

Command &H80 &H0E GetCardFile (Result$)
  Private F : F = Open Result$ : Line Input #F, Result$ : Close
End Command

Dir "\"                                  ' crée 2 fichiers dans le répertoire racine
  File "DATA1.TXT"                       ' File name
  Read Lock Key = 99,100                 ' Access conditions
  "Contenu du fichier DATA1.TXT"         ' Contents

  File "DATA2.TXT"                       ' File name
  Read Write Lock Key = 99,100           ' Access conditions
  "Contenu du fichier DATA2.TXT"         ' Contents
End Dir
```

Exemple avec les fichiers (2/3) Coté Terminal

```
Private ID$ : Call GetApplicationID (ID$) : Call CheckSW1SW2()
If ID$ <> "FILECARD" Then Print "Card FILECARD is required for this test" : Exit
ResetCard : Call CheckSW1SW2()
ChDrive "@"
Private F, S$

Open "DATA1.TXT" As F
If FileError = feAccessDenied Then Print "Access Denied to DATA1.TXT" : Exit
FileError = feFileOK

Read Key File "GOODKEYS.BAS" : Call CheckFileError ("Error reading key file")
Call StartEncryption (P2 = 99, Rnd) : Call CheckSW1SW2()
F= Open "DATA1.TXT" : Call CheckFileError ("Error opening DATA1.TXT")
Line Input #F, S$ : Call CheckFileError ("Error reading DATA1.TXT") : Print S$
S$="DATA1.TXT" : Call GetCardFile(S$) : Print S$
Close F : Call CheckFileError ("Error closing DATA1.TXT")
Call EndEncryption() : Call CheckSW1SW2()
```

Exemple avec les fichiers (3/3) Coté Terminal

```
Rem Ouverture et Création d'un fichier
F = Open "DATA3.TXT" For Output : Call CheckFileError ("Error opening for output")
Print #F, "abcdefg" : Print #F, "hijklem" : Call CheckFileError ("Error writing to file")
Close F : Call CheckFileError ("Error closing file")
F = Open "DATA3.TXT" : Call CheckFileError ("Error opening for input")
Line Input #F, S$ : Call CheckFileError ("Error reading a line")
Print "DATA3.TXT contains:",S$  ' abcdefg
Line Input #F, S$ : Call CheckFileError ("Error reading a line")
Print "DATA3.TXT contains:",S$  ' hijklem
Line Input #F, S$
If FileError = feReadError Then Print "ReadError since no more input"
FileError = feFileOK
Close : Call CheckFileError ("Error closing all files")
Kill "DATA3.TXT" : Call CheckFileError ("Error deleting file")
```

Commandes prédéfinies (cf 7.7)

- CLA=C0
- INS= (disponible selon les états)
  - **GET STATE** **00** Get the state and version of the card
  - **EEPROM SIZE** **02** Get the address and length of EEPROM
  - **CLEAR EEPROM** **04** Set specified bytes to FF
  - **WRITE EEPROM** **06** Load data into EEPROM
  - **READ EEPROM** **08** Read data from EEPROM
  - **EEPROM CRC** **0A** Calculate CRC over a specified EEPROM address range
  - **SET STATE** **0C** Set the state of the card
  - **GET APPLICATION ID** **0E** Get the Application ID string
  - **START ENCRYPTION** **10** Start automatic encryption of cmd/resp. data
  - **END ENCRYPTION** **12** End automatic encryption
  - **ECHO** **14** Echo the command data
  - **ASSIGN NAD** **16** Assign a Node Address to the card
  - **FILE IO** **18** Execute a file system operation

Programmation Terminal depuis un autre langage

- Java via OCF
- VB, VC++, C# sur Windows via PCSC

- Format des commandes BasicCard
  - Cf BasicCard Manual 3.13.1, 3.5
  - Voir les journaux d'APDU

Programmation ZC-Basic Terminal vers d'autres Cartes

- Echange avec cartes non BasicCard
  - JavaCard, CB, SV …
- Déclaration d'une commande SendAPDU (3.13.1)

```
Declare Command SendAPDU(Data as String)
```

- Appel de SendAPDU (3.14.3)

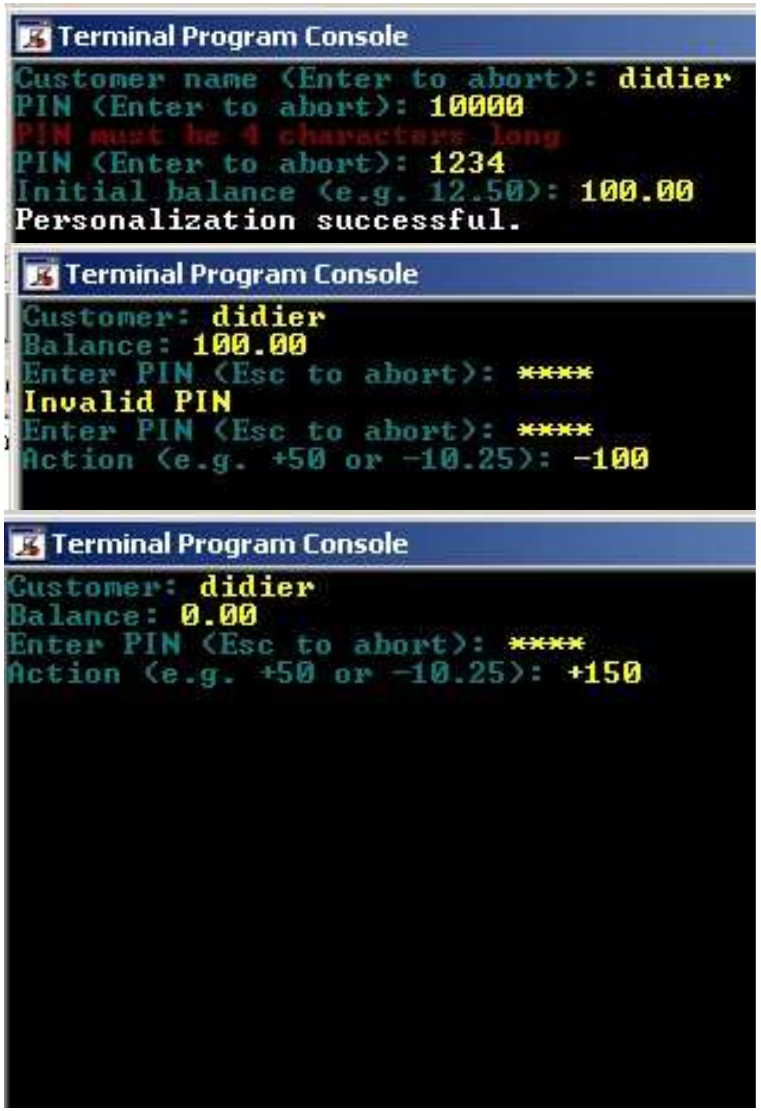
```
Private PCLA As Byte = &HC8
Private PINS  As Byte = &HA0
Private PP1  As Byte = &H11
Private PP2  As Byte = &H22
Private PData$ = "ABC" ' equivalent à &H40, &H41, &H42
Private PLe  As Byte = 45
Call WaitForCard()
ResetCard : Call CheckSW1SW2()
Call SendAPDU(CLA=PCLA,INS=PINS,P1=PP1,P2=PP1,PData$,Le=PLe)
Print SW1SW2
Print ValH(PData$) ' returned bytes
```

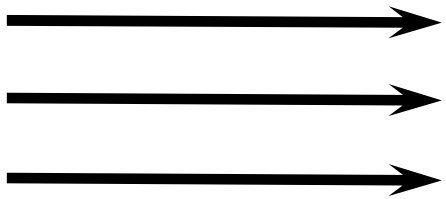
Un exemple : ePurse *(Porte Monnaie Electronique)*

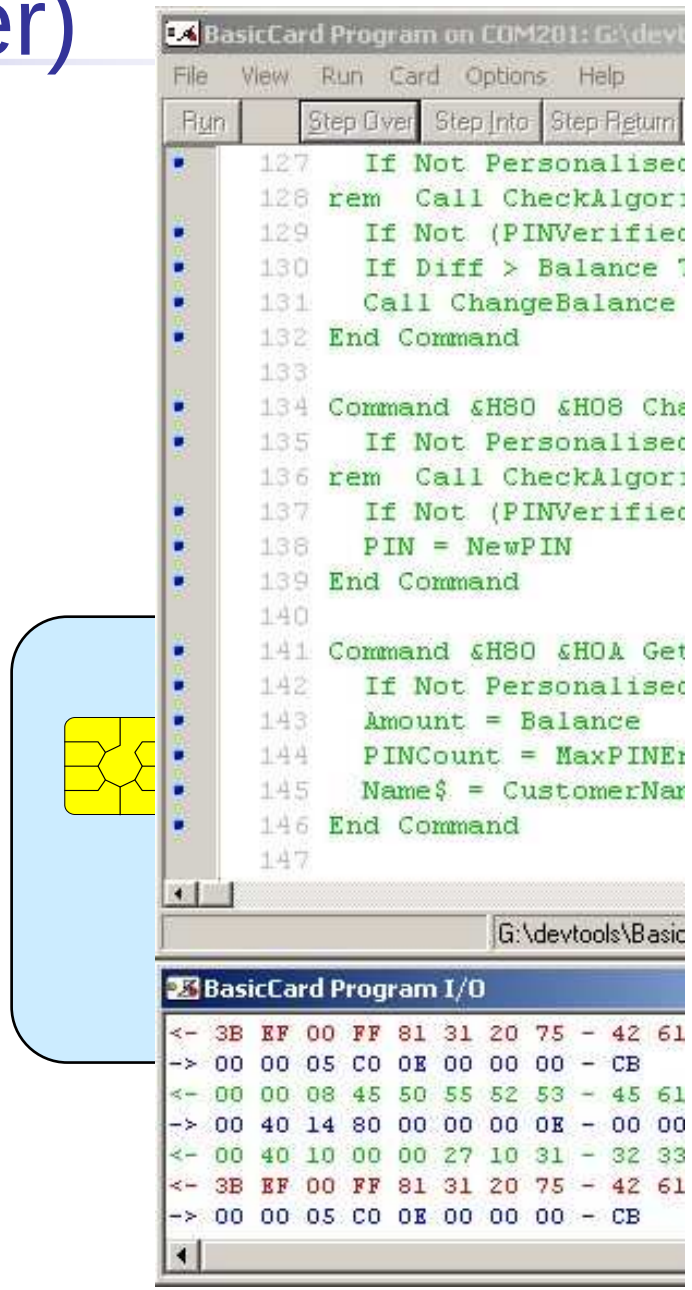
- Une application carte

- Deux applications terminal
  - Issuer (Emetteur)
  - Bearer (Porteur)

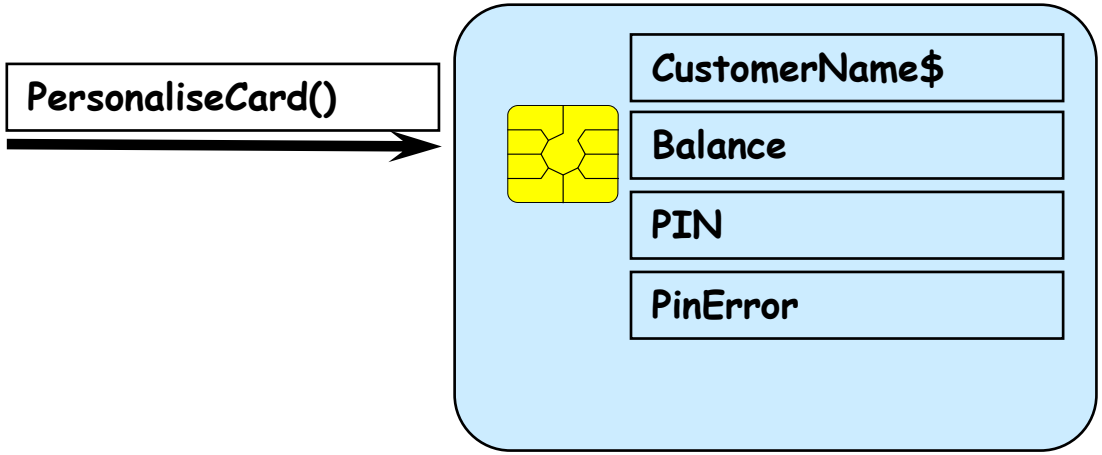
ePurse (Issuer + Bearer)



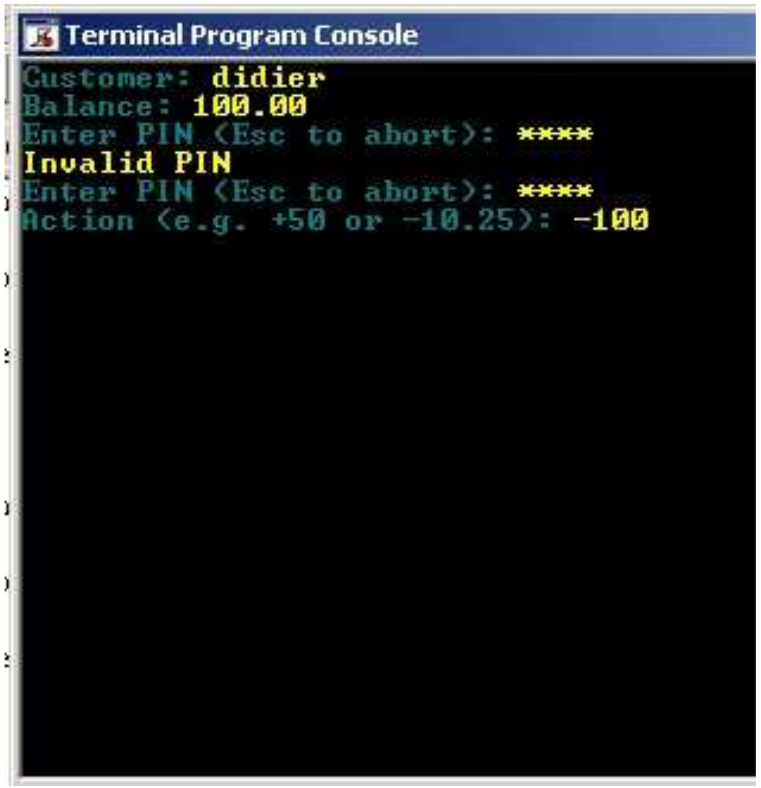


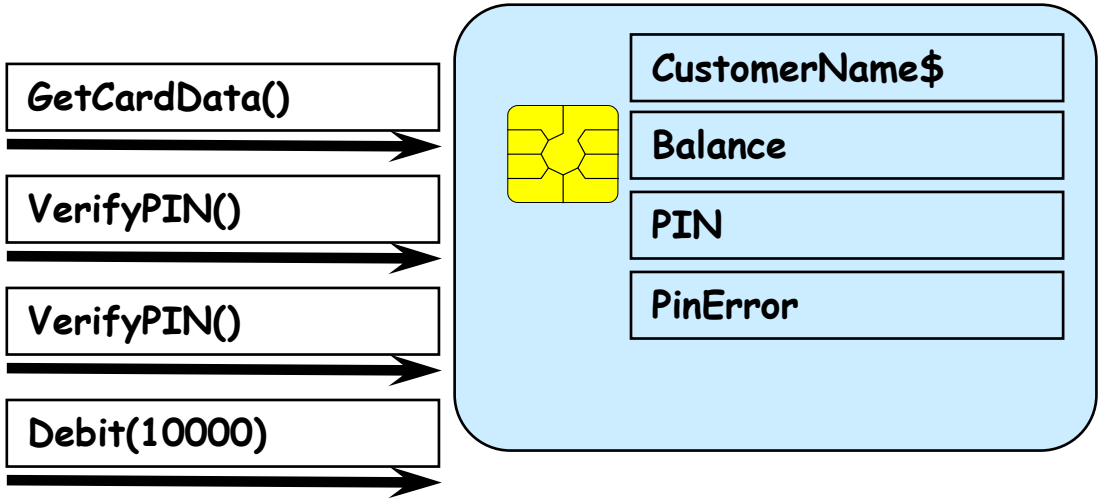


ePurse (Issuer)

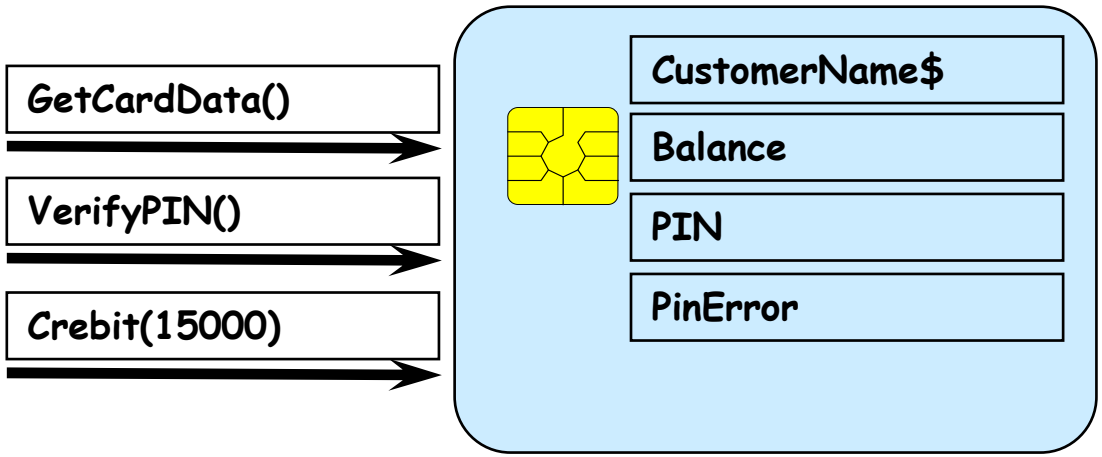


ePurse (Bearer -100)

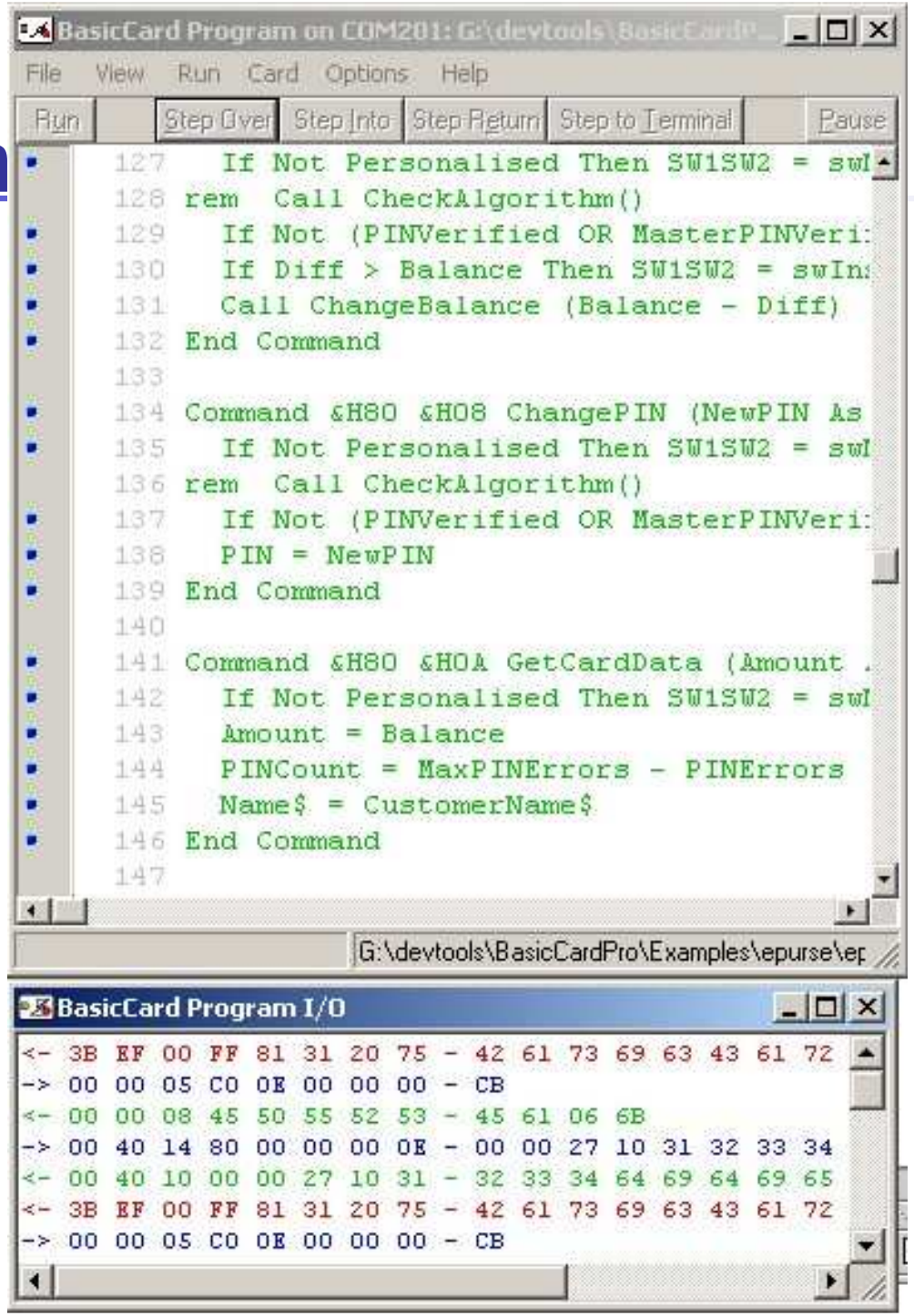




ePurse (Bearer +150)



Sim


BasicCard Program on COM201: G:\devtools\BasicCardP...
File View Run Card Options Help
Run Step Over Step Into Step Return Step to Terminal Pause
127 If Not Personalised Then SW1SW2 = swI
128 rem Call CheckAlgorithm()
129 If Not (PINVerified OR MasterPINVeri:
130 If Diff > Balance Then SW1SW2 = swIn:
131 Call ChangeBalance (Balance - Diff)
132 End Command
133
134 Command &H80 &H08 ChangePIN (NewPIN As
135 If Not Personalised Then SW1SW2 = swI
136 rem Call CheckAlgorithm()
137 If Not (PINVerified OR MasterPINVeri:
138 PIN = NewPIN
139 End Command
140
141 Command &H80 &H0A GetCardData (Amount
142 If Not Personalised Then SW1SW2 = swI
143 Amount = Balance
144 PINCount = MaxPINErrors - PINErrors
145 Name$ = CustomerName$
146 End Command
147
G:\devtools\BasicCardPro\Examples\epurse\ep
BasicCard Program I/O
<- 3B EF 00 FF 81 31 20 75 - 42 61 73 69 63 43 61 72
-> 00 00 05 C0 0E 00 00 00 - CB
<- 00 00 08 45 50 55 52 53 - 45 61 06 6B
-> 00 40 14 80 00 00 00 0E - 00 00 27 10 31 32 33 34
<- 00 40 10 00 00 27 10 31 - 32 33 34 64 69 64 69 65
<- 3B EF 00 FF 81 31 20 75 - 42 61 73 69 63 43 61 72
-> 00 00 05 C0 0E 00 00 00 - CB

Mini-projet BasicCard

- La CasinoCard offre plusieurs jeux de hasard encartés qui permettent au porteur de miser, gagner/perdre sa mise.

- Astuce
  - Completez le code du ePurse
    pour en faire une carte CasinoCard.
  - Ajoutez à cette application carte
    1. un taux de restitution des gains par le Casino qui opère la carte
    2. et une fonction « bet » prenant en paramêtre le montant parié et la probabilité du pari (1/2 pour Pile ou Face, 1/36 pour la roulette, …)
  - Complétez l'application terminal

Webographie

- BasicCard
  - http://basiccard.com/
  - Le Kit est téléchargeable
  - Le développement ne requiert pas de kit matériel au départ
- Fabricant
  - http://www.zeitcontrol.de/

Vos suggestions et vos remarques

- Merci de me les retourner à
  - Didier DONSEZ, Didier.Donsez@imag.fr
- Avez vous trouvé ce cours instructif ?
  - Est il complet ?
  - Qu 'est qu 'il manque ?
  - Qu 'est que vous auriez aimé voir plus développé ?
  - Est il bien organisé ?
  - ...
- Quels sont votre fonction et votre domaine d 'activité ?