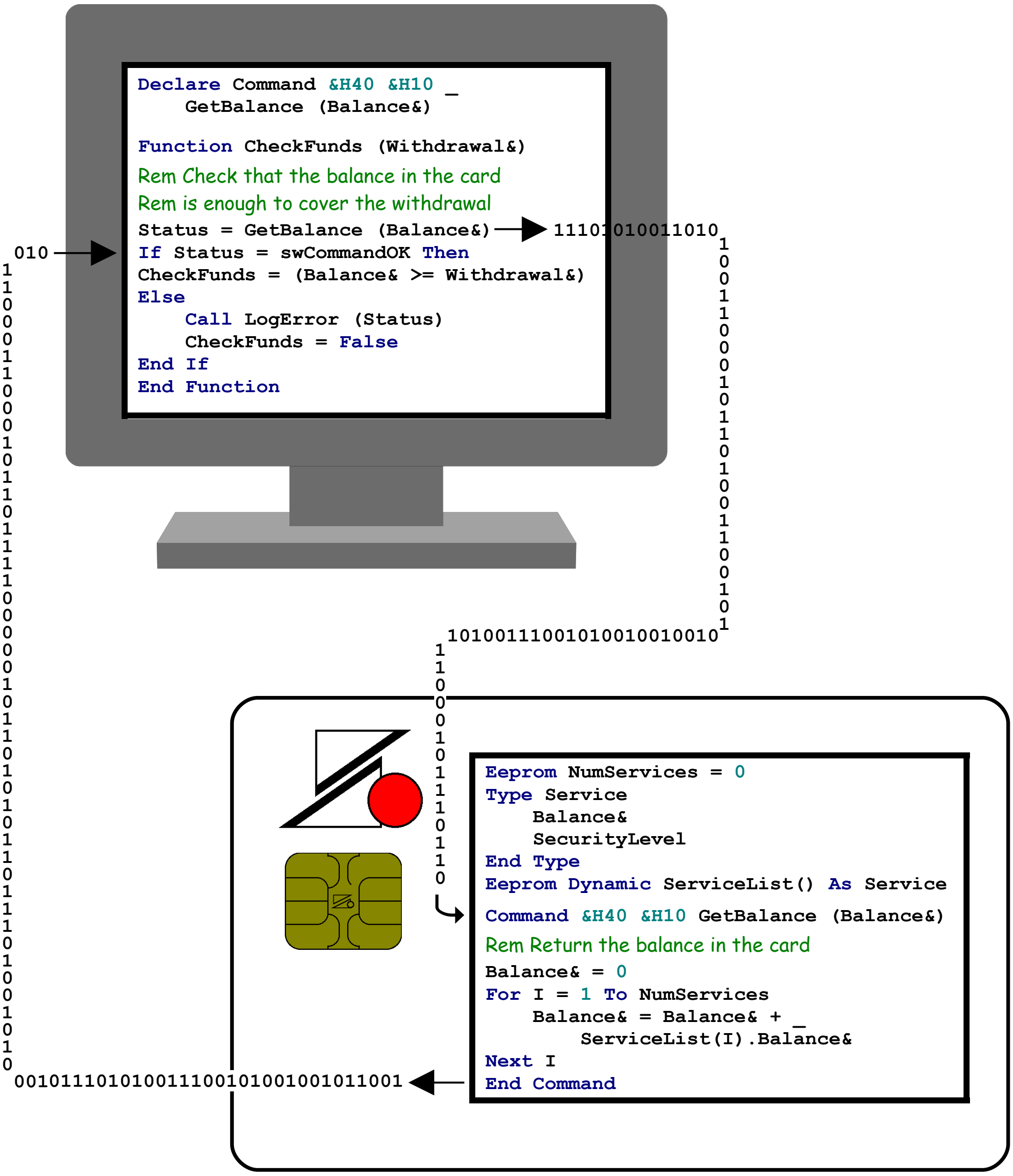
Professional and MultiApplication BasicCard Datasheet



ZeitControl
cardsystems GmbH

The ZeitControl Professional and MultiApplication BasicCards

Document version 8.13
29th June 2012

Author: Tony Guilfoyle
e-mail: development@ZeitControl.de

Siedlerweg 39
D-32429 Minden
Germany

Tel: +49 (0) 571-50522-0
Fax: +49 (0) 571-50522-99

Web sites:

http://www.ZeitControl.de

http://www.BasicCard.com

Introduction

This document lists the features of all the currently available Professional and MultiApplication BasicCard versions. Whenever a new BasicCard is released, this document will be updated accordingly. The following cards are currently available:

ZC5.4, **ZC5.5**, **ZC5.6**	Elliptic Curve Cryptography with **AES** and **DES** encryption
ZC6.5	MultiApplication BasicCard with Elliptic Curve Cryptography, **AES**, and **DES** encryption
ZC7.4, **ZC7.5**, **ZC7.6**	**RSA** and Elliptic Curve Cryptography with **AES** and **DES** encryption, plus **ISO** Secure Messaging
ZC8.4, **ZC8.5**, **ZC8.6**	MultiApplication BasicCard with **RSA**, Elliptic Curve Cryptography, **AES**, and **DES** encryption, plus **ISO** Secure Messaging

This document gives the following information for each of these cards:

- the interfaces supported by the card
- the name of the card's configuration file
- the latest Operating System revision
- the size of available memory (**EEPROM** and **RAM**)
- cryptographic algorithms supported
- the default **ATR** (and the default **ATS** for contactless cards)
- library procedures available in the card
- compile-time options supported for the card

The ZC-Basic language, and the features common to all BasicCards, are described in the separate document **BasicCard: The ZeitControl BasicCard Family**, which can be downloaded from our website at http://www.BasicCard.com.

All these cards support both the **T=0** and **T=1** communication protocols. The default protocol for each card is given along with its default **ATR**; you can override these in the BasicCard source code – see the ZC-Basic documentation for details.

Series 5 Professional BasicCard

The **ZC5-**series Professional BasicCard provides public-key cryptography based on Elliptic Curves over the fields **GF**(2^{167}) and **GF**(2^{211}). There are three versions, **ZC5.4**, **ZC5.5**, and **ZC5.6**, which differ only in the amount of memory available. These cards support symmetric encryption algorithms **AES** and **DES**.

Professional BasicCards ZC5.4, ZC5.5, and ZC5.6

Interface

Contact interface using **T=0** or **T=1** protocol, as defined in **ISO/IEC 7816**

Configuration Files

ZC54_M.ZCF
ZC55_M.ZCF
ZC56_M.ZCF

Operating System Revisions

ZC5.4 REV M
ZC5.5 REV M
ZC5.6 REV M

Available Memory

EEPROM	**ZC5.4**: 16383 (hex **3FFF**) bytes **ZC5.5**: 32767 (hex **7FFF**) bytes **ZC5.6**: 61439 (hex **EFFF**) bytes
RAM	1930 (hex **78A**) bytes

Cryptographic Algorithms		*Key size (bits)*
EC-167	Elliptic Curve public-key cryptography over **GF**(2^{167})	167
EC-211	Elliptic Curve public-key cryptography over **GF**(2^{211})	211
EAX	Encryption with Authentication, using block cipher **AES**	128, 192, and 256
OMAC	One-Key CBC MAC, using block cipher **AES**	128, 192, and 256
AES	Advanced Encryption Standard (Rijndael)	128, 192, and 256
DES	Data Encryption Standard	56, 112, and 168
SHA-1	Secure Hash Algorithm, revision 1	160-bit hash
SHA-256	Secure Hash Algorithm with 256-bit hash	256-bit hash

Default ATRs

ZC5.4: **3B FB 13 00 FF 81 31 80 75 'ZC5.4 REV M' LRC**
ZC5.5: **3B FB 13 00 FF 81 31 80 75 'ZC5.5 REV M' LRC**
ZC5.6: **3B FB 13 00 FF 81 31 80 75 'ZC5.6 REV M' LRC**

T=1 protocol indicated.

Library Procedures

EC167 Library

Sub EC167SetPrivateKey (*Key$***)**
Function EC167SharedSecret (*PublicKey$***) As String**
Function EC167Sign (*Hash$***) As String**
Function EC167Verify (*Signature$***,** *Hash$***,** *PublicKey$***)**

EC211 Library

Sub EC211SetPrivateKey (*Key$***)**
Function EC211SharedSecret (*PublicKey$***) As String**
Function EC211Sign (*Hash$***) As String**
Function EC211Verify (*Signature$***,** *Hash$***,** *PublicKey$***)**

EAX Library

Sub EAXInit (*Type%*, *Key$***)**
Sub EAXProvideNonce (*N$***)**
Sub EAXProvideHeader (*H$***)**
Sub EAXComputeCiphertext (*M$***)**
Sub EAXComputePlaintext (*M$***)**
Function EAXComputeTag() As String

OMAC Library

Function OMACInit (*Type%*, *Key$***) As String**
Function OMAC (*Type%*, *Key$*, *Mess$***) As String**
Function OMACStart (*OmacState* **As String)**
Function OMACAppend (*OmacState* **As String,** *Key$*, *Mess$***)**
Function OMACEnd (*OmacState* **As String,** *Key$***) As String**

AES Library

Function AES (*Type%*, *Key$*, *Block$***) As String**
All key lengths are supported: 128, 192, and 256 bits.

SHA Library (**SHA-1** and **SHA-256**)

Function ShaHash (*S$***) As String**
Sub ShaStart (*HashBuff$***)**
Sub ShaAppend (*HashBuff$*, *S$***)**
Function ShaEnd (*HashBuff$***) As String**

Function Sha256Hash (*S$***) As String**
Sub Sha256Start (*HashBuff$***)**
Sub Sha256Append (*HashBuff$*, *S$***)**
Function Sha256End (*HashBuff$***) As String**

MISC Library

Sub RandomString (*S$*, *Len%***)**
Function LePresent()
Sub SuspendSW1SW2Processing()
Function CardSerialNumber() As String
Function SetProcessorSpeed (*Divider@***) As Byte**

Compile-Time Options

#Pragma Allow9XXX

Series 6 MultiApplication BasicCard

The **ZC6-**series MultiApplication BasicCard enables multiple Applications to be loaded into a single BasicCard without compromising each other's security. It provides public-key cryptography based on Elliptic Curves over the fields **GF**(2^{167}) and **GF**(2^{211}), and symmetric encryption algorithms **AES** and **DES**, with the **EAX** algorithm for Encryption with Authentication, and the **OMAC** algorithm for Authentication.

Interface

Contact interface using **T=0** or **T=1** protocol, as defined in **ISO/IEC 7816**

Configuration File

ZC65_J.MCF

Operating System Revision

ZC6.5 REV J

Available Memory

EEPROM	30975 (hex **78FF**) bytes
RAM	1520 (hex **5F0**) bytes

Cryptographic Algorithms		*Key size (bits)*
EC-167	Elliptic Curve cryptography over the field **GF**(2^{167})	167
EC-211	Elliptic Curve cryptography over the field **GF**(2^{211})	211
EAX	Encryption with Authentication, using block cipher **AES**	128, 192, and 256
OMAC	One-Key CBC MAC, using block cipher **AES**	128, 192, and 256
AES	Advanced Encryption Standard (Rijndael)	128, 192, and 256
DES	Data Encryption Standard	56, 112, and 168
SHA-1	Secure Hash Algorithm, revision 1	160-bit hash
SHA-256	Secure Hash Algorithm with 256-bit hash	256-bit hash

Default ATR

3B FB 13 00 FF 81 31 80 75 'ZC6.5 REV J' LRC
T=1 protocol indicated.

Library Procedures

COMPONENT Library

Sub SelectApplication (*filename$***)**
Sub CreateComponent (*type@*, *name$*, *attr$*, *data$***)**
Sub DeleteComponent (*CID%***)**
Sub WriteComponentAttr (*CID%*, *attr$***)**
Function ReadComponentAttr (*CID%***) As String**
Sub WriteComponentData (*CID%*, *data$***)**
Function ReadComponentData (*CID%***) As String**
Function FindComponent (*type@*, *name$***) As Integer**
Function ComponentName (*CID%***) As String**
Sub GrantPrivilege (*CID%*, *filename$***)**
Function AuthenticateFile (*KeyCID%*, *Signature$*, *Filename$***) As Integer**
Function ReadRightsList (*Filename$*, *RightsList%***()) As Integer**
Sub LoadSequence (*Phase@***)**

EC167 Library

Sub EC167SetCurve (*filename$***)**
Function EC167SharedSecret (*PrivateKey$*, *PublicKey$***) As String**
Function EC167Sign (*PrivateKey$*, *Hash$***) As String**
Function EC167Verify (*Signature$*, *Hash$*, *PublicKey$***)**
Function EC167MakePublicKey (*PrivateKey$***) As String**

EC211 Library

Sub **EC211SetCurve (***filename$***)**
Function EC211SharedSecret (*PrivateKey$***,** *PublicKey$***) As String**
Function EC211Sign (*PrivateKey$***,** *Hash$***) As String**
Function EC211Verify (*Signature$***,** *Hash$***,** *PublicKey$***)**
Function EC211MakePublicKey (*PrivateKey$***) As String**

EAX Library

Sub **EAXInit (***Type%***,** *Key$***)**
Sub **EAXProvideNonce (***N$***)**
Sub **EAXProvideHeader (***H$***)**
Sub **EAXComputeCiphertext (***M$***)**
Sub **EAXComputePlaintext (***M$***)**
Function EAXComputeTag() As String

OMAC Library

Function OMACInit (*Type%***,** *Key$***) As String**
Function OMAC (*Type%***,** *Key$***,** *Mess$***) As String**
Function OMACStart (*OmacState* **As String)**
Function OMACAppend (*OmacState* **As String,** *Key$***,** *Mess$***)**
Function OMACEnd (*OmacState* **As String,** *Key$***) As String**

AES Library

Function AES (*Type%***,** *Key$***,** *Block$***) As String**
All key lengths are supported: 128, 192, and 256 bits.

SHA Library (**SHA-1** and **SHA-256**)

Function ShaHash (*S$***) As String**
Sub **ShaStart (***HashBuff$***)**
Sub **ShaAppend (***HashBuff$***,** *S$***)**
Function ShaEnd (*HashBuff$***) As String**

Function Sha256Hash (*S$***) As String**
Sub **Sha256Start (***HashBuff$***)**
Sub **Sha256Append (***HashBuff$***,** *S$***)**
Function Sha256End (*HashBuff$***) As String**

MISC Library

Sub **RandomString (***S$***,** *Len%***)**
Function LePresent()
Sub **SuspendSW1SW2Processing()**
Function CardSerialNumber() As String
Function SetProcessorSpeed (*Divider@***) As Byte**
Sub **GetFreeMem (***Mem* **As FreeMemoryData)**

Compile-Time Options

#Pragma Allow9XXX
#Pragma CatchUndefinedCommands

Series 7 Professional BasicCard

The **ZC7-**series Professional BasicCards provide:

- contactless and dual-interface versions, as well as the standard **ISO-7816** interface;
- Mifare™ capability;
- **RSA** public-key cryptography for modulus *n* up to 4096 bits long;
- public-key cryptography based on Elliptic Curves over prime fields **GF**(*p*), for *p* up to 544 bits long;
- public-key cryptography based on Elliptic Curves over the fields **GF**(2^{167}) and **GF**(2^{211});
- symmetric encryption algorithms **AES** and **DES**;
- Secure Hash algorithms **SHA-1**, **SHA-224**, **SHA-256** , **SHA-384, and SHA-512**;
- built-in **ISO** Secure Messaging;
- multiple-precision large integer arithmetic;
- EEPROM Transaction Manager for uninterruptable write sequences;
- TLV (Tag-Length-Value) library.

There are three versions, **ZC5.4**, **ZC5.5**, and **ZC5.6**, which differ only in the amount of memory available.

Professional BasicCards ZC7.4, ZC7.5, and ZC7.6

Available Interfaces

Contact interface using **T=0** or **T=1** protocol, as defined in **ISO/IEC 7816**

RFID using **T=CL Type A** contactless protocol, as defined in **ISO/IEC 14443**

Dual interface, contact + **RFID**

Cards supporting **RFID** also provide Mifare™ capability

Configuration Files

ZC74_D.ZCF
ZC75_D.ZCF
ZC76_D.ZCF

Operating System Revisions

ZC7.4 REV D
ZC7.5 REV D
ZC7.6 REV D

Available Memory

EEPROM	**ZC7.4**: 16384 (hex **4000**) bytes **ZC7.5**: 32768 (hex **8000**) bytes **ZC7.6**: 73728 (hex **12000**) bytes
RAM	4214 (hex **1076**) bytes

Cryptographic Algorithms *Key size (bits)*

RSA	Rivest-Shamir-Adleman public-key cryptography	Up to 4096
EC-p	Elliptic Curve public-key cryptography over **GF**(p)	Up to 544
EC-Binary	Elliptic Curve public-key cryptography over **GF**(2^n)	167 and 211
EAX	Encryption with Authentication, using block cipher **AES**	128, 192, and 256
OMAC	One-Key CBC MAC, using block cipher **AES**	128, 192, and 256
AES	Advanced Encryption Standard (Rijndael)	128, 192, and 256
DES	Data Encryption Standard	56, 112, and 168
SHA-1 to **SHA-512**	Secure Hash Algorithm, revision 1	160- to 512-bit hash
SM	**ISO** Secure Messaging	Up to 256 bits

Default ATR's

ZC7.4: **3B DB 18 FF C0 80 B1 FE 75 1F 03 'ZC7.4 REV D' LRC**
ZC7.5: **3B DB 18 FF C0 80 B1 FE 75 1F 03 'ZC7.5 REV D' LRC**
ZC7.6: **3B DB 18 FF C0 80 B1 FE 75 1F 03 'ZC7.6 REV D' LRC**

T=0 and **T=1** protocols indicated.

Default ATS's

37 11 E1 'ZC7.4' CRC
37 11 E1 'ZC7.5' CRC
37 11 E1 'ZC7.6' CRC

Library Procedures

RSA Library

Function RsaExPseudoPrime (*n$*, *nRounds***)**
Sub RsaExGenerateKey (*nBits%*, *pBits%*, *eBits%*, *e$*, *PrK$***)**
Sub RsaExPublicKey (*PrK$*, *PuK$***)**
Sub RsaExEncryptRaw (*Mess$*, *PuK$***)**
Sub RsaExDecryptRaw (*Mess$*, *PrK$***)**
Sub RsaExPKCS1Sign (*Hash$*, *PrK$*, *Sig$***)**
Function RsaExPKCS1Verify (*Hash$*, *PuK$*, *Sig$***)**
Sub RsaExPKCS1Encrypt (*Mess$*, *PuK$***)**
Function RsaExPKCS1Decrypt (*Mess$*, *PrK$***)**
Sub RsaExOAEPEncrypt (*HashLen%*, *Mess$*, *EP$*, *PuK$***)**
Function RsaExOAEPDecrypt (*HashLen%*, *Mess$*, *EP$*, *PrK$***)**
Sub RsaExPSSSign (*Hash$*, *SaltLen%*, *PrK$*, *Sig$***)**
Function RsaExPSSVerify (*Hash$*, *SaltLen%*, *PuK$*, *Sig$***)**
Function RsaExGeneratePrime (*Bytelen%*, *MSW%***) As String**
Function RsaExConstructKey (*p$*, *q$*, *e$*, *PrK$***)**
Function RsaExSetFastPrKOps (*On%***)**
Function RsaExGetFastPrKOps()

EC-p Library

All fourteen **Brainpool Standard Curves**, and all five **NIST Recommended Elliptic Curves**, are available as pre-defined curves.

Sub ECpSetCurve (*CurveIndex%***)**
Sub ECpSetCurveFromFile (*Filename$***)**
Function ECpBitLength()
Sub ECpGenerateKeyPair (*PrK$*, *PuK$***)**
Sub ECpMakePublicKey (*PrK$*, *PuK$***)**
Sub ECpPackPublicKey (*PuK$***)**
Sub ECpUnpackPublicKey (*PuK$***)**
Sub ECpSharedSecret (*PrK$*, *PuK$*, *Secret$***)**
Sub ECpSignNR (*Hash$*, *PrK$*, *Sig$***)**
Function ECpVerifyNR (*Hash$*, *PuK$*, *Sig$***)**
Sub ECpSignDSA (*Hash$*, *PrK$*, *Sig$***)**

Function ECpVerifyDSA (*Hash$*, *PuK$*, *Sig$*)
Sub ECpAddPoints (*P$*, *Q$*)
Sub ECpMultiplyPoint (*P$*, *n$*)

EC-167 Library

Sub EC167SetCurve (ReadOnly *Filename$*)
Function EC167SharedSecret (*PrivateKey$*, *PublicKey$*) **As String**
Function EC167SignNR (*PrivateKey$*, *Hash$*) **As String**
Function EC167VerifyNR (*Signature$*, *Hash$*, *PublicKey$*)
Function EC167MakePublicKey (*PrivateKey$*) **As String**
Function EC167SignDSA (*PrivateKey$*, *Hash$*) **As String**
Function EC167VerifyDSA (*Signature$*, *Hash$*, *PublicKey$*)
Sub EC167SetCurveIndex (*CurveIndex%*)
Function EC167GetCurve() As EC167DomainParams

EC-211 Library

Sub EC211SetCurve (ReadOnly *Filename$*)
Function EC211SharedSecret (*PrivateKey$*, *PublicKey$*) **As String**
Function EC211SignNR (*PrivateKey$*, *Hash$*) **As String**
Function EC211VerifyNR (*Signature$*, *Hash$*, *PublicKey$*)
Function EC211MakePublicKey (*PrivateKey$*) **As String**
Function EC211SignDSA (*PrivateKey$*, *Hash$*) **As String**
Function EC211VerifyDSA (*Signature$*, *Hash$*, *PublicKey$*)
Sub EC211SetCurveIndex (*CurveIndex%*)
Function EC211GetCurve() As EC211DomainParams

Crypto Library

Function CryptoCheckDESKeyParity (ReadOnly *Key$*)
Sub CryptoSetDESKeyParity (*Key$*)
Sub CryptoMAC (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*,_
ReadOnly *Data$*, *MAC* **As String)**
Sub CryptoMACStart (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*)
Sub CryptoMACUpdate (ReadOnly *Data$*)
Sub CryptoMACEnd (*MAC* **As String)**
Sub CryptoEncrypt (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*,_
Data$)
Sub CryptoDecrypt (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*,_
Data$)
Function CryptoSMDecryptCommand (ReadOnly *SMSpec*(),_
ReadOnly *MacKey$*, *MacIV$*, **ReadOnly** *EncKeys$*, *EncIV$*,_
CLA@, **ByVal** *INS@*, **ByVal** *P1P2%*, *IDATA$*, *Le%*)
Sub CryptoSMEncryptResponse (ReadOnly *SMSpec*(),_
ReadOnly *MacKey$*, *MacIV$*, **ReadOnly** *EncKeys$*, *EncIV$*,_
ODATA$, *SW1SW2%*)
Sub CryptoSMEnable (ReadOnly *SMSpec*(),_
ReadOnly *MacKey$*, *MacIV$*, **ReadOnly** *EncKeys$*, *EncIV$*,_
ByVal *Immediate%*)
Sub CryptoSMConfigure (ReadOnly *SMSpec*())
Sub CryptoSMDisable (ByVal *Immediate%*)
Function CryptoSMStatus()
Sub CryptoSetCardKDP (ReadOnly *KDP$*)

SHA Library

Function ShaHash (*S$*) **As String**
Sub ShaStart (*HashBuff$*)
Sub ShaAppend (*HashBuff$*, *S$*)
Function ShaEnd (*HashBuff$*) **As String**
Function Sha256Hash (*S$*) **As String**
Sub Sha256Start (*HashBuff$*)
Sub Sha256Append (*HashBuff$*, *S$*)

Function Sha256End (*HashBuff$***) As String**
Function Sha224Hash (*S$***) As String**
Sub Sha224Start (*HashBuff$***)**
Sub Sha224Append (*HashBuff$***,** ***S$***)**
Function Sha224End (*HashBuff$***) As String**
Function Sha384Hash (*S$***) As String**
Sub Sha384Start (*HashBuff$***)**
Sub Sha384Append (*HashBuff$***,** ***S$***)**
Function Sha384End (*HashBuff$***) As String**
Function Sha512Hash (*S$***) As String**
Sub Sha512Start (*HashBuff$***)**
Sub Sha512Append (*HashBuff$***,** ***S$***)**
Function Sha512End (*HashBuff$***) As String**

EAX Library

Sub EAXInit (*Type%***,** ***Key$***)**
Sub EAXProvideNonce (*N$***)**
Sub EAXProvideHeader (*H$***)**
Sub EAXComputeCiphertext (*M$***)**
Sub EAXComputePlaintext (*M$***)**
Function EAXComputeTag() As String

OMAC Library

Function OMACInit (*Type%***,** ***Key$***) As String**
Function OMAC (*Type%***,** ***Key$***,** ***Mess$***) As String**
Function OMACStart (*OmacState* **As String)**
Function OMACAppend (*OmacState* **As String,** ***Key$***,** ***Mess$***)**
Function OMACEnd (*OmacState* **As String,** ***Key$***) As String**

AES Library

Function AES (*Type%***,** ***Key$***,** ***Block$***) As String**
All key lengths are supported: 128, 192, and 256 bits.

BigInt Library

Function BigIntCompare (ReadOnly *x$***, ReadOnly** *y$***) As Integer**
Function BigIntAdd (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntAddInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntSub (ReadOnly *x$***, ReadOnly** *y$***,** *Negative%***) As String**
Sub BigIntSubInPlace (*x$***, ReadOnly** *y$***,** *Negative%***)**
Function BigIntMul (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntMulInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntDiv (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntDivInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntRem (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntRemInPlace (*x$***, ReadOnly** *y$***)**
Sub BigIntDivRemInPlace (*x$***,** *y$***)**
Function BigIntShiftLeft (ReadOnly *x$***,** *Shift%***) As String**
Sub BigIntShiftLeftInPlace (*x$***,** *Shift%***)**
Function BigIntShiftRight (ReadOnly *x$***,** *Shift%***) As String**
Sub BigIntShiftRightInPlace (*x$***,** *Shift%***)**
Function BigIntAnd (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntAndInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntOr (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntOrInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntXor (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntXorInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntPower (ReadOnly *x$***, ReadOnly e$, ReadOnly** *n$***) As String**
Sub BigIntPowerInPlace (*x$***, ReadOnly e$, ReadOnly** *n$***)**
Function BigIntHCF (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntHCFInPlace (*x$***, ReadOnly** *y$***)**

Function BigIntInvert (ReadOnly *x$*, **ReadOnly** *n$*) **As String**
Sub BigIntInvertInPlace (*x$*, **ReadOnly** *n$*)
Function BigIntSquareRoot (ReadOnly *x$*, **ReadOnly** *p$*) **As String**
Sub BigIntSquareRootInPlace (*x$*, **ReadOnly** *p$*)
Function BigIntJacobiSymbol (ReadOnly *a$*, **ReadOnly** *m$*) **As Integer**

TMLib Library

Sub TMAddTransactionEntry (*Transaction$*, **ReadOnly** *Dest$*, **ReadOnly** *Src$*)
Sub TMCommitTransaction (ReadOnly *Transaction$*)

TLV Library

Sub TLVInitObject (ByRef *Parent* **As TlvPointer, ReadOnly** *Data$*)
Sub TLVInitChild (ReadOnly *Parent* **As TlvPointer,** *Child* **As TlvPointer)**
Function TLVFirstChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ReadOnly** *Data$*)
Function TLVNextChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ReadOnly** *Data$*)
Function TLVFirstMatchingChild (ReadOnly *Parent* **As TlvPointer,**
Child **As TlvPointer, ByVal** *Tag*, **ReadOnly** *Data$*)
Function TLVNextMatchingChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ByVal** *Tag*, **ReadOnly** *Data$*)
Function TLVLastMatchingChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ByVal** *Tag*, **ReadOnly** *Data$*)

Sub TLVEnumInit (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$*)
Function TLVEnumFirst (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$*)
Function TLVEnumNext (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$*)
Function TLVEnumFirstMatching (ByRef *Ptr* **As TlvPointer,**
ReadOnly *Data$,* **ByVal** *Tag*)
Function TLVEnumNextMatching (ByRef *Ptr* **As TlvPointer,**
ReadOnly *Data$*, **ByVal** *Tag*)
Function TLVEnumFirstFX (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$*)

Function TLVCreateObject (ByVal *Tag* **as Integer, ReadOnly** *Value$*) **As String**

Sub TLVAddChild (ReadOnly *Parent* **As TlvPointer, ByVal** *InsertPos*,**_**
ByVal *Tag* **as Integer, ReadOnly** *Value$*, *Data$*)
Sub TLVDeleteChild (ReadOnly *Child* **As TlvPointer,** *Data$*)
Sub TLVReplaceChild (ReadOnly *Child* **As TlvPointer,_**
ByVal *Tag* **as Integer, ReadOnly** *Value$*, *Data$*)

Sub TLVFullObject (*Object* **As TlvPointer, ReadOnly** *Data$*)

Mifare Library

Sub MifareWriteBlock (*BlockNum@*, *Key$*, *Data$*)
Function MifareReadBlock (*BlockNum@*, *Key$*) **As String**
Sub MifareResetBlock (*BlockNum@*)

MISC Library

Sub UpdateCCITTCRC16 (*CRC%*, *S$*)
Sub RandomString (*S$*, *Len%*)
Function LePresent()
Sub SuspendSW1SW2Processing()
Function CardSerialNumber() As String
Function SetProcessorSpeed (*Percent@*) **As Byte**
Function InStr (*Start%*, *S1$*, *S2$*, *Compare@*) **As Integer**
Sub CommParams (*Protocol@*, *Speed@*, *ExtendedLcLe@*)
Sub GetFreeMemory (*Mem* **As ProFreeMemoryData)**

Compile-Time Options

#Pragma Allow9XXX
#Pragma InverseConvention
#Pragma DisableRF
#Pragma RsaFastPrKOps
#Pragma RsaDisableFastPrKOps
#Pragma DSACompatibilityMode
#Pragma EnableMifare

Series 8 MultiApplication BasicCard

The **ZC8-**series MultiApplication BasicCard enables multiple Applications to be loaded into a single BasicCard without compromising each other's security. It provides:

- contactless and dual-interface versions, as well as the standard **ISO-7816** interface;
- Mifare™ capability;
- **RSA** public-key cryptography for modulus *n* up to 4096 bits long;
- public-key cryptography based on Elliptic Curves over prime fields **GF**(p), for p up to 544 bits long;
- public-key cryptography based on Elliptic Curves over the fields **GF**(2^{167}) and **GF**(2^{211});
- symmetric encryption algorithms **AES** and **DES**;
- Secure Hash algorithms **SHA-1**, **SHA-224**, **SHA-256** , **SHA-384, and SHA-512**;
- built-in **ISO** Secure Messaging;
- multiple-precision large integer arithmetic;
- EEPROM Transaction Manager for uninterruptable write sequences;
- TLV (Tag-Length-Value) library.

There are three versions, **ZC8.4**, **ZC8.5**, and **ZC8.6**, which differ only in the amount of memory available.

MultiApplication BasicCards ZC8.4, ZC8.5, and ZC8.6

Available Interfaces

Contact interface using **T=0** or **T=1** protocol, as defined in **ISO/IEC 7816**

RFID using **T=CL Type A** contactless protocol, as defined in **ISO/IEC 14443**

Dual interface, contact + **RFID**

Cards supporting **RFID** also provide Mifare™ capability

Configuration Files

ZC84_D.MCF
ZC85_D.MCF
ZC86_D.MCF

Operating System Revisions

ZC8.4 REV D
ZC8.5 REV D
ZC8.6 REV D

Available Memory

EEPROM	**ZC8.4**: 16384 (hex **4000**) bytes **ZC8.5**: 32768 (hex **8000**) bytes **ZC8.6**: 73728 (hex **12000**) bytes
RAM	4214 (hex **1076**) bytes

Cryptographic Algorithms | | *Key size (bits)*

RSA	Rivest-Shamir-Adleman public-key cryptography	Up to 4096
EC-p	Elliptic Curve public-key cryptography over **GF**(p)	Up to 544
EC-Binary	Elliptic Curve public-key cryptography over **GF**(2^n)	167 and 211
EAX	Encryption with Authentication, using block cipher **AES**	128, 192, and 256
OMAC	One-Key CBC MAC, using block cipher **AES**	128, 192, and 256
AES	Advanced Encryption Standard (Rijndael)	128, 192, and 256
DES	Data Encryption Standard	56, 112, and 168
SHA-1 to **SHA-512**	Secure Hash Algorithm, revision 1	160- to 512-bit hash
SM	**ISO** Secure Messaging	Up to 256 bits

Default ATR's

ZC8.4: **3B DB 18 FF C0 80 B1 FE 75 1F 03 'ZC8.4 REV D' LRC**
ZC8.5: **3B DB 18 FF C0 80 B1 FE 75 1F 03 'ZC8.5 REV D' LRC**
ZC8.6: **3B DB 18 FF C0 80 B1 FE 75 1F 03 'ZC8.6 REV D' LRC**

T=0 and **T=1** protocols indicated.

Default ATS's

37 11 E1 'ZC8.4' CRC
37 11 E1 'ZC8.5' CRC
37 11 E1 'ZC8.6' CRC

Library Procedures

RSA Library

Function RsaExPseudoPrime (*n$*, *nRounds***)**
Sub RsaExGenerateKey (*nBits%*, *pBits%*, *eBits%*, *e$*, *PrK$***)**
Sub RsaExPublicKey (*PrK$*, *PuK$***)**
Sub RsaExEncryptRaw (*Mess$*, *PuK$***)**
Sub RsaExDecryptRaw (*Mess$*, *PrK$***)**
Sub RsaExPKCS1Sign (*Hash$*, *PrK$*, *Sig$***)**
Function RsaExPKCS1Verify (*Hash$*, *PuK$*, *Sig$***)**
Sub RsaExPKCS1Encrypt (*Mess$*, *PuK$***)**
Function RsaExPKCS1Decrypt (*Mess$*, *PrK$***)**
Sub RsaExOAEPEncrypt (*HashLen%*, *Mess$*, *EP$*, *PuK$***)**
Function RsaExOAEPDecrypt (*HashLen%*, *Mess$*, *EP$*, *PrK$***)**
Sub RsaExPSSSign (*Hash$*, *SaltLen%*, *PrK$*, *Sig$***)**
Function RsaExPSSVerify (*Hash$*, *SaltLen%*, *PuK$*, *Sig$***)**
Function RsaExGeneratePrime (*Bytelen%*, *MSW%***) As String**
Function RsaExConstructKey (*p$*, *q$*, *e$*, *PrK$***)**
Function RsaExSetFastPrKOps (*On%***)**
Function RsaExGetFastPrKOps()

EC-p Library

All fourteen **Brainpool Standard Curves**, and all five **NIST Recommended Elliptic Curves**, are available as pre-defined curves.

Sub ECpSetCurve (*CurveIndex%***)**
Sub ECpSetCurveFromFile (*Filename$***)**
Function ECpBitLength()
Sub ECpGenerateKeyPair (*PrK$*, *PuK$***)**
Sub ECpMakePublicKey (*PrK$*, *PuK$***)**
Sub ECpPackPublicKey (*PuK$***)**
Sub ECpUnpackPublicKey (*PuK$***)**
Sub ECpSharedSecret (*PrK$*, *PuK$*, *Secret$***)**
Sub ECpSignNR (*Hash$*, *PrK$*, *Sig$***)**
Function ECpVerifyNR (*Hash$*, *PuK$*, *Sig$***)**
Sub ECpSignDSA (*Hash$*, *PrK$*, *Sig$***)**

Function ECpVerifyDSA (*Hash$*, *PuK$*, *Sig$*)
Sub ECpAddPoints (*P$*, *Q$*)
Sub ECpMultiplyPoint (*P$*, *n$*)

EC-167 Library

Sub EC167SetCurve (ReadOnly *Filename$*)
Function EC167SharedSecret (*PrivateKey$*, *PublicKey$*) **As String**
Function EC167SignNR (*PrivateKey$*, *Hash$*) **As String**
Function EC167VerifyNR (*Signature$*, *Hash$*, *PublicKey$*)
Function EC167MakePublicKey (*PrivateKey$*) **As String**
Function EC167SignDSA (*PrivateKey$*, *Hash$*) **As String**
Function EC167VerifyDSA (*Signature$*, *Hash$*, *PublicKey$*)
Sub EC167SetCurveIndex (*CurveIndex%*)
Function EC167GetCurve() As EC167DomainParams

EC-211 Library

Sub EC211SetCurve (ReadOnly *Filename$*)
Function EC211SharedSecret (*PrivateKey$*, *PublicKey$*) **As String**
Function EC211SignNR (*PrivateKey$*, *Hash$*) **As String**
Function EC211VerifyNR (*Signature$*, *Hash$*, *PublicKey$*)
Function EC211MakePublicKey (*PrivateKey$*) **As String**
Function EC211SignDSA (*PrivateKey$*, *Hash$*) **As String**
Function EC211VerifyDSA (*Signature$*, *Hash$*, *PublicKey$*)
Sub EC211SetCurveIndex (*CurveIndex%*)
Function EC211GetCurve() As EC211DomainParams

Component Library

Sub SelectApplication (*filename$*)
Sub CreateComponent (*type@*, *name$*, *attr$*, *data$*)
Sub DeleteComponent (*CID%*)
Sub WriteComponentAttr (*CID%*, *attr$*)
Function ReadComponentAttr (*CID%*) **As String**
Sub WriteComponentData (*CID%*, *data$*)
Function ReadComponentData (*CID%*) **As String**
Function FindComponent (*type@*, *name$*) **As Integer**
Function ComponentName (*CID%*) **As String**
Sub GrantPrivilege (*CID%*, *filename$*)
Function AuthenticateFile (*KeyCID%*, *Signature$*, *Filename$*) **As Integer**
Function ReadRightsList (*Filename$*, *RightsList%*()) **As Integer**
Sub LoadSequence (*Phase@*)
Sub WriteCardConfig (*DataItem@*, *Data$*)
Function ReadCardConfig (*DataItem@*) **As String**

Crypto Library

Function CryptoCheckDESKeyParity (ReadOnly *Key$*)
Sub CryptoSetDESKeyParity (*Key$*)
Sub CryptoMAC (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*,_
ReadOnly *Data$*, *MAC* **As String)**
Sub CryptoMACStart (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*)
Sub CryptoMACUpdate (ReadOnly *Data$*)
Sub CryptoMACEnd (*MAC* **As String)**
Sub CryptoEncrypt (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*,_
Data$)
Sub CryptoDecrypt (ByVal *Algorithm%*, **ReadOnly** *Key$*, **ReadOnly** *IV$*,_
Data$)
Function CryptoSMDecryptCommand (ReadOnly *SMSpec*(),_
ReadOnly *MacKey$*, *MacIV$*, **ReadOnly** *EncKeys$*, *EncIV$*,_
CLA@, **ByVal** *INS@*, **ByVal** *P1P2%*, *IDATA$*, *Le%*)
Sub CryptoSMEncryptResponse (ReadOnly *SMSpec*(),_

ReadOnly *MacKey$*, *MacIV$*, **ReadOnly** *EncKeys$*, *EncIV$*,_
ODATA$, *SW1SW2%*)

Sub **CryptoSMEnable (ReadOnly** *SMSpec*(),_
ReadOnly *MacKey$*, *MacIV$*, **ReadOnly** *EncKeys$*, *EncIV$*,_
ByVal *Immediate%*)

Sub **CryptoSMConfigure (ReadOnly** *SMSpec*())

Sub **CryptoSMDisable (ByVal** *Immediate%*)

Function CryptoSMStatus()

Sub **CryptoSetCardKDP (ReadOnly** *KDP$*)

SHA Library

Function ShaHash (*S$***) As String**
Sub **ShaStart (***HashBuff$***)**
Sub **ShaAppend (***HashBuff$*, *S$***)**
Function ShaEnd (*HashBuff$***) As String**
Function Sha256Hash (*S$***) As String**
Sub **Sha256Start (***HashBuff$***)**
Sub **Sha256Append (***HashBuff$*, *S$***)**
Function Sha256End (*HashBuff$***) As String**
Function Sha224Hash (*S$***) As String**
Sub **Sha224Start (***HashBuff$***)**
Sub **Sha224Append (***HashBuff$*, *S$***)**
Function Sha224End (*HashBuff$***) As String**
Function Sha384Hash (*S$***) As String**
Sub **Sha384Start (***HashBuff$***)**
Sub **Sha384Append (***HashBuff$*, *S$***)**
Function Sha384End (*HashBuff$***) As String**
Function Sha512Hash (*S$***) As String**
Sub **Sha512Start (***HashBuff$***)**
Sub **Sha512Append (***HashBuff$*, *S$***)**
Function Sha512End (*HashBuff$***) As String**

EAX Library

Sub **EAXInit (***Type%*, *Key$***)**
Sub **EAXProvideNonce (***N$***)**
Sub **EAXProvideHeader (***H$***)**
Sub **EAXComputeCiphertext (***M$***)**
Sub **EAXComputePlaintext (***M$***)**
Function EAXComputeTag() As String

OMAC Library

Function OMACInit (*Type%*, *Key$***) As String**
Function OMAC (*Type%*, *Key$*, *Mess$***) As String**
Function OMACStart (*OmacState* **As String)**
Function OMACAppend (*OmacState* **As String,** *Key$*, *Mess$***)**
Function OMACEnd (*OmacState* **As String,** *Key$***) As String**

AES Library

Function AES (*Type%*, *Key$*, *Block$***) As String**
All key lengths are supported: 128, 192, and 256 bits.

BigInt Library

Function BigIntCompare (ReadOnly *x$*, **ReadOnly** *y$***) As Integer**
Function BigIntAdd (ReadOnly *x$*, **ReadOnly** *y$***) As String**
Sub **BigIntAddInPlace (***x$*, **ReadOnly** *y$***)**
Function BigIntSub (ReadOnly *x$*, **ReadOnly** *y$*, *Negative%***) As String**
Sub **BigIntSubInPlace (***x$*, **ReadOnly** *y$*, *Negative%***)**
Function BigIntMul (ReadOnly *x$*, **ReadOnly** *y$***) As String**
Sub **BigIntMulInPlace (***x$*, **ReadOnly** *y$***)**
Function BigIntDiv (ReadOnly *x$*, **ReadOnly** *y$***) As String**

Sub BigIntDivInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntRem (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntRemInPlace (*x$***, ReadOnly** *y$***)**
Sub BigIntDivRemInPlace (*x$***,** *y$***)**
Function BigIntShiftLeft (ReadOnly *x$***,** *Shift%***) As String**
Sub BigIntShiftLeftInPlace (*x$***,** *Shift%***)**
Function BigIntShiftRight (ReadOnly *x$***,** *Shift%***) As String**
Sub BigIntShiftRightInPlace (*x$***,** *Shift%***)**
Function BigIntAnd (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntAndInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntOr (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntOrInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntXor (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntXorInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntPower (ReadOnly *x$***, ReadOnly e$, ReadOnly** *n$***) As String**
Sub BigIntPowerInPlace (*x$***, ReadOnly e$, ReadOnly** *n$***)**
Function BigIntHCF (ReadOnly *x$***, ReadOnly** *y$***) As String**
Sub BigIntHCFInPlace (*x$***, ReadOnly** *y$***)**
Function BigIntInvert (ReadOnly *x$***, ReadOnly** *n$***) As String**
Sub BigIntInvertInPlace (*x$***, ReadOnly** *n$***)**
Function BigIntSquareRoot (ReadOnly *x$***, ReadOnly** *p$***) As String**
Sub BigIntSquareRootInPlace (*x$***, ReadOnly** *p$***)**
Function BigIntJacobiSymbol (ReadOnly *a$***, ReadOnly** *m$***) As Integer**

TMLib Library

Sub TMAddTransactionEntry (*Transaction$***, ReadOnly** *Dest$***, ReadOnly** *Src$***)**
Sub TMCommitTransaction (ReadOnly *Transaction$***)**

TLV Library

Sub TLVInitObject (ByRef *Parent* **As TlvPointer, ReadOnly** *Data$***)**
Sub TLVInitChild (ReadOnly *Parent* **As TlvPointer,** *Child* **As TlvPointer)**
Function TLVFirstChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ReadOnly** *Data$***)**
Function TLVNextChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ReadOnly** *Data$***)**
Function TLVFirstMatchingChild (ReadOnly *Parent* **As TlvPointer,**
Child **As TlvPointer, ByVal** *Tag***, ReadOnly** *Data$***)**
Function TLVNextMatchingChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ByVal** *Tag***, ReadOnly** *Data$***)**
Function TLVLastMatchingChild (ReadOnly *Parent* **As TlvPointer,_**
Child **As TlvPointer, ByVal** *Tag***, ReadOnly** *Data$***)**

Sub TLVEnumInit (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$***)**
Function TLVEnumFirst (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$***)**
Function TLVEnumNext (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$***)**
Function TLVEnumFirstMatching (ByRef *Ptr* **As TlvPointer,**
ReadOnly *Data$,* **ByVal** *Tag***)**
Function TLVEnumNextMatching (ByRef *Ptr* **As TlvPointer,**
ReadOnly *Data$***, ByVal** *Tag***)**
Function TLVEnumFirstFX (ByRef *Ptr* **As TlvPointer, ReadOnly** *Data$***)**

Function TLVCreateObject (ByVal *Tag* **as Integer, ReadOnly** *Value$***) As String**

Sub TLVAddChild (ReadOnly *Parent* **As TlvPointer, ByVal** *InsertPos***,_**
ByVal *Tag* **as Integer, ReadOnly** *Value$***,** *Data$***)**
Sub TLVDeleteChild (ReadOnly *Child* **As TlvPointer,** *Data$***)**
Sub TLVReplaceChild (ReadOnly *Child* **As TlvPointer,_**
ByVal *Tag* **as Integer, ReadOnly** *Value$***,** *Data$***)**

Sub TLVFullObject (*Object* **As TlvPointer, ReadOnly** *Data$***)**

Mifare Library

Sub MifareWriteBlock (*BlockNum@*, *Key$*, *Data$***)**
Function MifareReadBlock (*BlockNum@*, *Key$***) As String**
Sub MifareResetBlock (*BlockNum@***)**

MISC Library

Sub UpdateCCITTCRC16 (*CRC%*, *S$***)**
Sub RandomString (*S$*, *Len%***)**
Function LePresent()
Sub SuspendSW1SW2Processing()
Function CardSerialNumber() As String
Function SetProcessorSpeed (*Percent@***) As Byte**
Function InStr (*Start%*, *S1$*, *S2$*, *Compare@***) As Integer**
Sub CommParams (*Protocol@*, *Speed@*, *ExtendedLcLe@***)**
Sub GetFreeMemory (*Mem* **As ProFreeMemoryData)**

Compile-Time Options

#Pragma Allow9XXX
#Pragma InverseConvention
#Pragma DisableRF
#Pragma RsaFastPrKOps
#Pragma RsaDisableFastPrKOps
#Pragma DSACompatibilityMode
#Pragma EnableMifare