Advanced Card Systems Limited

Card and Reader Technologies

PORTING GUIDE

ACR120S to ACR120U and Vice Versa

Contents

I. Overview

This document serves as a guide to properly port your ACR120S applications to the new ACR120U device and vice-versa.

Note:

1. Make sure you have the right ACR120 Device.
2. Make sure you have installed the right ACR120 driver. The ACR120 Drivers of both interface can be downloaded from our website (www.acs.com.hk).

Language	ACR120S (ACR120.dll)	ACR120U (ACR120U.dll)
VC6	ACR120.h ACR120.lib	ACR120U.h ACR120U.lib
VB6	ACR120.bas	ACR120U.bas
Delphi7	ACR120.pas	ACR120U.pas

II. Explanation and Differences

Serial API	USB API	Remarks
ACR120_Open(Port, Baudrate)	ACR120_Open(Port)	**ACR120S parameters:** *Port *Baudrate **ACR120U parameter:** * Port
ACR120_Close (rHandle)	ACR120_Close (hReader)	No change.
ACR120_Reset(rHandle, stationID)	ACR120_Reset(hReader)	**ACR120S parameters:** * rHandle *station ID **ACR120U parameter:** * hReader
ACR120_Select(rHandle, stationID, pHaveTag, pTAG, pSN[10]);	ACR120_Select(hReader, pResultTagType, pResultTagLength, pResultSN[10]);	**ACR120S parameters:** * rHandle *stationID *pHaveTag (Indicate whether the TAG Type Identification is returned) *pTAG (Contains the TAG Type Identification if returned) *pSN(card serial number) **ACR120U parameters:** *hReader *pResultTagType(Contain the selected Tag Type) *pResultTagLength(Contain the Length of the selected TAG.) *pResultSN[10](Card Serial Number)

ACR120_Login(rHandle, stationID, sector, keyType, storedNo, pKey[6]);	ACR120_Login(hReader, Sector, KeyType, StoredNo, pKey[6]);	**ACR120S parameters:** *rHandle , *station ID *sector , *keyType -ACR120_LOGIN_KEYTYPE_AA -ACR120_LOGIN_KEYTYPE_BB -ACR120_LOGIN_KEYTYPE_FF -ACR120_LOGIN_KEYTYPE_STORED_A -ACR120_LOGIN_KEYTYPE_STORED_B *storedNo , *pKey[6] **ACR120U parameters:** *hReader , *Sector *KeyType -AC_MIFARE_LOGIN_KEYTYPE_A = 0xAA (KeyType2 = 0xAA) -AC_MIFARE_LOGIN_KEYTYPE_B = 0xBB (KeyType2 = 0xBB) - AC_MIFARE_LOGIN_KEYTYPE_DEFAULT _A = 0xAD (KeyType2 = 0xAA) - AC_MIFARE_LOGIN_KEYTYPE_DEFAULT _B = 0xBD (KeyType2 = 0xBB) - AC_MIFARE_LOGIN_KEYTYPE_DEFAULT _F = 0xFD (KeyType2 = 0xFF) - AC_MIFARE_LOGIN_KEYTYPE_STORED_ A = 0xAF (KeyType2 = 0xAA) - AC_MIFARE_LOGIN_KEYTYPE_STORED_ B = 0xBF (KeyType2 = 0xBB) *StoredNo ,*pKey[6]
ACR120_Read(rHandle, stationID, block, pBlockData[16]);	ACR120_Read(hReader, Block, pBlockData[16]);	**ACR120S parameters:** *rHandle *station ID *block *pBlockData[16] **ACR120U parameters:** *hReader *Block *pBlockData[16]

ACR120_ReadValue(rHandle, stationID, block, pValueData);	ACR120_ReadValue(hReader, Block, pValueData);	**ACR120S parameters:** *rHandle *station ID *block *pValueData **ACR120U parameters:** *hReader *Block *pValueData
ACR120_ReadEEPROM(rHandle, stationID, reg, pEEPROMData);	ACR120_ReadEEPROM(hReader, RegNo, pEEPROMData);	**ACR120S parameters:** *rHandle *stationID *reg (register number) *pEEPROMData **ACR120U parameters:** *hReader *RegNo (register number) *pEEPROMData
ACR120_ReadLowLevelRegister(rHandle, stationID, reg, pRegData);	ACR120_ReadRC531Reg(hReader, RegNo, pValue);	**In ACR120S ACR120_ReadLowLevelRegister was used.** **Parameters:** * rHandle * stationID * reg (register number) * pRegData (Contains the register's value.) **In ACR120U ACR120_ReadRC531Reg was used.** **Parameters:** * hReader * RegNo (register number) * pValue (Contains the register's value.)

ACR120_Write(rHandle, stationID, block, pBlockData[16]);	ACR120_Write(hReader, Block, pBlockData[16]);	**ACR120S parameters:** *rHandle *stationID *block *pBlockData[16] **ACR120U parameters:** *hReader *Block *pBlockData[16]
ACR120_WriteValue(rHandle, stationID, block, valueData);	ACR120_WriteValue(hReader, Block, ValueData);	**ACR120S parameters:** *rHandle *stationID *block *valueData **ACR120U parameters:** *hReader *Block *ValueData
ACR120_WriteEEPROM(rHandle, stationID, reg, eepromData);	ACR120_WriteEEPROM(hReader, RegNo, EEPROMData);	**ACR120S parameters:** *rHandle *stationID *reg (register number) *eepromData **ACR120U parameters:** *hReader *RegNo (register number) *EEPROMData

ACR120_WriteLowLevelRegister(rHandle, stationID, reg, registerData);	ACR120_WriteRC531Reg(hReader, RegNo, Value);	**In ACR120S** **ACR120_WriteLowLevelRegister was used.** **Parameters:** * rHandle * stationID * reg (register number) * registerData (Contains the register value.to write) **In ACR120U ACR120_WriteRC531Reg was used.** **Parameters:** * hReader * RegNo (register number) * Value (Contains the register value to write.)
ACR120_WriteMasterKey(rHandle, stationID, keyNo, pKey[6]);	ACR120_WriteMasterKey(hReader, KeyNo, pKey[6]);	**ACR120S parameters:** *rHandle *stationID *keyNo *pKey[6] **ACR120U parameters:** *hReader *KeyNo *pKey[6]
ACR120_Inc(rHandle, stationID, block, value, pNewValue);	ACR120_Inc(hReader, Block, Value, pNewValue);	**ACR120S parameters:** *rHandle *stationID *block *value *pNewValue **ACR120U parameters:** *hReader *Block *Value *pNewValue

ACR120_Dec(rHandle, stationID, block, value, pNewValue);	ACR120_Dec(hReader, Block, Value, pNewValue);	**ACR120S parameters:** *rHandle *station ID *block *value *pNewValue **ACR120U parameters:** *hReader *Block *Value *pNewValue
ACR120_Copy(rHandle, stationID, srcBlock, desBlock, pNewValue);	ACR120_Copy(hReader, srcBlock, desBlock, pNewValue);	**ACR120S parameters:** *rHandle *stationID *srcBlock *desBlock *pNewValue **ACR120U parameters:** *hReader *srcBlock *desBlock *pNewValue
ACR120_Power(rHandle, stationID, bOn);	ACR120_Power(hReader, State);	**ACR120S parameters:** *rHandle *stationID *bOn (Boolean | Turn on (TRUE) or off (FALSE).) **ACR120U parameters:** *hReader *State (INT8 | Turn OFF (0) or ON (1).)
ACR120_ReadUserPort(rHandle, stationID, pUserPortState);	ACR120_ReadUserPort(hReader, pUserPortState);	**ACR120S parameters:** *rHandle *stationID *pUserPortState Contains the port state (only LSB is used). **ACR120U parameters:** *hReader *pUserPortState Contain the port state (only Bit 2 & Bit 6 are used).

ACR120_WriteUserPort(rHandle, stationID, userPortState);	ACR120_WriteUserPort(hReader, UserPortState);	**ACR120S parameters**: *rHandle *station ID *userPortState (Clear the port pin if userPortState = 0. Otherwise it's set.) **ACR120U parameters:** *hReader *UserPortState Contain the port state to write (only Bit 2 & Bit 6 are used).
ACR120_GetID(rHandle, pNumID, pStationID);	**N/A**	**Function not applicable for ACR120U**
ACR120_ListTag(rHandle, stationID, pNumTagFound, pHaveTag, pTAG, pSN);	ACR120_ListTags(hReader, pNumTagFound, pTagType[4], pTagLength[4], pSN[4][10]);	**ACR120S parameters:** *rHandle *stationID *pNumTagFound *pHaveTag (Whether the TAG Type Identification is listed.) *pTAG (The list of TAG Type Identification. If pHaveTag is false, this is an array of serial number length of the cards detected. If pHaveTag is true, this is an array of Tag type. The corresponding serial number length could then be determined from the Tag type.) *pSN(The flat array of serial numbers. All serial numbers are concatenated with length of either 4, 7 or 10 numbers. The lengths are indicated in pTag field) **ACR120U parameters:** *hReader *pNumTagFound * pTagType[4] (Contains the TAG Type) * pTagLength[4] (Contains the length of the serial number) * pSN[4][10] (The flat array of serial numbers. All serial numbers are concatenated with fixed length – 10 bytes.)

ACR120_MultiTagSelect(rHandle, stationID, pSN[10], pHaveTag, pTAG, pResultSN[10]);	ACR120_MultiTagSelect(hReader, TagLength, SN[10], pResultTagType, pResultTagLength, pResultSN);	**ACR120S parameters:** *rHandle *stationID *pSN[10] (Contains the serial number of the TAG to be selected) *pHaveTag (Whether the TAG Type Identification of selected tag is returned..) *pTAG (The list of TAG Type Identification. If pHaveTag is false, this is an array of serial number length of the cards detected. If pHaveTag is true, this is an array of Tag type. The corresponding serial number length could then be determined from the Tag type.) *pResultSN(The serial number of selected TAG) **ACR120U parameters:** *hReader * TagLength (Contains the length of the serial number of the TAG to be selected.) * SN[10] (Contain the serial number of the TAG to be selected) * pResultTagType (Contain the selected Tag Type) * pResultTagLength (Contain the length of the serial number of the selected TAG) * pResultSN (The serial number of the selected TAG.)

ACR120_TxDataTelegram (rHandle, stationID, length, bParity, bOddParity, bCRCGen, bCRCCheck, bCryptoInactive, bitFrame, data, pRecvLen, recvData);	ACR120_TxDataTelegram(hReader, SendDataLength, pSendData pReceivedDataLength, pReceivedData);	**ACR120S parameters:** *rHandle *stationID *length (The length of user specific data frame) * bParity (TRUE if parity generation is enabled) * bOddParity (TRUE if parity is odd. Otherwise it's even) * bCRCGen (TRUE if CRC generation for transmission is enabled) * bCRCCheck (TRUE if CRC checking for receiving is enabled) * bCryptoInactive (TRUE if Crypto unit is deactivated before transmission start) * bitFrame (number of bits from last byte transmitted) * data (contains the user specific data frame) * pRecvLen (it returns the length of response data receive) * recvData (contains the response data receive) **ACR120U parameters:** * hReader * SendDataLength (the length of data to be sent) * pSendData (the data to be sent) * pReceivedDataLength (the length of received data) * pReceivedData (the received data)
ACR120_RequestDLLVersion(pVersionInfoLen, pVersionInfo);	ACR120_RequestDLLVersion(pVersionInfoLength, pVersionInfo);	**ACR120S parameters:** * pVersionInfoLen (It returns the length of the DLL Version string.) * pVersionInfo **ACR120U parameters:** * pVersionInfoLength (It returns the length of the DLL Version string.) * pVersionInfo

N/A	ACR120_Status(hReader, pFirmwareVersion[20], pReaderStatus);	**New Function for ACR120U** **It Returns the firmware version and the Reader status.** Parameters: * hReader * pFirmwareVersion[20] (The firmware version will be returned (20 bytes)) * pReaderStatus (The Reader status.) *See ACR120U API Documentation for more details*
N/A	ACR120_DirectSend (hReader, DataLength pData, pResponseDataLength, pResponseData, TimedOut);	**New Function for ACR120U** **To send data to the Mifare Chip directly.** **Parameters:** * hReader * DataLength (The Data Length (maximum 66 bytes)) * pData (The Data to be sent) * pResponseDataLength (The Response Data Length) * pResponseData (The Response Data) * TimedOut (The Time Out for waiting the response data in m-sec) *See ACR120U API Documentation for more details*
N/A	ACR120_DirectReceive(hReader, RespectedDataLength, pReceivedDataLength, pReceivedData, TimedOut);	**New Function for ACR120U** **To receive data from the Mifare Chip directly.** **Parameters:** * hReader * RespectedDataLength (The Respected Data Length to be received (maximum 64 bytes)) * pReceivedDataLength (The Data Length of the received data) * pReceivedData (The Received Data) * TimedOut (The Time Out for waiting the received data in m-sec) *See ACR120U API Documentation for more details*

III. Error Codes

ACR120 (Serial Interface)	
Error Codes	**Description**
ERR_ACR120_INTERNAL_UNEXPECTED(-1000)	Library internal unexpected error.
ERR_ACR120_PORT_INVALID(-2000)	The port is invalid.
ERR_ACR120_PORT_OCCUPIED(-2010)	The port is occupied by another application.
ERR_ACR120_HANDLE_INVALID(-2020)	The handle is invalid.
ERR_ACR120_INCORRECT_PARAM(-2030)	Incorrect Parameter.
ERR_ACR120_READER_NO_TAG(-3000)	No TAG in reachable range / selected.
ERR_ACR120_READER_READ_FAIL_AFTER_OP(-3010)	Read fail after operation.
ERR_ACR120_READER_NO_VALUE_BLOCK(-3020)	Block doesn't contain value.
ERR_ACR120_READER_OP_FAILURE(-3030)	Operation failed.
ERR_ACR120_READER_UNKNOWN(-3040)	Reader unknown error.
ERR_ACR120_READER_LOGIN_INVALID_STORED_KEY_FORMAT(-4010)	Invalid stored key format in login process.
ERR_ACR120_READER_WRITE_READ_AFTER_WRITE_ERROR(-4020)	Reader can't read after write operation.
ERR_ACR120_READER_DEC_FAILURE_EMPTY(-4030)	Decrement failure (empty).

ACR120 (USB Interface)	
Error Codes	**Description**
ERR_INTERNAL_UNEXPECTED(-1000)	Library internal unexpected error. *#Handled by the DLL*
ERR_PORT_INVALID(-2000)	The port is invalid. *#Handled by the DLL*
ERR_PORT_OCCUPIED(-2010)	The port is occupied by another application. *#Handled by the DLL*
ERR_HANDLE_INVALID(-2020)	The handle is invalid. *#Handled by the DLL*
ERR_INCORRECT_PARAM(-2030)	Incorrect Parameter. *#Handled by the DLL.*
ERR_READER_NO_TAG(-3000, or 0xF448)	No TAG in reachable range / selected. *#Corresponding to the << Response Status 'N' >>.*
ERR_READER_OP_FAILURE(-3030, or 0xF42A)	Operation failed. *#Corresponding to the << Response Status 'F' >>.*
ERR_READER_UNKNOWN(-3040, or 0xF420)	Reader unknown error. *#Corresponding to the << Response Status 'C', 'O', 'X' & '?' >>.*
ERR_READER_LOGIN_INVALID_STORED_KEY_FORMAT(-4010, or 0xF056)	Invalid stored key format in login process. *#Handled by the DLL.*
ERR_READER_LOGIN_FAIL(-4011, or 0xF055)	Login failed. *#Corresponding to the << Response Status 'I' >>.*
ERR_READER_OP_AUTH_FAIL(-4012, or 0xF054)	The operation or access is not authorized. *#Corresponding to the << Response Status 'I' >>.*
ERR_READER_VALUE_DEC_EMPTY(-4030, or 0xF042)	Decrement failure (empty). *#Corresponding to the << Response Status 'E' >>.*
ERR_READER_VALUE_INC_OVERFLOW(-4031, or 0xF041)	Increment Overflow. *#Corresponding to the << Response Status 'E' >>.*
ERR_READER_VALUE_OP_FAILURE (-4032, 0xF040)	Value Operations failure. E.g. Value Increment *#Corresponding to the << Response Status 'I' >>.*
ERR_READER_VALUE_INVALID_BLOCK(-4033, 0xF03F)	Block doesn't contain value. *#Corresponding to the << Response Status 'F' >>.*
ERR_READER_VALUE_ACCESS_FAILURE (-4034, 0xF03E)	Value Access failure. *#Corresponding to the << Response Status 'U' >>.*