Advanced Card Systems Limited

Card and Reader Technologies

API REFERENCE MANUAL

ACR88 Handheld Portable Smart Card Reader

Contents

1. Introduction

This manual describes the use of ACR88 software programming interface to control the built-in accessories of the ACR88 multi-function card reader.

Built-in accessories are defined to be the keypad, LCD display, LEDs, buzzer and real-time clock, embedded in ACR88. Such components are not controlled through the smart card reader library.

The programming interfaces are implemented as a dynamic link library (or DLL). We will use the term ACR88 DLL to refer to the interfaces in the following text. This ACR88 DLL is based on the C programming language and is available on Windows XP and 2000. The name of the DLL is acr88.dll and the functions described in this document can be found in acr88.h, the header file that exposes the functions to be used by applications.

1.1 Architecture

The architecture of the ACR88 library can be visualized as the following diagram:

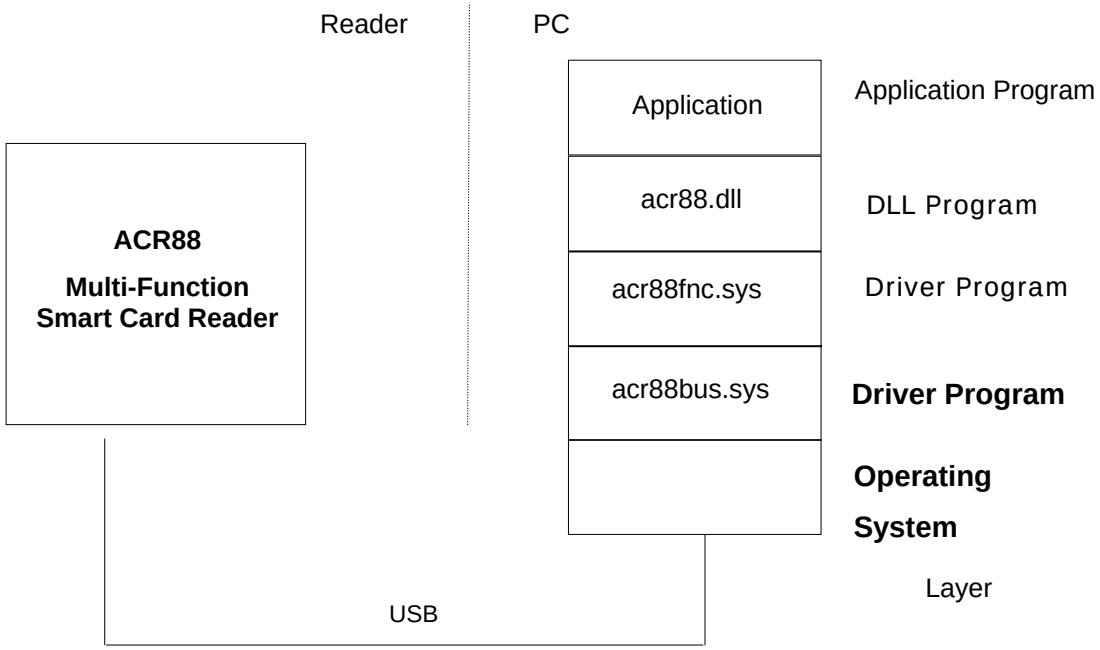


Figure 1.1

1.2 PC/SC API Dependency

ACR88 DLL is implemented as a library completely independent of the PC/SC sub-system of Windows. The library does not use any PC/SC API to communicate between built-in accessories of ACR88 and the application program.

2. ACR88 API Declarations

2.1 Enumerators

2.1.1 Port numbers

```
enum
{
        AS_USB1                 = 0x00,
        AS_USB2                 = 0x01,
        AS_USB3                 = 0x02,
        AS_USB4                 = 0x03,
        AS_USB5                 = 0x04,
        AS_USB6                 = 0x05,
        AS_USB7                 = 0x06,
        AS_USB8                 = 0x07
};
```

Used by AS_Open to select the USB port where the ACR88 reader is connected. Up to eight USB ports can be selected.

2.1.2 LCD_CLEAR_MODE

```
typedef enum _LCD_CLEAR_MODE {
        LCD_CLR_FULL        = 0x00,
        LCD_CLR_ROWS        = 0x01,
        LCD_CLR_COLS        = 0x02
} LCD_CLEAR_MODE;
```

Used by AS_ClearLCDDisplay to select the mode for clearing the LCD display.

Data Member	Value	Description
LCD_CLR_FULL	0x00	Clear the full LCD Screen
LCD_CLR_ROWS	0x01	Clear one or more rows of the LCD screen
LCD_CLR_COLS	0x02	Clear one or more columns of the LCD screen

2.1.3 LED_OPTION

```
typedef enum _LED_OPTION {
        LED_UNCHANGED     = 0x00,
        LED_OFF           = 0x01,
        LED_RED           = 0x02,
        LED_GREEN         = 0x03,
        LED_YELLOW        = 0x04
} LED_OPTION;
```

Used by AS_SetLED to set the color of one of the three LED's on the ACR88.

Data Member	Value	Description
LED_UNCHANGED	0x00	Do not change the color of the LED.
LED_OFF	0x01	Turn the LED off.
LED_RED	0x02	Switch the LED on and make it red.
LED_GREEN	0x03	Switch the LED on and make it green.
LED_YELLOW	0x04	Switch the LED on and make it yellow.

2.1.4 EEPROM_ACCESS

```
typedef enum _EEPROM_ACCESS {
        READ_EEPROM             = 0x00,
        WRITE_EEPROM      = 0x01
} EEPROM_ACCESS;
```

Used by AS_AccessEEProm to select reading or writing from/to the internal EEProm of the ACR88.

Data Member	Value	Description
READ_EEPROM	00 H	Read data from the EEProm
WRITE_EEPROM	01 H	Write data from the EEProm

\

2.2 Reader Command Data Structures

2.2.1 INITDEVICE

```
typedef struct _INIT_DEVICE {
        CHAR        szSerialNum[9];
        CHAR        szAuthKey[16];
        BYTE        cbDesType;
        BYTE        cbKeyIdx;
} INITDEVICE, *PINITDEVICE;
```

Used by AS_InitDevice.

Data Member	Value	Description
szSerialNum	9 bytes ASCII-value serial number	Contains the serial number to be set in ACR88.
szAuthKey	16 bytes HEX-value key	Manufacturer secret key challenge response and vendor injected secret key (Fills null string if not N/A)
cbDesType	0 or 1	0 – Single length key 1 – Double length key
cbKeyIdx	00_H to 06_H	Key index 00_H – 06_H

2.2.2 KEYPADCONFIG

```
typedef struct _KEYPAD_CONFIG {
        BYTE        cbMaxKeyString;
        BYTE        KeyDisplayRow;
} KEYPADCONFIG, *PKEYPADCONFIG;
```

Used by AS_ConfigureKeyPad.

Data Member	Value	Description
cbMaxKeyString	00_H to $0F_H$	Maximum number of keys allowed for a key string in key string input mode (see PC_to_ACR88_InputKey command).
KeyDisplayRow	00_H to 03_H	Starting row number on the LCD for displaying the keys input.

2.2.3 KEYPADINPUT

```
typedef struct _KEYPAD_INPUT{
        BOOLEAN         bEnableKeyString;
        BOOLEAN         bEnableAlphanumeric;
        BOOLEAN         bEnableKeyDisplay;
        BOOLEAN         bEnableMaskedDisplay;
        BOOLEAN         bDisableTimeout;
        BOOLEAN         bEnableKeyEncryption;
        BOOLEAN         bEnableControlKeys;
        BOOLEAN         bReserved2;
        BYTE            cbTimeout;
} KEYPADINPUT, *PKEYPADINPUT;
```

Used by AS_GetKeyInput.

Data Member	Value	Description
BEnableKeyString	0 or 1	Input Mode 0 – single key input 1 – key string input (In key string input mode, the key string input is completed when the “Enter” key is pressed.)
BEnableAlphanumeric	0 or 1	Keyboard Mode 0 – numeric input 1 – alphanumeric input
BEnableKeyDisplay	0 or 1	Key Display Mode 0 – key display disabled 1 – key display enabled
BEnableMaskedDisplay	0 or 1	Key Masked Display Mode 0 – key display as plaintext 1 – key display as ‘*’
BDisableTimeout	0 or 1	Enable or disable key input timeout 0 – enable timeout 1 – disable timeout
BEnableKeyEncryption	0 or 1	Secure key transfer 0 – plaintext transfer 1 – encrypted key transfer (RFU)
BEnableControlKeys	0 or 1	Enable of disable Control Keys (F1~F4 & directional keys) 0 – disable control keys 1 – enable control keys
bReserved2	-	RFU
CbTimeout	0 to 255	Key input timeout time value counted in 100ms (e.g. 100 stands for 10 seconds).

2.2.4 LCDCURSOR

```
typedef struct _LCD_CURSOR {
        BYTE            cbRowPosition; // 0 - 7
        BYTE            cbColPosition;   // 0 - 83
} LCDCURSOR, *PLCDCURSOR;
```

Used in AS_SetLcdCursor to position the cursor position on the LCD screen

Data Member	Value	Description
cbRowPosition	00_H to 07_H	cursor row position
cbColPosition	00_H to 80_H	cursor column position

2.2.5 LCDBACKLIGHT

```
typedef struct _LCD_BACKLIGHT {
        BOOLEAN     bEnableBackLight;
} LCDBACKLIGHT, *PLCDBACKLIGHT;
```

Used by AS_SetLcdBacklight to enable or disable the LCD backlight.

Data Member	Value	Description
bEnableBackLight	0 or 1	0 - turns off backlight 1 - turns on backlight

2.2.6 LCDGRAPHICS

```
typedef struct _LCD_GRAPHICS {
        LPCTSTR     szBitmapFile;
} LCDGRAPHICS, *PLCDGRAPHICS;
```

Used by AS_SetLcdDisplayGraphics.

Data Member	Value	Description
szBitmapFile	-	Full path to a Windows bitmap file to be displayed. The Dimension of the bitmap can be any size within a range of 128 pixels wide by 64 pixels high and the color depth can be 1-bit, 8-bit or 24-bit. *Note: the LCD screen of the ACR88 only displays monochrome graphics.

2.2.7 LCDMESSAGE

```
typedef struct _LCD_MESSAGE {
```

```
        BYTE        cbCharCoding;
        LPCTSTR     pMessage;
        USHORT      wMessageLen;
} LCDMESSAGE, *PLCDMESSAGE;
```

Used by AS_SetLcdDisplayMessage.

Data Member	Value	Description
cbCharCoding	00_H	Data encoding format used in the pMessage field. Character size depends on data format. 00_H – ASCII All other values are RFU
pMessage	ASCII String	Character string of encoding format stated in cbCharCoding field
wMessageLen	Positive Integer	The number of characters stored in pMessage

2.2.8 LCDCONTRAST

```
typedef struct _LCD_CONTRAST {
        BYTE        cbContrastLevel;
} LCDCONTRAST, *PLCDCONTRAST;
```

Used by AS_SetLcdSetContrast to set the contrast of the LCD screen.

Data Member	Value	Description
cbContrastLevel	00_H to 63_H	New LCD contrast level

2.2.9 LCDCLEAR

```
typedef struct _LCD_CLEAR {
        BYTE        cbClearMode;
        BYTE        cbNumber;
} LCDCLEAR, *PLCDCLEAR;
```

Used by AS_ClearLcdDisplay to clear (part of) the LCD screen.

Data Member	Value	Description
cbClearMode	LCD_CLEAR_MODE	LCD_CLR_FULL = Clear the complete LCD screen LCD_CLR_ROWS = Clear rows LCD_CLR_COLS = Clear columns
cbNumber	LCD_CLR_ROWS	LCD_CLR_ROWS = Number of rows to be cleared * LCD_CLR_COLS = Number of columns to be cleared * *Ignored in LCD_CLR_FULL mode.

2.2.10 LED

```
typedef struct _LED {
        BYTE        cbLedPower;         // see LED_OPTION
        BYTE        cbLedSlot1;         // see LED_OPTION
        BYTE        cbLedSlot2;         // see LED_OPTION
} LED, *PLED;
```

Used by AS_SetLED to control the LED's of the ACR88.

Data Member	Value	Description
CbLedPower	LED_OPTION	Control the Power LED LED_UNCHANGED – Do not change LED LED_OFF – Turn LED off LED_RED – Turn LED red LED_GREEN- Turn LED green LED_YELLOW – Turn LED yellow
cbLedSlot1	LED_OPTION	Control the LED of card slot 1 For possible options see LED_OPTION and above.
cdLedSlot2	LED_OPTION	Control the LED of card slot 2 For possible options see LED_OPTION and above.

2.2.11 BUZZER

```
typedef struct _BUZZER {
        BYTE        cbBuzzerState;
        BYTE        cbBuzzerOnDuration;
} BUZZER, *PBUZZER;
```

Used in AS_SetBuzzer.

Data Member	Value	Description
cbBuzzerState	0 or 1	0 – Buzzer off 1 – Buzzer on
cbBuzzerOnDuration	0 - 255	Duration of buzzer on counted in 100ms (e.g. 100 stands for 10 seconds).

2.2.12 SCRIPT

```
typedef struct _SCRIPT {
        LPCTSTR szBinFile;
} SCRIPT, *PSCRIPT;
```

Used by AS_LoadScript.

Data Member	Value	Description
szBinFile	-	Full path to a binary script file. This is a script that has been compiled using the ACR88ScriptBuilder.

2.3 Reader Response Data Structures

2.3.1 AS_STATUS

```
typedef struct _AS_STATUS {
        DLL_ERROR           DllError;
        LONG                W32Error;
} AS_STATUS;
```

Data Member	Value	Description
DllError	00_H – 20_H	Contains the error code set by the DLL during command execution. See also Appendix A. Error Codes
W32Error	Win32 Error Code	Contains the error code set by Windows system during the execution of Win32 API.

2.3.2 INFO

```
typedef struct _INFO {
        USHORT      szFirmwareVersion[8];
        CHAR        szSeriaNum[9];
        BYTE        cbDeviceMode;
        BYTE        cbVendor;
        DWORD           dwAppSignature;
        DWORD           dwReaderCapabilities;
        USHORT      wSmartCardSupported;
} INFO, *PINFO;
```

Returned by AS_GetInfo, contains details of the ACR88 reader and it capabilities.

Data Member	Value	Description
szFirmwareVersion	8 bytes ASCII version number	Firmware version in ASCII
szSeriaNum	9 bytes ASCII serial number	Serial number of this device Fills null if not applicable
cbDeviceMode	01_H, 02_H, 10_H or FF_H	FFh – Device not initialized 01h – PC-Linked normal mode 02h – PC-Linked script debug mode 10h – Standalone normal mode Other values are RFU
cbVendor	1 byte	When the device is initialized to a vendor, this byte is freely use by vendor for identification. Value is vendor specific.
dwAppSignature	4 bytes	Application signature of the loaded script application. 00000000_H = No script application is loaded
dwReaderCapabilities	4 bytes	Reader capabilities in bitmap format with each bit representing a particular capability (“0” represents capability is disabled, while “1” represents capability is enabled)

		Bit 0: USB (PC-linked mode) Bit 1: Standalone Mode Bit 2: LCD Bit 3: Keypad Bit 4: Buzzer Bit 5: RTC Bit 6: Smart Card 1 Bit 7: Smart Card 2 Bit 8: SAM 1 Bit 9: SAM 2 Bit 10: SAM 3 Bit 11: Script capability Bit 12: TFM Bit 13: Contactless Bit 14: Serial Others: RFU
wSmartCardSupported	2 bytes	Smart cards supported by the reader in bitmap format with each bit representing a particular protocol or card type (“0” represents unsupported protocols or smart cards, while “1” represents supported protocols or smart cards.) Bit 0: T=0 protocol Bit 1: T=1 protocol Bit 2: I2C with memory <= 16k bits Bit 3: I2C with memory >= 32k bits Bit 4: SLE4406/4436/5536 Bit 5: SLE4418/4428 Bit 6: SLE4432/5542 Bit 7: Mifare Contactless Bit 8: Type A Contactless Bit 9: Type B Contactless Others: RFU

2.3.3 KEYPADSTATUS

```
typedef struct _KEYPAD_STATUS {
        BYTE            cbMaxKeyString;
        BYTE            cbKeyDisplayMode;
} KEYPADSTATUS, *PKEYPADSTATUS;
```

Returned by AS_GetKeyPadConfig and AS_ConfigureKeyPad.

Data Member	Value	Description
cbMaxKeyString	0 – 255	Maximum number of keys allowed for a key string in key string input mode (see AS_InputKey command).
cbKeyDisplayMode	0 – 7	Starting row number on the LCD for displaying the keys input.

2.3.4 DISPLAYSTATUS

```
typedef struct _DISPLAY_STATUS {
        BYTE        cbRowPosition;
        BYTE        cbColumnPosition;
} DISPLAYSTATUS, *PDISPLAYSTATUS;
```

Returned by AS_SetLcdCursor, AS_SetLcdBacklight, AS_SetLcdDisplayGraphics, AS_SetLcdDisplayMessage, AS_SetLcdSetContrast and AS_ClearLcdDisplay.

Data Member	Value	Description
cbRowPosition	0 - 7	Current cursor row position
cbColumnPosition	0 - 127	Current cursor column position

2.3.5 TESTREPORT

```
typedef struct _TEST_REPORT {
        USHORT      wFirmwareVersion;
        CHAR        szSerialNum[9];
        USHORT      wResult;
} TESTREPORT, *PTESTREPORT;
```

Data Member	Value	Description
wFirmwareVersion	2 bytes BCD version	Firmware version in binary-coded decimal (e.g. 3.80 is 0380h)
szSerialNum	9 bytes ASCII serial number	Serial number of this device
wResult		Each bit in this field represents the test result of a particular tested function. (bit=0 for pass, bit=1 for fail) Bit-function mapping to be determined.

2.3.6 DEBUGENV (Definition of API is at preliminary stage)

```
typedef struct _DEBUG_ENV {
        DWORD       dwScriptIP;
        DWORD       dwMemDumpSize;
        PBYTE pMemDump;
} DEBUGENV, *PDEBUGENV;
```

Data Member	Value	Description
dwScriptIP	DWORD (4 bytes)	Contents to be determined
dwMemDumpSize	DWORD (4 bytes)	Contents to be determined
pMemDump		Contents to be determined

2.4 Reader Shared Command/Response Data Structures

2.4.1 TIMESTAMP

```
typedef struct _TIMESTAMP {
        CHAR            szRTCValue[6];
} TIMESTAMP, *PTIMESTAMP;
```

Used in AS_ReadRTC and AS_SetRTC to retrieve or set the value of the run time clock of the ACR88.

Data Member	Value	Description
szRTCValue[0]	00 – 99	Year (short format)
szRTCValue[1]	1 – 12	Month
szRTCValue[2]	1 – 31	Day
szRTCValue[3]	1 – 23	Hours
szRTCValue[4]	0 – 59	Minutes
szRTCValue[5]	0 - 59	Seconds

2.4.2 DATABLOCK

```
typedef struct _DATA_BLOCK {
        USHORT          wDataLen;
        PBYTE           pDataBlock;
} DATABLOCK, *PDATABLOCK;
```

Returned by AS_GetKeyInput, AS_InitScriptMode and AS_StepScript, contains the data returned by those functions.

Data Member	Value	Description
wDataLen		The length of the size of pDataBlock before command execution. Stores the length of ACR88 returned data block after command execution.
pDataBlock		The data to input to a command or the data returned by ACR88.

2.4.3 ACCESSEEPROM

```
typedef struct _ACCESS_EEPROM {
        BYTE            cbFunction;
        BYTE            cbDeviceNumber;
        DWORD                   dwAddress;
        USHORT          wDataLength;
        PBYTE           pData
} ACCESSEEPROM, *PACCESSEEPROM;
```

Data Member	Value	Description
cbFunction	EEPROM_ACCESS	00_H – READ_EEPROM 01_H – WRITE_EEPROM
cbDeviceNumber	00_H or 01_H	00_H – Slave EEPROM 01_H – Chinese font EEPROM
dwAddress	4 byte double word (hex)	Address of EEPROM
wDataLength	2 byte word (hex)	Length of Data (Write/Read)
pData	Pointer to buffer of wDataLength	Read EEPROM: pointer to buffer to store the read EEPROM data. Write EEPROM: pointer to buffer containing data to write to EEPROM

3. ACR88 API Functions

3.1 General Description

All functions return a status code AS_STATUS, which is a structure consisting of a DLL defined error and a WIN32 error. See also chapter 2.3.1 for more information about the AS_STATUS structure. Code AS_STATUS.DllError == CMD_SUCCESS means success. AS_STATUS.W32Error is defined and used to provide additional error information to developers only when AS_STATUS.DllError != CMD_SUCCESS.

The API functions are classified into 7 categories according to the type of accessories they will control as follows:

- Port Functions
- Device Functions
- LCD Functions
- Keypad Functions
- Real-Time Clock Functions
- Script Functions
- Other Functions

3.2 Port Functions

The Port Functions allow the creation and closure of the logical connection to ACR88.

3.2.1 AS_Open

This function opens a logical connection to ACR88. This function must be called before calling any other API function.

```
AS_STATUS AS_DECL AS_Open (
                        IN      INT nReaderType,
                        IN      INT nPort,
                        OUT     INT *nDevId);
```

Parameters:

nReaderType

[in] Must be ACR88 (00_H, as defined in acr88.h).

nPort

[in] The instance of the reader connected to USB port. E.g. AS_USB1 refers to the first connected ACR88 detected by the PC. See also chapter 2.1.1 Port numbers for possible options.

nDevId

[out] Handle to be returned upon successful creation of the connection. This handle will be used in all the subsequent calls to other API functions.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
INT             nDid;
AS_STATUS       status;

//open a connection to the ACR88
status = AS_Open(ACR88,AS_USB1,&nDid);
if(status.DllError == CMD_SUCCESS) {
        // connection success, do something with the ACR88

}
Else {
        // error occurred
        return status;
}
```

3.2.2 AS_Close

This function closes a logical connection to ACR88.

```
AS_STATUS AS_DECL AS_Close (
                        IN      INT nDevId);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
INT             nDid;
AS_STATUS       status;

//open a connection to the ACR88
status = AS_Open(ACR88, AS_USB1, &nDid);
if(status.DllError == CMD_SUCCESS) {
        //connection success, do something
        .
        .
        .
        //done, close connection
        status = AS_Close(nDid);
}
else {
        //error occurred
        return status;
}
```

3.3 Device Functions

The Device Functions allow the initialization and retrieval of various parameters to and from the ACR88 as well as performing a self-test to the device.

3.3.1 AS_InitDevice (Definition of API is at preliminary stage)

This function personalizes the device for release of a specific vendor. This is done as the first step after manufacturing or to recover the device after tampering.

```
AS_STATUS AS_DECL AS_InitDevice (
                        IN      INT nDevId,
                        IN      PINITDEVICE pInitDev,
                        OUT     PINFO pInfo);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pInitDev

[in] Pointer to an *INITDEVICE* structure that includes the parameters to be initialized. See also chapter 2.2.1 INITDEVICE for more information about the *INITDEVICE* structure.

pInfo

[out] Pointer to an *INFO* structure that saves the original parameters of the device. See also chapter 2.3.2 INFO for more information about the *INFO* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example: (Not available for preliminary API)

```
N/A
```

3.3.2 AS_GetInfo

This function retrieves general information of the ACR88.

```
AS_STATUS AS_DECL AS_GetInfo (
                        IN      INT nDevId,
                        OUT     PINFO pInfo);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pInfo

[out] Pointer to an *INFO* structure that saves the general information of the ACR88 device. See also chapter 2.3.2 INFO for more information about the *INFO* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
INT             nDid;
INFO            Info;
AS_STATUS       status;

//open a connection to the ACR88
status = AS_Open(ACR88,AS_USB1,&nDid);
if(status.DllError == CMD_SUCCESS) {

        //connection success, get the reader information.
        status = AS_GetInfo(nDid,&Info);
        if (status.DLLError == CMD_SUCCESS) {
                //do something with the retrieved information
        }

        //close the connection
        status = AS_Close(nDid);
}
else {
        return status;
}
```

3.3.3 AS_SelfTest (Definition of API is at preliminary stage)

This function is reserved for future implementation.

```
AS_STATUS AS_DECL AS_SelfTest (
                        IN      INT nDevId,
                        OUT     PTESTREPORT pTestReport);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pInfo

[out] Pointer to a *TESTREPORT* structure that saves the test result of the device. See also chapter 2.3.5 TESTREPORT for more information about the *TESTREPORT* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example: (Not available for preliminary API)

```
N/A
```

3.3.4 AS_AccessEEPRom

This function allows user to write or read data to/from the EEPROM. Maximum allowed data length is 256 bytes.

```
AS_STATUS AS_DECL AS_AccessEEPRom (
                        IN      INT nDevId,
                        IN      PACCESSEEPROM pEEPRom);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pEEPRom

[in] Pointer to an *ACCESSEEPROM* structure that contains the data to be written to ACR88. or the data read from the ACR88. See also chapter 2.4.3 ACCESSEEPROM for more information about the *ACCESSEEPROM* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
INT                     nDid;
ACCESSEEPROM            eeprom;
BYTE                    aData[256];
AS_STATUS               status;

//read the EEPROM data from address 0x0000
//assumed is a connection has already been established eeprom.cbAccessMode  = READ_EEPROM;
eeprom.wAddress                 = 0x0000;
eeprom.wDataLength      = 0x0100;
eeprom.pData            = aData;
status = AS_AccessEEPRom(nDid, &eeprom);
if (status.DLLError == CMD_SUCCESS) {
        //do something with the data read
}
else {
        //error occurred
}
return status;
```

3.4 LCD Functions

The LCD Functions control the contrast, backlight status and the cursor position of the LCD panel. They are also used to display graphics and text on the LCD panel.

3.4.1 AS_SetLcdCursor

This function sets the LCD position cursor to a new position.

```
AS_STATUS AS_DECL AS_SetLcdCursor (
                        IN      INT nDevId,
                        IN      PLCDCURSOR pLcdCursor,
                        OUT     PDISPLAYSTATUS pDisplayStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pLcdCursor

[in] Pointer to a *LCDCURSOR* structure that includes the cursor position to be set. See also chapter 2.2.4 LCDCURSOR for more information about the *LCDCURSOR* structure.

pDisplayStatus

[out] Pointer to a *DISPLAYSTATUS* structure that saves the newly set position parameters. See also chapter 2.3.4 DISPLAYSTATUS for more information about the *DISPLAYSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
LCDCURSOR           lcdCursor;
DISPLAYSTATUS       displayStatus;
AS_STATUS           status;

//position the cursor at the upper left corner of the LCD
//assumed is a connection has already been established
lcdCursor.cbColPosition    = 0;
lcdCursor.cbRowPosition    = 0;

status = AS_SetLcdCursor(nDid, &lcdCursor, &displayStatus);
```

3.4.2 AS_SetLcdBacklight

This function turns the backlight of the LCD on or off.

```
AS_STATUS AS_DECL AS_SetLcdBacklight (
                        IN      INT nDevId,
                        IN      PLCDBACKLIGHT pLcdBacklight,
                        OUT     PDISPLAYSTATUS pDisplayStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pLcdBacklight

[in] Pointer to a *LCDBACKLIGHT* structure that includes the cursor position parameters to be set. See also chapter 2.2.5 LCDBACKLIGHT for more information about the *LCDBACKLIGHT* structure.

pDisplayStatus

[out] Pointer to a *DISPLAYSTATUS* structure containing the cursor position after the action. See also chapter 2.3.4 DISPLAYSTATUS for more information about the *DISPLAYSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

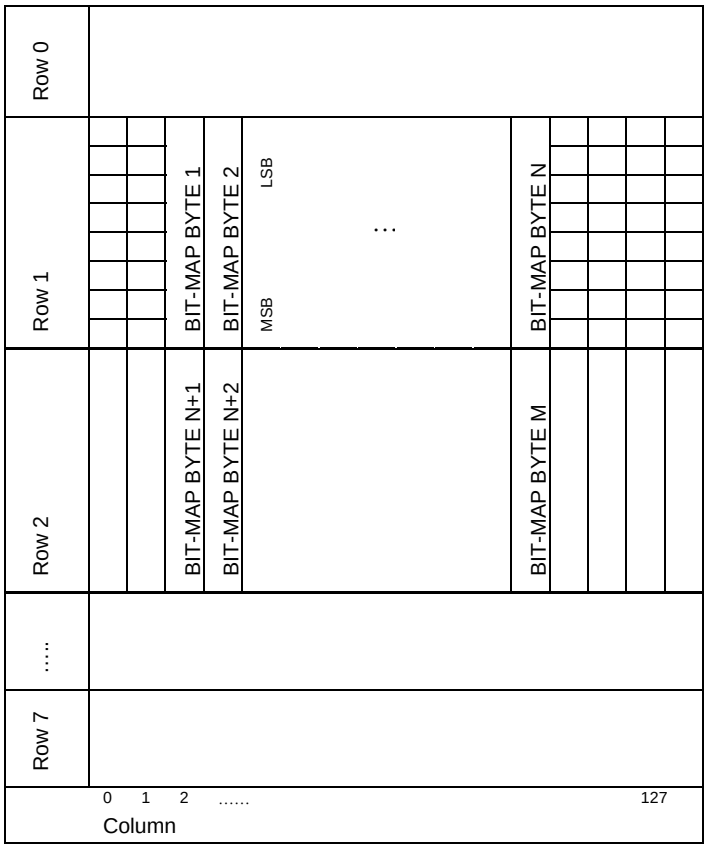
Example:

```
LCDBACKLIGHT lcdLight;
DISPLAYSTATUS       lcdStatus;
AS_STATUS           status;

//turn on the LCD backlight
//assumed is that a connection has already been established.
lcdLight.bEnableBackLight = TRUE;
status = AS_SetLcdBacklight(nDid, &lcdLight, &lcdStatus);
```

3.4.3 AS_SetLcdDisplayGraphics

This function transfers bit-map graphics to ACR88 and display the graphics on the LCD from the current cursor position. The bit-map format is shown in the diagram below. The cursor will be moved to the position next to the lower right corner of the graphic after executing this command. The maximum dimensions for the bitmap are 128 pixels wide by 64 pixels high.




```
AS_STATUS AS_DECL AS_SetLcdDisplayGraphics (
                        IN      INT nDevId,
                        IN      PLCDGRAPHICS pLcdGraphics,
                        OUT     PDISPLAYSTATUS pDisplayStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pLcdGraphics

[in] Pointer to a *LCDGRAPHICS* structure that specifies the path to the bitmap file. See also chapter 2.2.6 LCDGRAPHICS for more information about the *LCDGRAPHICS* structure.

pDisplayStatus

[out] Pointer to a *DISPLAYSTATUS* structure containing the cursor position after displaying the graphics. See also chapter 2.3.4 DISPLAYSTATUS for more information about the *DISPLAYSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
LCDGRAPHICS         lcdGrafx;
DISPLAYSTATUS       lcdStatus;
AS_STATUS           status;

//display a bitmap file called “MyLogo.bmp”.
//assumed is that a connection has already been established.
lcdGrafx.szBitmapFile     = “MyLogo.bmp”;
status = AS_SetLcdDisplayGraphics(nDid, &lcdGrafx, &lcdStatus);
```

3.4.4 AS_SetLcdDisplayMessage

This function displays a string of characters using the ACR88 built-in font library. The string will be displayed horizontally from the current cursor position. ACR88 will automatically calculate the absolute coordinates from the character position and character size and the cursor will be moved accordingly. When the text reaches the end of a display line, the text will be wrapped.

```
AS_STATUS AS_DECL AS_SetLcdDisplayMessage (
                        IN      INT nDevId,
                        IN      PLCDMESSAGE pLcdMessage,
                        OUT     PDISPLAYSTATUS pDisplayStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pLcdMessage

[in] Pointer to a *LCDMESSAGE* structure that specifies the alphanumeric text to be displayed on LCD. See also chapter 2.2.7 LCDMESSAGE for more information about the *LCDMESSAGE* structure.

pDisplayStatus

[out] Pointer to a *DISPLAYSTATUS* structure containing the cursor position after displaying the message. See also chapter 2.3.4 DISPLAYSTATUS for more information about the *DISPLAYSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
const char          szText[]="Welcome to the ACR88";
LCDMESSAGE          lcdMsg;
DISPLAYSTATUS       lcdStatus;
AS_STATUS           status;

//display the above text
//assumed is that a connection has already been established
lcdMsg.cbCharCoding     = 0x00;
lcdMsg.pMessage                 = szText;
lcdMsg.wMessageLen      = strlen(szText);
status = AS_SetLcdDisplayMessage(nDid, &lcdMsg, &lcdStatus);
```

3.4.5 AS_SetLcdSetContrast

This function sets the contrast level of the LCD.

```
AS_STATUS AS_DECL AS_SetLcdSetContrast (
                        IN      INT nDevId,
                        IN      PLCDCONTRAST pLcdContrast,
                        OUT     PDISPLAYSTATUS pDisplayStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pLcdContrast

[in] Pointer to a *LCDCONTRAST* structure that specifies the path to the bitmap file. See also chapter 2.2.8 LCDCONTRAST for more information about the *LCDCONTRAST* structure.

pDisplayStatus

[out] Pointer to a *DISPLAYSTATUS* structure containing the cursor position after displaying the graphics. See also chapter 2.3.4 DISPLAYSTATUS for more information about the *DISPLAYSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
LCDCONTRAST         lcdContrast;
DISPLAYSTATUS       lcdStatus;
AS_STATUS           status;

//Set the contrast of the LCD to 100%
//assumed is that a connection has already been established
lcdContrast. cbContrastLevel = 0x3f;
status = AS_SetLcdSetContrast (nDid, &lcdContrast, &lcdStatus);
```

3.4.6 AS_ClearLcdDisplay

This function clears one or more rows or columns on the LCD display. The cursor will be moved to the position at the starting point of the cleared block after executing this command.

```
AS_STATUS AS_DECL AS_ClearLcdDisplay (
                        IN      INT nDevId,
                        IN      PLCDCLEAR pLcdClear,
                        OUT     PDISPLAYSTATUS pDisplayStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pLcdClear

[in] Pointer to a *LCDCLEAR* structure that specifies the LCD clear mode. See also chapter 2.2.9 LCDCLEAR for more information about the *LCDCLEAR* structure.

pDisplayStatus

[out] Pointer to a *DISPLAYSTATUS* structure containing the cursor position after displaying the graphics. See also chapter 2.3.4 DISPLAYSTATUS for more information about the *DISPLAYSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
LCDCLEAR            lcdClear;
DISPLAYSTATUS       lcdStatus;
AS_STATUS           status;

//clear the full LCD screen
//assumed is that a connection has already been established
lcdClear.cbClearMode    = LCD_CLR_FULL;
lcdClear.cbNumber               = 0x00; //ignored

status = AS_ClearLcdDisplay(nDid, &lcdClear, &lcdStatus);
```

3.5 Keypad Functions

The Keypad Functions allow configuration of the keypad of the ACR88 and handling of key input.

3.5.1 AS_GetKeyPadConfig (Definition of API is at preliminary stage)

This function reads the current configuration of the keypad of the ACR88.

```
AS_STATUS AS_DECL AS_GetKeyPadConfig (
                        IN      INT nDevId,
                        OUT     PKEYPADSTATUS pKeypadStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pKeypadStatus

[out] Pointer to a *KEYPADSTATUS* structure that holds the current keypad configuration. See also chapter 2.3.3 KEYPADSTATUS for more information about the *KEYPADSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example: (Not available for preliminary API)

N/A

3.5.2 AS_ConfigureKeyPad (Definition of API is at preliminary stage)

This function configures the keypad of the ACR88.

```
AS_STATUS AS_DECL AS_ConfigureKeyPad (
                        IN      INT nDevId,
                        IN      PKEYPADCONFIG pKeypadConfig,
                        OUT     PKEYPADSTATUS pKeypadStatus);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pKeypadConfig

[in] Pointer to *KEYPADCONFIG* structure that specifies keypad configuration to set in ACR88 keypad. See also chapter 2.2.2 KEYPADCONFIG for more information about the *KEYPADCONFIG* structure.

pKeypadStatus

[out] Pointer to *KEYPADSTATUS* structure that saves the current keypad configuration. See also chapter 2.3.3 KEYPADSTATUS for more information about the *KEYPADSTATUS* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example: (Not available for preliminary API)

```
N/A
```

3.5.3 AS_GetKeyInput

This function enables key input on the ACR88. This can be single key input or string input depending on the options set in the KEYPADINPUT structure. The pressed keys will be returned in the following format:

Mode	Key	Value
Numeric	0 ~ 9	0 ~ 9
Alphanumeric	0 ~ 9	ASCII code
All modes	Clear	0x10
	Enter	0x0D
	F1	0x3D
	F2	0x3E
	F3	0x3F
	F4	0x0C

Notes:

- When a function key is pressed while in string input mode, the input is cancelled and the function code is returned instead.
- In string input mode, Enter will return the keys pressed and Clear will clear the last entered key.
- In string input more, when the inputted string has been cleared completely, AS_KeyInput will return with an empty string.

- The direction keys will never be returned, but are only used for navigation of the cursor on the ACR88 LCD screen.

```
AS_STATUS AS_DECL AS_GetKeyInput (
                        IN      INT nDevId,
                        IN      PKEYPADINPUT pKeypadInput,
                        OUT     PDATABLOCK pDataBlock);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pKeypadInput

[in] Pointer to a *KEYPADINPUT* structure that specifies the options to use when ACR88 captures key input. See also chapter 2.2.3 KEYPADINPUT for more information about the *KEYPADINPUT* structure.

pDataBlock

[out] Pointer to a *DATABLOCK* structure that contains the pressed keys (if any). See also chapter 2.4.2 *DATABLOCK* for more information about the *DATABLOCK* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
BYTE            aKeys[16];
KEYPADINPUT     kpInput;
DATABLOCK       dataBlk;
AS_STATUS       status;

//let the user input a string
//assumed is that a connection has already been established
dataBlk.pDataBlock = aKeys;
kpInput.bEnableKeyString  = TRUE;  //input a string
kpInput.bEnableAlphanumeric       = TRUE;  //string is alphanumeric
kpInput.bEnableKeyDisplay         = TRUE;  //display the keys on the LCD
kpInput.bEnableMaskedDisplay      = FALSE; //no masking of the keys
kpInput.bEnableControlKeys    = FALSE; //control keys disabled
kpInput.bDisableTimeout           = 1;          //no timeout
kpInput.bEnableKeyEncryption      = 0;        //no encryption of returned keys
status = AS_GetKeyInput(nDid,&kpInput,&data);
```

3.6 Real-Time Clock Functions

The Real-Time Clock Functions allow reading and setting of the built-in Run Time Clock of the ACR88.

3.6.1 AS_ReadRTC

This function reads the current real time clock value from the built-in real time clock. The real time clock increments the value every ½ second.

```
AS_STATUS AS_DECL AS_ReadRTC (
                        IN      INT nDevId,
                        OUT     PTIMESTAMP pTimeStamp);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pTimeStamp

[out] Pointer to *TIMESTAMP* structure that contains current time returned by the built-in real time clock of the ACR88. See also chapter 2.4.1 TIMESTAMP for more information about the *TIMESTAMP* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
AS_STATUS       status;
TIMESTAMP       tsRTC;
char            szTime[81];

//read the RTC of the ACR88
//assumed is that a connection has already been established
status = AS_ReadRTC(nDevId, &tsRTC);
if(status.DllError == CMD_SUCCESS)
        // display the returned date & time from the ACR88
        sprintf(szTime, "RTC: %d/%d/%d %d:%d:%d (yy/mm/dd hh:mm:ss)",
                tsRTC. szRTCValue[0], tsRTC. szRTCValue[1],
                tsRTC. szRTCValue[2], tsRTC. szRTCValue[3],
                tsRTC. szRTCValue[4], tsRTC. szRTCValue[5]);
        ::MessageBox(0L, szTime, "", MB_OK);
}
return status;
```

3.6.2 AS_SetRTC

This function sets the real time clock value of the built-in real time clock to the value specified in the TIMESTAMP structure.

```
AS_STATUS AS_DECL AS_SetRTC (
                        IN      INT nDevId,
                        IN      PTIMESTAMP pNewTime,
                        OUT     PTIMESTAMP pTimeStamp);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pNewTime

[in] Pointer to *TIMESTAMP* structure that contains the new time value to be set in the built-in real time clock. See also chapter 2.4.1 TIMESTAMP for more information about the *TIMESTAMP* structure.

pTimeStamp

[out] Pointer to *TIMESTAMP* structure that contains the newly set time value. See also chapter 2.4.1 TIMESTAMP for more information about the *TIMESTAMP* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
AS_STATUS       status;
TIMESTAMP       newTime;
TIMESTAMP       chkTime;

//set the RTC of the ACR88 using the values in the abTime array
//assumed is that a connection has already been established
CopyMemory(newTime.szRTCValue, abTime, 6);
Status = AS_SetRTC(pDevId, &newTime, &chkTime);
```

3.7 Script functions

3.7.1 AS_InitScriptMode (Definition of API is at preliminary stage)

```
AS_STATUS AS_DECL AS_InitScriptMode(
                        IN      INT nDevId,
                        IN      PDATABLOCK pDataBlock,
                        OUT     PDEBUGENV pDebugEnv);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pDataBlock

[in] Pointer to a DATABLOCK structure. See also chapter 2.4.2 DATABLOCK for more information about the *DATABLOCK* structure.

pDebugEnv

[out] Pointer to a *DEBUGENV* structure. See also chapter 2.3.6 DEBUGENV for more information about the *DEBUGENV* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

N/A

3.7.2 AS_StepScript (Definition of API is at preliminary stage)

```
AS_STATUS AS_DECL AS_StepScript(
                        IN      INT nDevId,
                        IN      PDATABLOCK pDataBlock,
                        OUT     PDEBUGENV pDebugEnv);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pDataBlock

[in] Pointer to a DATABLOCK structure. See also chapter 2.4.2 DATABLOCK for more information about the *DATABLOCK* structure.

pDebugEnv

[out] Pointer to a *DEBUGENV* structure. See also chapter 2.3.6 DEBUGENV for more information about the *DEBUGENV* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

N/A

3.7.3 AS_ExitScriptMode (Definition of API is at preliminary stage)

```
AS_STATUS AS_DECL AS_ExitScriptMode(
                        IN      INT nDevId,
                        OUT     PINFO pInfo);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pInfo

[out] Pointer to a *INFO* structure. See also chapter 2.3.2 INFO for more information about the *INFO* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

N/A

3.7.4 AS_LoadScript

Loads a binary script file into the memory of the ACR88. This file is a binary file compiled by the ACR88Scriptbuilder application.

```
AS_STATUS AS_DECL AS_LoadScript(
                        IN      INT nDevId,
                        IN      PSCRIPT pScript);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pScript

[out] Pointer to a *SCRIPT* structure. See also chapter 2.2.7 SCRIPT for more information about the *SCRIPT* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
AS_STATUS       status;
SCRIPT Script;

//Load a binary script file into the ACR88
//assumed is that a connection has already been established
Script.szBinFile = “c:\somewhere\script_to_load.bin”;
status = AS_LoadScript(pDevId, &Script);
```

3.8 Other Functions

The following functions allow the user to control the buzzer or the LEDs of the ACR88.

3.8.1 AS_SetBuzzer

This function enables or disables the buzzer of the ACR88.

```
AS_STATUS AS_DECL AS_ SetBuzzer (
                        IN      INT nDevId,
                        IN      PBUZZER pBuzzer);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pBuzzer

[in] Pointer to a *BUZZER* structure that contains the state to set of the buzzer. See also chapter 2.2.11 BUZZER for more information about the *BUZZER* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
AS_STATUS       status;
BUZZER          buzStat;

//turn the buzzer of the ACR88 on for 1 second
//assumed is that a connection has already been established
buzStat.cbBuzzerState = 1;
buzStat.cbBuzzerOnDuration = 10; // 1 second
status = AS_SetBuzzer(nDevId, &buzStat);

return status;
```

3.8.2 AS_SetLed

This function enables or disables any of the LEDs of the ACR88.

```
AS_STATUS AS_DECL AS_ SetLED (
                        IN      INT nDevId,
                        IN      PLED pLed);
```

Parameters:

nDevId

[in] Handle returned by a previous call to AS_Open.

pBuzzer

[in] Pointer to a *LED* structure that contains the state to set of the LEDs of the ACR88. See also chapter 2.2.10 LED for more information about the *LED* structure.

Return Values:

AS_STATUS

This functions returns different values depending on whether it succeeds or fails. AS_STATUS.DllError contains the status as returned by the DLL. AS_STATUS.W32Error contains the Win32 error code associated with the DLL error, if any. See also Appendix A for the possible return codes.

Example:

```
AS_STATUS       status;
LED             ledStat;

//turn on the LEDs and give them a different color
//assumed is that a connection has already been established
ledStat.cbLedPower = LED_RED;           // Set the Power LED to red
ledStat.cbLedSlot1 = LED_GREEN;// Set the Slot1 LED to green
ledStat.cbLedSlot2 = LED_YELLOW;        // Set the Slot1 LED to yellow
status = AS_SetLED(nDevId, &ledStat);
```

Appendix A. Error Codes (DLL Errors)

Only DLL Errors are listed below. For details of Win32 Errors, please refer to MSDN.

ErrorCode (Hex)	Error Description
0x00	CMD_SUCCESS
0x01	CMD_WARNING_BUFFER_OVERFLOW
0x02	CMD_ERROR_INVALID_OPTION
0x03	CMD_ERROR_INVALID_PARAMETER
0x04	CMD_ERROR_INVALID_RESPONSE_TYPE
0x05	CMD_ERROR_INVALID_PARAMETER_LENGTH
0x06	CMD_ERROR_LCD_INVALID_BITMAP_FILE
0x07	CMD_ERROR_LCD_LOAD_BITMAP_FILE
0x08	CMD_ERROR_LCD_INVALID_BITMAP_SIZE
0x09	CMD_ERROR_BUFFER_TOO_SMALL
0x0A	CMD_ERROR_BUFFER_ALLOCATION_FAILED
0x0B	CMD_ERROR_COMM_PORT_OCCUPIED
0x0C	CMD_ERROR_COMM_PORT_CANNOT_OPEN
0x0D	CMD_ERROR_COMM_PORT_NOT_OPENED
0x0E	CMD_ERROR_COMM_PORT_WRITE
0x0F	CMD_ERROR_COMM_PORT_READ
0x10	CMD_ERROR_COMM_DLL_GET_SYSTEMPATH
0x11	CMD_ERROR_COMM_DLL_FAILED_LOAD
0x12	CMD_ERROR_COMM_DLL_LOCATE_FUNCTION
0x13	CMD_ERROR_COMM_DLL_GET_DEVINFO
0x14	CMD_ERROR_COMM_DLL_INSUFF_BUFFER
0x15	CMD_ERROR_COMM_DLL_GET_DEVDETAIL
0x16	CMD_ERROR_COMM_NO_DEVICE_FOUND
0x17	CMD_ERROR_SCRIPT_INVALID_FILE
0x18	CMD_ERROR_SCRIPT_CANNOT_LOAD
0x19	CMD_ERROR_TFM_UNSUPPORTED
0x20	CMD_ERROR_SYSTEM_BUFFER_TOO_SMALL